UNIVERSITY OF SOUTHAMPTON

Decentralised Control of Wireless Sensor Networks

by

Johnsen Kho

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science
Intelligence, Agents, Multimedia Group

April 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Johnsen Kho

Wireless sensor networks are receiving a considerable degree of research interest due to their deployment in an increasing number and variety of applications. However, the efficient management of the limited energy resources of such networks in a way that maximises the information value of the data collected is a significant research challenge. To date, most of these systems have adopted a centralised control mechanism, but from a system's perspective this raises concerns associated with scalability, robustness, and the ability to cope with dynamism. Given this, decentralised approaches are appealing. But, the design of efficient decentralised regimes is challenging as it introduces an additional control issue related to the dynamic interactions between the network's interconnected nodes in the absence of a central coordinator.

Within this context, this thesis first concentrates on decentralised approaches to adaptive sampling as a means of focusing a node's energy consumption on obtaining the most important data. Specifically, we develop a principled information metric based upon Fisher information and Gaussian process regression that allows the information content of a node's observations to be expressed. We then use this metric to derive three novel decentralised control algorithms for information-based adaptive sampling which represent a trade-off in computational cost and optimality. These algorithms are evaluated in the context of a deployed sensor network in the domain of flood monitoring. The most computationally efficient of the three is shown to increase the value of information gathered by approximately 83%, 27%, and 8% per day compared to benchmarks that sample in a naïve non-adaptive manner, in a uniform non-adaptive manner, and using a state-of-the-art adaptive sampling heuristic (USAC) correspondingly. Moreover, our algorithm collects information whose total value is approximately 75% of the optimal solution (which requires an exponential, and thus impractical, amount of time to compute).

The second major line of work then focuses on the adaptive sampling, transmitting, forwarding, and routing actions of each node in order to maximise the information value of the data collected in resource-constrained networks. This adds additional complexity because these actions are inter-related, since each node's energy consumption must be optimally allocated between sampling and transmitting its own data, receiving and forwarding the data of other nodes, and routing any data. Thus, in this setting we develop two optimal decentralised algorithms to solve this distributed constraint optimization problem. The first assumes that the route by which data is forwarded to the base station is fixed (either because the underlying communication network is a tree, or because an arbitrary choice of route has been made) and then calculates the optimal integration of actions that each node should perform. The second deals with flexible routing, and makes optimal decisions regarding both the sampling, transmitting, and forwarding actions that each node should perform, and also the route by which this data should be forwarded to the base station. The two algorithms represent a trade-off in optimality, communication cost, and processing time. In an empirical evaluation on sensor networks (whose underlying communication networks exhibit loops), we show that the algorithm with flexible routing delivers approximately twice the quantity of information to the base station compared to the algorithm with fixed routing. However, this gain comes at a considerable communication and computational cost (increasing both by a factor of 100 times). Thus, while the algorithm with flexible routing is suitable for networks with a small numbers of nodes, it scales poorly, and as the size of the network increases, the algorithm with fixed routing should be favoured.

Contents

List of Figures

List of Tables

List of Algorithms

Declaration of Authorship

I, Johnsen Kho, declare that the thesis entitled Decentralised Control of Wireless Sensor Networks and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published in a number of publications (see section 1.4 for a list).

Signed:

Date:

Acknowledgements

I would like to express my sincere appreciation and gratitude to Professor Nicholas R. Jennings and Dr. Alex Rogers for their invaluable supervision, guidance, and support throughout my PhD. This thesis would not have been possible without their continuous encouragement and patience. I would also like to thank Long Tran-Thanh for his contributions towards collaborative work.

I am also thankful for the research scholarship, facilities, and support provided by the School of Electronics and Computer Science throughout my time at the University of Southampton. I also wish to thank the people of the Intelligence, Agents, Multimedia Group who have created such an enjoyable and constructive working environment over the past three and a half years and who have offered their feedbacks on my work and helped improve it. Finally, I would also like to thank my parents, sisters, girlfriend, and friends for their endless support, love, and care.

Nomenclature

General

$\mathbb{Z}^+$	Set of positive, non-zero integer numbers (i.e. $\mathbb{Z}^+ = \{1, 2, \dots\}$).
$\mathbb{N}$	Set of natural numbers with zero (i.e. $\mathbb{N} = \{0, 1, 2, \dots\}$).
$\Re$	Set of real numbers including both rational numbers and irrational numbers.
$\Re^+$	Set of positive, non-zero real numbers.

Chapter 2

$I = \{1, \dots, n\}$	Set of sensors (or nodes).
i	i^{th} sensor in I.
χ	Target location.
z_i	Sensor i's predicted observation.
$p(\chi)$	Prior target location distribution.
$q(\chi)$	Probability distribution after a new observation.
$H(\chi \vert z_i)$	Expected conditional entropy of posterior target location distribution.
$p(z_i \vert \chi)$	Sensing models of candidate sensor i for selection.

Chapter 3

$I = \{1, \dots, n\}$	Set of agents (or nodes) within a WSN system.
i	i^{th} node in I.
n	Number of nodes in I.
$H = \{1, \dots, w\}$	Set of time slots within a day.
w	Number of time slots in H.
$C_i = \{c_i^1, \dots, c_i^{s_i}\}$	Set of possible sampling rates for node i over a period of time.
s_i	Number of different sampling rates for node i in C_i.
c_i^j	j^{th} sampling rate of node i in C_i (i.e. a positive integer describing the number of times node i samples during a time slot).
$c_i^{s_i}$	Maximum sampling rate of node i in C_i.
$Alloc_i = \left\{a_i^1, \dots, a_i^w\right\}$	Allocated set of sampling actions (or schedules) for node i.
a_i^k	Allocated sampling action for node i at time slot k.
$Y_i = \{y_i^1, \dots, y_i^{g_i}\}$	Set of noisy observations (or target values or samples) collected by node i.
y_i^j	j^{th} samples in Y_i.
$X_i = \{x_i^1, \dots, x_i^{g_i}\}$	Set of sampling (or training) points (corresponding to Y_i) collected by node i.

x_i^j	j^{th} sampling point in X_i.
g_i	Number of observations made by node i in Y_i.
g_i^j	Number of observations collected by node i by performing its sampling action, c_i^j.
B_i	Energy budget of node i.
e_i^{s}	Energy required by node i in order to take a sample.
$v : \mathcal{V} \rightarrow \mathcal{Val}$	Valuation function (a.k.a utility or mathematical function) used to calculate the value or goodness of a certain action taken by nodes.
$\mathcal{V}$	Set of node's alternative actions.
$\mathcal{Val}$	Set of possible outcomes (corresponding to $\mathcal{V}$) that is typically represented by a set of numerical values.
$\mathfrak{X} = \{\mathfrak{x}^1, \ldots, \mathfrak{x}^m\}$	Set of a node's test points.
$\mathfrak{x}^j$	j^{th} test point in $\mathfrak{X}$.
m	Number of test points in $\mathfrak{X}$.
$\widehat{\mu}(\mathfrak{X}) = \{\widehat{\mu}(\mathfrak{x}^1), \ldots, \widehat{\mu}(\mathfrak{x}^m)\}$	Set of mean values of the posterior predictive distribution inferred by GP regression after it has observed both the training and target sets (X and Y respectively), at the test set, $\mathfrak{X}$.
$\widehat{\mu}(\mathfrak{x}^j)$	j^{th} mean value in $\widehat{\mu}(\mathfrak{X})$.
$\widehat{\sigma}^2(\mathfrak{X}) = \{\widehat{\sigma}^2(\mathfrak{x}^1), \ldots, \widehat{\sigma}^2(\mathfrak{x}^m)\}$	Set of variance values of the posterior predictive distribution inferred by GP regression after it has observed both the training and target sets (X and Y respectively), at the test set, $\mathfrak{X}$.
$\widehat{\sigma}^2(\mathfrak{x}^j)$	j^{th} variance value in $\widehat{\sigma}^2(\mathfrak{X})$.
$\mathcal{C}$	$g \times g$ matrix for the training set covariances.
$\mathfrak{c}$	$g \times 1$ vector identifying the training-test set covariances.
$\mathfrak{c}^{\text{T}}$	Transpose of $\mathfrak{c}$.
$C(\mathfrak{x}^j, \mathfrak{x}^j)$	Covariance of $\mathfrak{x}^j$.
I_g	$g \times g$ identity matrix.
σ^2	Added Gaussian noise of the training set.
$FI(X)$	Fisher information value calculated using GP regression over the interval spanned by $\mathfrak{X}$ of m test points, conditioned on the set of sensor readings, Y, taken at times X.
$C_{sqe}(x, \mathfrak{x})$	Squared exponential covariance function describing the correlation between any training point, x, and any test point, $\mathfrak{x}$.
λ_{sqe}	Length scale (or correlation length) for the squared exponential covariance function.
v_{sqe}	Weighting term for the squared exponential covariance function.
$C_{per}(x, \mathfrak{x})$	Periodic covariance function describing the correlation between any training point, x, and any test point, $\mathfrak{x}$.
$\mathfrak{p}$	Periodicity of the training set.
λ_{per}	Length scale (or correlation length) for the periodic covariance function.
v_{per}	Weighting term for the periodic covariance function.
X_i^*	Subset of sampling points of node i (satisfying its energy constraint) that maximises the Fisher information value over the set of test points, $\mathfrak{X}$, conditioned on the set of its readings, Y_i, taken at times X_i.
$b^1(X_i)$	Slope variable found by node i using a simple linear regression technique on its sets of data points and values, X_i and Y_i respectively.

$b^0(X_i)$	Intercept variable found by node i using a simple linear regression technique on its sets of data points and values, X_i and Y_i respectively.
$\hat{y}_i^j$	Newly found y_i^j data value using the slope and intercept variables.
$\bar{y}_i$	Mean value of Y_i.
$\bar{x}_i$	Mean value of X_i.
$Sd(X_i)$	Standard deviation found by node i under a simple linear regression technique on its sets of data points and values, X_i and Y_i respectively.
$\tau_i^t(X_i)$	Distance of the confidence bands from the simple linear regression line (calculated by node i using a simple linear regression technique) at point x_i^t, conditioned on its set of sensor readings (or data values), Y_i, taken at times (or data points) X_i.
$Td^{g_i}(k)$	Area between the confidence bands that represents the total deviation (or uncertainty) over the set of g_i data points, X_i, and the set of data values, Y_i, collected by node i in time slot k.
$Gain_i^j(k)$	Reduction in total deviation that node i can achieve by taking samples at rate c_i^j (and, hence, collecting g_i^j samples) rather than the minimum sampling rate, c_i^1, in time slot k.
G_i	Table that node i computes to determine its set of allocated sampling actions, $Alloc_i$. The table consists of s_i number of rows and w number of columns. The element of the table in j^{th} row and the column with label k has the value of $Gain_i^j(k)$.
v_i^{jk}	Value (in matrix V_i) that node i will get if it chooses to perform sampling action, c_i^j, in time slot k (i.e. $Gain_i^j(k)$).
d_i^{jk}	Decision variable (in matrix D_i) where "1" represents a state where node i carries out the corresponding c_i^j action in time slot k, whilst '0" represents another state where the node does not carry out the corresponding c_i^j action.
D_i^*	Decision matrix of node i that maximises the Fisher information collected while satisfying the node's energy constraint over a day.
$Bc_i(k)$	Energy consumed by node i per time slot k.
β_i	Cut-off threshold of node i's energy supply (or battery).
$Bmax_i$	Full battery capacity of node i.
$B_i(k)$	Energy remaining of node i at time slot k.
Bh_i	Preset energy increase available each day, provided by node i's energy harvester (or solar panel).
$ns = df$	Beginning of night period or end of day period.
$nf = ds$	End of night period or beginning of day period.
$Bn_i(t)$	Preset energy increase of node i at any time, t, during the night (that is from ns till nf).
Bd_i	Total energy recharging of node i during the day (that is from ds till df).
$\delta_i(k)$	Thickness of the cloud above node i during time slot k.
$Bp_i(k)$	Energy recharging rate of node i during time slot k.
$\mathsf{t}(k)$	Beginning of time slot k.
$\mathsf{t}'(k)$	End of time slot k.
ci	Confidence interval parameter used in USAC.
$Lo(ci)$	Lower bound of ci.
$Up(ci)$	Upper bound of ci.
$Sr_i(t)$	Sampling rate of node i at time t dictated by USAC.

$dat(t+1)$	Next predicted data calculated by USAC.

Chapter 4

$I = \{1, \ldots, n\}$	Set of agents (or nodes) within a WSN system.
i	i^{th} node in I.
n	Number of nodes in I.
$C_i = \{c_i^1, \ldots, c_i^{s_i}\}$	Set of possible sampling rates for node i over a period of time.
s_i	Number of different sampling rates for node i in C_i.
c_i^j	j^{th} sampling rate of node i in C_i (i.e. positive integer describing the number of times node i samples during a certain period of time).
$c_i^{s_i}$	Maximum sampling rate of node i in C_i.
$\mathfrak{F}_i = \left[(0,0), \left(c_i^1, v_i^{c_i^1}\right), \ldots, \left(c_i^{s_i}, v_i^{c_i^{s_i}}\right)\right]$	Private set of tuples, known only to node i, each of which consists of: (i) the sampling action, c_i^j, that the node may perform, and (ii) the corresponding information content of the samples acquired, $v_i^{c_i^j}$. The first tuple represents that should the node choose to acquire no samples, it will yield zero information value.
B_i	Energy budget of node i.
e_i^{s}	Energy required by node i in order to take a sample.
e_i^{Tx}	Energy required by node i in order to transmit a sample.
e_i^{Rx}	Energy required by node i in order to receive a sample.
E_i^{S}	Energy required by node i in order to *sense* (i.e. sample and transmit) a sample of its own.
E_i^{F}	Energy required by node i in order to *forward* (i.e. receive and transmit) a sample of other nodes.
$R(c_i^j) = (r_i^1, \ldots, r_i^b)$	Vector of nodes, each of which the samples (collected by node i with its c_i^j sampling action) route through in order to arrive at the base station.
r_i^l	l^{th} node in vector $R(c_i^j)$.
$\mathfrak{d}_i^{R(c_i^j)}$	Decision variable where "1" represents a state where node i carries out its c_i^j sampling action and the collected samples follow the vector route, $R(c_i^j)$, to arrive at the base station, and "0" represents the state where the node does not carry out its c_i^j sampling action.
$\mathfrak{D}_i^*$	Decision variables of node i that maximises the information content of the forwarded samples while satisfying the node's energy constraint.
D_i	Set of neighbourhood nodes of node i.
f_i	Total number of incoming samples received by node i from its set of neighbour, D_i.
q_i	Total number of unique routes from node i to the base station.
α_i	Weighting factor of node i.
$Cmax_I$	Set of 3-tuples indicating for each node in network I, the sensing and forwarding actions that it should perform and the specific route that it should use to transmit each bundle of its own and forwarded data to the base station.
$\mathfrak{O}_i = \left[\left(b_i^1, Vmax_i^1, Cmax_i^1\right), \ldots, \left(b_i^{K_i}, Vmax_i^{K_i}, Cmax_i^{K_i}\right)\right]$	Meta-data array of node i, each element of which is a 3-tuples.
b_i^k	k^{th} energy limit of node i.

K_i	Number of feasible b_i^k-s.
$Vmax_i^k$	Maximum information value that node i can transmit to its parent by using at most b_i^k energy and taking into account its own and others' forwarded data.
$Cmax_i^k$	Set of actions that will produce the data with value $Vmax_i^k$.
p_i	Unique parent node of node i (only applies in the context of fixed routing algorithm).
B_{p_i}	Energy budget of node i's parent node, p_i.
$E_{p_i}^{\mathrm{F}}$	Energy required by node i's parent node, p_i, to *forward* (i.e. receive and transmit) a forwarded sample.
$J_i = \left\{ j_i^1, \ldots, j_i^{M_i} \right\}$	Set of node i's child nodes.
j_i^m	m^{th} child node of node i in J_i.
M_i	Number of node i's child nodes in J_i.
$\mathfrak{D}_{j_i^1}, \ldots, \mathfrak{D}_{j_i^{M_i}}$	Meta-data arrays of node i's child nodes in J_i.
$Vmax_{j_i^m}^k$	Maximum information value that node i's child node, j_i^m, can transmit to node i, and that can subsequently be relayed to node i's parent node, p_i, by using at most b_i^k energy.
$\mathfrak{D}_i[x]$	x^{th} element of $\mathfrak{D}_i$.
T_i	Table that node i derives in order to determine its meta-data array, $\mathfrak{D}_i$, by taking into account only its energy limits, b_i^k-s.
$T_i[x, y]$	Element of table T_i that is in the x^{th} row and the column with label b_i^y.
b_i^l	l^{th} energy limit of node i's parent node, p_i.
L_i	Number of feasible b_i^l-s.
U_i	Table that node i derives in order to determine its meta-data array, $\mathfrak{D}_i$, which will be sent to its parent node, by taking into account both its own and its parent's energy limits (b_i^k-s and b_i^l-s respectively).
$U_i[x, y]$	Element of table U_i that is in the x^{th} row and the column with label b_i^y.
$Cmax_i^l$	Set of actions that will produce data with the value of $U_i\left[M_i, l\right]$, which is the maximum information value that node i can transmit to its parent node (by using at most b_i^k energy). The parent node can then forward the received i's data by using at most b_i^l energy.
$\mathfrak{C}_i$	Set of node i's descendant nodes.
$\vert\mathfrak{C}_i\vert$	Number of node i's descendant nodes.
$\mathfrak{P}_i = \left\{ p_i^1, \ldots, p_i^{\mathfrak{N}_i} \right\}$	Set of node i's parent nodes.
$p_i^{\mathfrak{n}}$	$\mathfrak{n}^{\mathrm{th}}$ parent node of node i in $\mathfrak{P}_i$.
$\vert\mathfrak{P}_i\vert$	Number of node i's parent nodes in $\mathfrak{P}_i$.
$\mathfrak{L}_i$	Set of possible partitions of forwarding at node i, given $\vert\mathfrak{C}_i\vert$ (+1 of its own) bundles of samples and $\vert\mathfrak{P}_i\vert$ different shortest paths to choose from for each of the bundles.
$l_i^t = \left[F\left(u_i^1\right), \ldots, F\left(u_i^{\vert\mathfrak{C}_i\vert+1}\right) \right]$	t^{th} partition of forwarding at node i in $\mathfrak{L}_i$.
u_i^j	j^{th} bundle that might arrive at node i from one of its descendants in $\mathfrak{C}_i$.
$F\left(u_i^j\right)$	Mapping function that decides the forwarding direction (or path) for bundle u_i^j.
$\mathfrak{D}_{j_i^m}^i$	Meta-data array arriving at node i from its child node, j_i^m in J_i.
$\mathfrak{D}_i^{p_i^{\mathfrak{n}}}$	Meta-data array of node i's for its parent node, $p_i^{\mathfrak{n}}$ in $\mathfrak{P}_i$.

$\mathfrak{O}_i^{p_i^{\mathfrak{n}}(l_i^t)}$	Meta-data array of node i's for its parent node, $p_i^{\mathfrak{n}}$ in $\mathfrak{P}_i$. However, this particular meta-data only contains a partition of forwarding, l_i^t in $\mathfrak{L}_i$.
$U_i^{p_i^{\mathfrak{n}}}$	Table that node i derives in order to determine its meta-data array, $\mathfrak{O}_i^{p_i^{\mathfrak{n}}}$, for its parent node, $p_i^{\mathfrak{n}}$ in $\mathfrak{P}_i$.

Chapter 1

Introduction

A *wireless sensor network* (WSN) is here defined as an array of small, locally battery-powered sensor nodes that wirelessly communicate information sampled from events in a surrounding environment to a base station (a.k.a. sink or gateway) via a wireless communication channel (see figure 1.1 for an illustration). Information that arrives at the base station is then transmitted outside of the network to be processed further for the purpose of fulfilling the goals of WSN deployment. In most cases, a base station is commonly a specialised node that has significantly more power than the ordinary ones and, moreover, there may be more than one should system robustness be a key concern.

In WSNs, each wireless sensor node (a.k.a. mote) in the network is equipped with a sensor module that enables it to capture signals generated by the events through a sensor channel, and a network stack that enables it to send information to the base station through a wireless channel (see figure 1.2). The combination of the sensor module, the network stack (including a radio frequency transceiver with its typical operational states of *transmit*, *receive*, *idle* and *sleep*), the sensor application interface (i.e. the interface available for a programmer to specify the sensor's behaviour), the processing or micro-controller module, and multiple types of memories (program, data, or flash memories for storage capability) is usually called the sensor function model (Raghunathan et al., 2002). Additionally, the node will typically have a power model containing the energy-producing components (e.g. a battery or a solar panel) and the energy-consuming components (e.g. the radio transceiver, the CPU micro-controller, and the sensor module) that enable it to perform its functions.

Now, WSNs, composed of a number of these wireless nodes, have recently generated significant research interest within the academic literature of computer science and electronic engineering as they have many advantages over their wired counterparts, particularly due to their flexibility and ease of deployment. They are, therefore, becoming increasingly prevalent in a wide variety of applications ranging from environment and habitat monitoring, structural health surveillance, smart buildings, home automation,

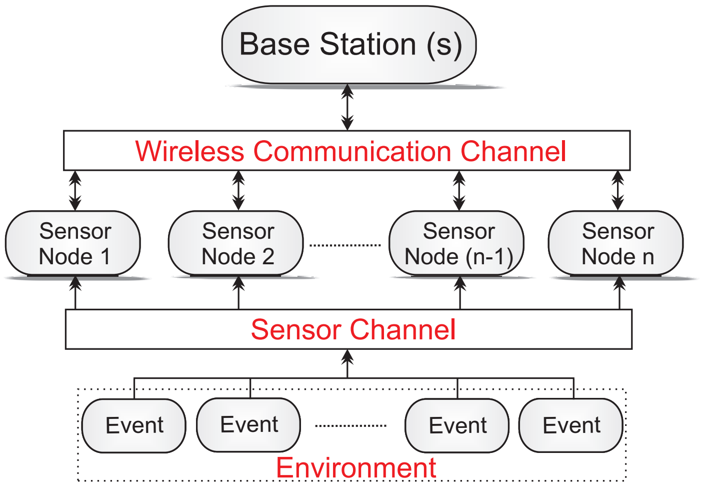


FIGURE 1.1: Typical WSN environment.

object tracking, area surveillance, traffic control, to other security and health related applications (see chapter 2 for more details). Within this context, this thesis focuses on developing algorithms for controlling such networks in a decentralised manner. That is, algorithms that give the flexibility for each node in the networks to locally adapt its sampling, transmitting, receiving, and routing behaviour (by driving its sensor and network stack modules) without any external coordinator.

To this end, we begin in this chapter by describing the basic background to our work and outlining the research requirements and contributions of this thesis. More specifically, section 1.1 identifies the main research challenges in the domain of WSNs particularly as this pertain to control. In section 1.2, we then describe two types of control regime in such networks to overcome these challenges. From the preceding discussion, we then motivate our research and identify the requirements of our work in section 1.3. This is followed by an overview of our research contributions to the state-of-the-art in section 1.4. Finally, in section 1.5, we give the overall structure of the remainder of this thesis.

1.1 Wireless Sensor Network Research Challenges

Within the WSNs domain, there are a number of significant research challenges to be addressed. These include real-world integration, real-time performance, cost and size of sensor nodes, security and privacy, power management, and programming abstraction (Chong and Kumar, 2003; Stankovic, 2004). In this thesis, however, we focus in particular on the challenges associated with energy management. This is of critical importance since it allows WSNs to survive long term operation in their deployed environment with minimal human intervention. Moreover, it dictates the amount of useful information that can be gathered over the lifetime of each node. Unfortunately, in many cases, the nodes have limited energy storage capabilities, and this energy is depleted rapidly. For

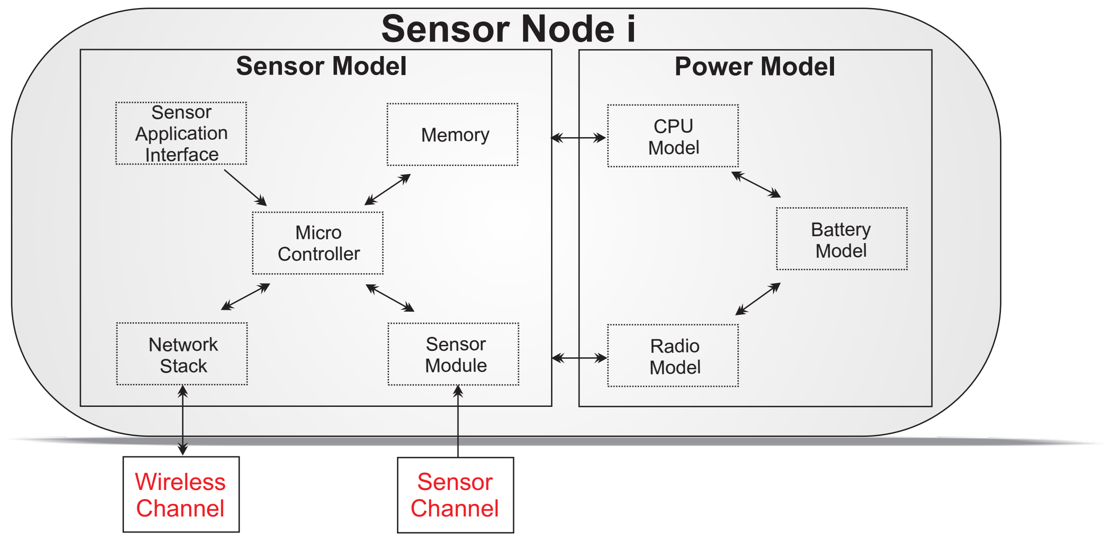
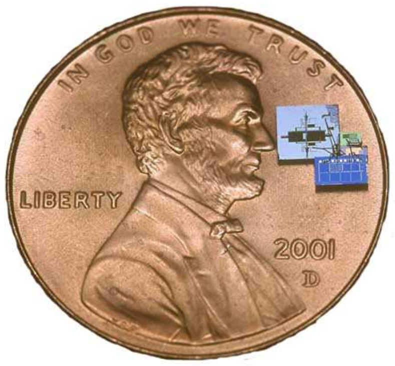


FIGURE 1.2: Internal architecture of a wireless sensor node.

example, many devices, such as the Golem Dust or Mica2Dot[1] nodes (as shown in figure 1.3), use small batteries as an energy storage. Depending on their activity level, their lifetime may only be a few months or even weeks if no efficient power management schemes are used. Since most deployed systems require a much longer lifetime, significant research has been undertaken to increase the lifetime of WSNs while still meeting their functional requirements. So far, there are two primary perspectives from which solutions have been proposed; namely (i) hardware and (ii) software.

Within the hardware context, an over-engineering process, such as installing a bigger and more powerful battery, helps improve the longevity of the nodes and, in turn, the network's lifetime (Shinozuka et al., 2004). Advances in chip manufacture have successfully reduced the power consumption of nodes in order to improve their longevity (Ekanayake et al., 2004; Hempstead, 2005; Kleihorst et al., 2006). Moreover, in some WSN applications, an additional facility is available to ameliorate the energy problem by using rechargeable batteries and energy harvesting technologies such as solar panels, wind turbines, piezo-electric harvesters, or other transducers (Meninger et al., 1999; Jiang et al., 2005; Raghunathan et al., 2005). In such cases, the harvester's energy output varies with time depending on the environmental conditions, and these are obviously outside the control of the network's designer. Hence, the available environmental energy at each node location may not be the same. A further advantage of these technologies is that a node which has exhausted its battery may start operating again in the next available energy harvesting opportunity. However, these approaches can incur greater costs in terms of the hardware's production, operation, and maintenance.

[1]Mica2Dot are produced by Crossbow Technology Incorporation. The node's specification can be viewed from `http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2DOT_Datasheet.pdf`. A list of prototype and other commercial sensor nodes available, can be found from `http://en.wikipedia.org/wiki/Sensor_node`. Both links are checked on 02/02/2009.



(a) Golem Dust (taken from (Warneke and Pister, 2004)).

(b) Mica2Dot.

Figure 1.3: Wireless sensor motes.

Another technique which adopts a hardware solution aims to reduce the power consumption of the sensor nodes by designing nodes with discrete transceiving power levels and dynamic radio ranges so that power is not wasted through transmitting further than is required (Yoon et al., 2004). This technique can be critical in reducing a node's power consumption since the power to reliably transmit over a distance d is proportional to the m^{th} power of this distance (i.e. power $\approx d^m$), where $m = 2$ in a simple deterministic radio model (Mhatre and Rosenberg, 2004). In more complicated models, where the power to transmit also depends on the transmission medium, m can be as low as 1, or as high as 3 (Akyildiz et al., 2002). Transmission is worse, for instance, when it needs to penetrate various forms of obstructions including walls, water, or other objects. In this thesis, we are not focusing on hardware, however, we have endeavored to use realistic models for building a WSN simulator in order to analyse our algorithms (detailed in chapters 3 and 4).

From the software perspective, a number of researchers have been examining how the individual nodes can be made more effective and intelligent so that they can perform in a more energy efficient manner. Specifically, the two main actions that such intelligent sensor nodes can vary in order to improve their energy management are to adapt: (i) their *sampling* and (ii) their *communication* capabilities. In addition to sampling and communicating its own data to other nodes, a sensor node also acts as a router, forwarding data messages for others. Thus, researchers have aimed to adopt effective and efficient *adaptive sampling* and *adaptive routing* policies so as to achieve the network's goals, while using the minimum energy resources possible.

In more detail, the former policy includes sets of rules that adapt a node's *sampling rate* (i.e. *how often* a node is required to sample during a particular time interval) and *schedule* (i.e. *when* a node is required to sample). On the other hand, the latter contains sets of rules that determine the way a node transmits data back to the base station, including: (i) *transmitting* (i.e. *how many* samples of its own, a node is required to transmit during a transmission period), (ii) *forwarding* (i.e. *how many* samples of other nodes, a node is required to receive and forward during a transmission period), and (iii) *routing* (i.e. *which* routes a node should choose to transmit its own and forwarded samples through) decisions. Given the energy issue of limited power availability, another substantial challenge in a WSN deployment, and also a focus in our work, is to maximise the information value of the data collected at the base station. Thus, considerable research has been undertaken in this area and much work is targeted at this intelligent protocols field (for more details, see chapter 2). Another energy management technique in the literature is data compression. However, the effectiveness of the compression techniques is highly dependent on the processing power of the nodes. Here, the total computational overhead increases as both the source and destination node, now, have to spend some processing energy in order to compress and decompress data accordingly (Kimura and Latifi, 2005; Dong et al., 2006). Given this, this thesis will focus on the adaptive sampling and routing option.

With regard to figure 1.2, our proposed intelligent energy management algorithms can later be installed inside the micro-controller module to drive the sensor module and the network stack. Specifically, each node inside a network will therefore detect an event (based on its adaptive sampling protocol) from the environment. It then passes this information data to its controller for more intelligent processing, such as deciding how and which of the sensed data should be reported back to the base station (that is in accordance with its adaptive routing protocol).

Now, within this context, the challenge is to achieve these objectives given the distinguishing characteristics of WSNs, including:

- **Physical Constraints:** The nodes are often highly constrained in their communicational, computational, energy, and storage resources, and hence they need to manage these in the most efficient manner possible.
- **Dynamism:** The environment being monitored is typically highly dynamic and events within it cannot readily be forecast a long time in advance. Furthermore, the network topology can sometimes vary during operation since nodes may fail, be moved, or be added at any time.
- **Deployment in Hostile Environments:** The communication links between nodes are subject to additional noise and interference. The dynamism-related problems are also further aggravated within hostile and inaccessible environments as the nodes are more likely to fail.
- **Scalability:** Since the nodes are often relatively small and inexpensively mass produced, they tend to be deployed in large numbers and at high densities in order to produce high data rates and fidelity. Moreover, it is often not possible

to build a global addressing scheme as the overhead of identifier maintenance is typically high.

- **Correlated Sensor Readings:** With high spatial densities of sensor nodes in the deployed area, nodes often sense the same events from their environment and, thus, forwarding the same data to the base station can result in redundant information. Intelligent local processing can be used to prevent correlated events being sampled and transmitted in which some energy resources will be wasted.
- **Real-World Integration:** Many WSN applications are monitoring real-time phenomena in which time and space are of an extreme importance.

Given these characteristics, there are two main types of *control regime* (i.e. a protocol dictating the actions of the sensor nodes contingent on their states and the observations they make) that can be deployed in a WSN; namely *centralised* and *decentralised.* We will compare and contrast them in the following section.

1.2 Centralised Versus Decentralised Control

Both centralised and decentralised control are typically designed to intelligently use the nodes' limited resources in order to meet the objectives set out by the owner of the network. However, in the former, a single coordinator node, usually the base station, receives data from all the nodes, computes the actions to be taken by these nodes, and then issues commands to all the nodes indicating how they should sample, transmit, forward, and route data. In contrast, in the latter case such a central coordinating node does not exist. Instead, the nodes are autonomous and each decides its individual actions based on its own local state and observations, and those of its parent and children nodes.

In this thesis, we focus on a decentralised regime for controlling the nodes' behaviour in such networks. We do so because it increases the system's robustness compared to its centralised counterpart (Lesser, 2003). Specifically, the robustness is increased by reducing bottlenecks in decision processing due to the absence of the central node. In contrast, centralised control is often not a viable option because the centre may have to do an infeasibly large number of computations (as a result of the scale) and the individual actions dictated by the centre may not be efficient (as a result of the dynamism).

The design of an efficient decentralised control regime is, however, not straight forward. By removing the centre, the decentralised approach introduces an additional control issue related to the dynamic interactions between the interconnected nodes in the network. Given this, it is often far from obvious how the individual node's processes (which when taken together constitute the decentralised control) need to be designed such that their interactions can meet the overall design objectives. This is because it is difficult to check that the solution found by the distributed system is globally optimal since no single node has a global view. Furthermore, it is typically hard to predict the global system behaviour of distributed systems as their components' interactions often give unpredictable results.

To address these issues, we adopt an agent-based approach in which each sensor node is represented by an autonomous agent that acts and interacts in an autonomous fashion in order to achieve its individual and collective objectives (Jennings, 2001). The distributed and dynamic nature of the complex, interconnected, and rapidly changing WSN systems lend themselves to a *multi-agent system* (MAS) methodology in which the individual agents need to coordinate their activities cooperatively in a distributed manner towards achieving system-wide goals. Thus, in the following section we outline the research requirements that we aim to address in order to build such a system for a WSN.

1.3 Research Requirements

Though most of the existing WSN applications (of which each considers only tens of homogeneous nodes) have been revolutionary in replacing their traditional wired counterparts, WSN technology generally still remains at a relatively immature stage (Habib et al., 2007; Karl and Willig, 2007). However, with the imminent introduction of much cheaper wireless sensor nodes (due to the rapid increases of their computational power, memory capacity, wireless speed, and sensing technologies), we believe that future WSN applications will involve much larger scale deployments with heterogeneous nodes of different functions and characteristics, some of which are possibly mobile for more dynamic coverage (Kahn et al., 1999; Chen and Ma, 2006; Oh et al., 2006).

The increase in performance of hardware components is, however, not followed by that of the nodes' batteries which historically, has only doubled in capacity every ten years (Chalamala, 2007). Despite the significant advances in the energy density of rechargeable batteries and the energy scavenging technologies of solar panels or wind turbines, there are no near-term solutions for the widening energy gap (see figure 1.4). As a result, the need to efficiently manage these power resources will still remain of critical importance. Moreover, the large scale deployment of WSNs will additionally raise concerns about scalability as most of the currently implemented systems rely on a central coordinator to operate effectively. Such systems are likely to become infeasible in governing many future real-world applications. To remedy this, we believe it is desirable to add autonomous capabilities for self-configuring and self-organizing at the network's deployment stage, self-adapting during any dynamic changes, self-sustaining throughout any component's failures, and self-optimizing in order to achieve the deployment objectives. Our rationale here is that it will be intractable for the central coordinator to find the optimal actions of a large number of nodes (of which each has its own private action space) in a timely

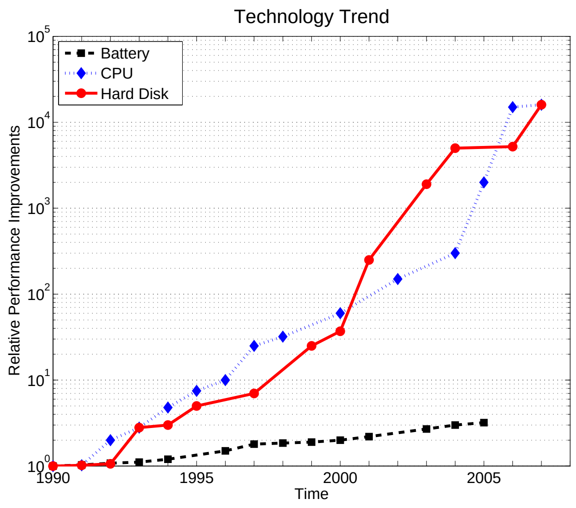


FIGURE 1.4: Relative performance improvements in the energy capacity of lithium-ion batteries (Matsushita Batteries) versus the storage capacity of hard disk drives (Hitachi Storage Technologies) and the computational power of microprocessors (Intel Corporation). This graph is reproduced from (Chalamala, 2007).

manner, regardless of its processing power (as explained in the previous section).

Therefore, we aim to develop a distributed, scalable, robust, and efficient energy-aware algorithm that is computationally applicable to a wide range of WSNs (of which most will involve larger scale deployments in future applications). The algorithm will intelligently adapt the sampling, transmitting, and forwarding actions that each node in the network should autonomously choose to perform, and also the route by which each item of sensed data should be forwarded to the base station, such that the total information collected at the base station is maximised, given the constraints of limited energy resources imposed on the nodes to perform their processing, sensing, storing, transmitting, or receiving activities. With this in mind, we would like to meet the following design objectives:

1. **Information Metric:** We require a principled approach that is based on a non-parametric model, to value various temporally correlated sensor observations (typically with noise and/or uncertainty). The metric will allow a node to make individual comparisons between different alternative actions and also comparisons between multiple nodes' actions. Moreover, in our exemplar FLOODNET environmental sensor network (more details of which can be found in chapter 2), hydrologists might not be just interested in the water-level depth at sampling points, but also at times when no reading occurs. In such cases, we observe that although many information valuation functions that are based on a parametric model often generate excellent predictions (e.g. the linear techniques for regression in Padhy et al. (2006) and Hastie et al. (2009)), there is usually no notion of uncertainty about these predictions. With this notion, however, we are likely to have better reading estimates at points near the actual collected sampling points, compared

to any others. This is because we are more certain that the predicted readings will have similar values to those of the contiguous sampling points.

2. **Decentralised Algorithms:** The algorithms using this information metric to control WSNs, should be distributed (i.e. decentralised); that is, there should be no central coordinator that has a partial or complete knowledge about the network in order to decide which node (or agent) does which action (Wagner and Wattenhofer, 2007; Boukerche, 2008). This is because a centralised approach suffers from a lack of robustness, scalability, and ability to cope with the network's dynamism since the central controller is required to perform a large number of computations in order to find each node's optimal actions. There is also the communication difficulty of providing all the relevant system states to the central coordinator in a timely manner. Additionally, such a controller increases dependence and vulnerability as a greater number of nodes rely on such a single processor that, if it were to fail, could potentially cause catastrophic failure of the whole system. Therefore, with the distributed algorithms, the nodes should be capable of taking autonomous local actions that benefit the overall system.

3. **Complete Algorithms:** The distributed algorithms should be able to find a solution (preferably an optimal one) if one exists, otherwise it should correctly report that no solution is feasible. Many incomplete distributed algorithms are available in the literature (see chapter 2 for a list), however, they are not desirable to be implemented on energy constrained wireless nodes as they drain a significant amount of power, particularly when the nodes are unable to converge to a stable state.

4. **Inter-Dependent Actions:** The algorithms should be able to deal with the *inter-dependent* actions of sampling, transmitting, forwarding, and routing. In a multi-hop routing scenario, a node located far from the base station requires the help of intermediate nodes to relay its data messages. The intermediate nodes therefore should consider allocating some of their energy to *relay* messages (i.e. forward or receive and transmit messages that are not its own) rather than selfishly spend all their resources sampling and transmitting their own data frequently (Tang et al., 2006; Chatterjee and Das, 2008). These local decisions are inter-dependent as no node in the network is able to make such a decision individually. This is because the sampling, transmitting, forwarding, and routing choices of an individual node will affect those available to all of the other nodes in the network.

5. **Scalability with Real-Time Performance:** In general, the distributed algorithms should be scalable to handle WSNs with large number of nodes. More specifically, we expect our algorithms to handle systems with hundreds or even thousands of nodes, each of which possesses private state or values initially known only to itself (for instance, project ExCal by Arora et al. (2005) have deployed a 1000+ node WSN). Therefore, the whole process of the algorithms should be

computationally feasible and have minimal memory requirements since nodes are usually limited in their power resources to drive their processing and storage modules.

6. **Minimum Communication:** Communication between the nodes, as well as the size of the coordination message being communicated, should be minimized. This is particularly important when the nodes have limited communication bandwidth. Moreover, communication in a sensor network has an inherited delay and could be unreliable in many situations, particularly, in hostile environments where WSNs are predominantly being deployed. Therefore, a good distributed algorithm should require a small number of message exchanges. Most of the distributed algorithms in the literature, however, require an exponential increase in the total number of messages being exchanged, and this is unrealistic for WSNs in which the nodes are typically installed with limited energy resources to receive and transmit messages.

7. **Dynamism:** The algorithms should adapt to the network's dynamism which includes nodes' failure (due to, for example, lack of power, physical damage, or environmental interference) or addition. The states of each node (e.g. its sampling rate, battery recharging rate, forwarding, and routing actions) vary over time and, therefore, the algorithms should also be reactive and proactive by responding accordingly to these changes while minimising the disruption to the overall process (Ulema et al., 2006).

8. **Node Heterogeneity:** In many studies, all sensor nodes are assumed to be homogeneous and to have equal capacity in terms of computation, communication, and power. However, sensor nodes are currently offered with varying resources, capabilities, and thus are more likely to have different roles since their characteristics can vary considerably (Lu et al., 2008). Hence, it is desirable if the algorithms make no assumptions about the set of nodes among which the algorithms are to be distributed.

1.4 Research Contributions

Against the research requirements outlined above, this thesis makes a number of contributions to the state-of-the-art:

1. We develop a novel generic information metric for sensor networks. This metric represents the temporal variation in the environmental parameter being sensed as an unknown function, and then uses *Gaussian process* (GP) regression to infer the characteristics (specifically its temporal correlation and periodicity) and value of this function, over a continuous interval, conditioned on samples made at discrete times within the interval. We then use the mean Fisher information over the entire

interval (including periods between which sensor samples were taken) as a measure of the information content of these actual sensor samples. Thus, informative sensor samples are those that minimise uncertainty in the value of the environmental parameter over the entire interval (i.e. a set of samples with the highest value is able to generate any sets of estimated readings with the least uncertainty at any times within the interval).

2. Using this information metric, we describe three novel decentralised control algorithms for information-based adaptive sampling which represent a trade-off in computational cost and optimality. The first uses GP regression within each sensor node to optimise the time at which a constrained number of future sensor readings should be taken. This process is exponential in the number of sensor readings taken and, thus, the second algorithm we present again uses GP regression within the sensor nodes, but performs a greedy approximate optimisation in order that it is more computationally tractable. Finally, we further reduce the computational cost by using a heuristic algorithm within each sensor node, rather than the GP regression, in order to select the times at which future sensor readings should be taken.

3. In order to ground and evaluate this approach, we need to exercise it in a particular domain and here we choose flood monitoring and, in particular, the FLOODNET sensor network. In this setting, the heuristic algorithm is empirically shown to increase the value of information gathered by approximately 83%, 27%, and 8% per day compared to benchmarks that sample in a naïve non-adaptive manner, in a uniform non-adaptive manner, and using a state-of-the-art adaptive sampling heuristic (USAC), correspondingly. Furthermore, it provides information whose total value is approximately 75% the optimal solution (which requires an exponential and, thus, impractical amount of time to compute).

4. We develop a novel decentralised algorithm that varies each sensor's inter-dependent sampling, transmitting, and forwarding rates (or actions) to ensure all nodes in a network focus their limited resources on maximising the information content of the data collected at the base station. This algorithm assumes that the route by which the data is forwarded to the base station is *fixed* (either because the underlying communication network is a tree or because an arbitrary choice of route has been made). This algorithm is scalable (running on thousands of nodes in real-time) since by utilizing distributed dynamic programming techniques, it allows each node to make computationally tractable and optimal local decisions regarding its integration of actions, which are influenced by its own local circumstances, and those of its parent and children.

5. Using the same distributed dynamic programming technique to extensively truncate the search space of potential allocation decisions, we extend the fixed route algorithm to deal with *flexible* routing. This allows each node to make optimal

decisions regarding both the inter-related sampling, transmitting, and forwarding actions that each node should perform, and also the route by which data should be forwarded to the base station.

6. In order to ground and evaluate these algorithms, we empirically evaluate them and show that they represent a trade-off in optimality, communication cost, and processing time. In more detail, we show that when deployed on sensor networks whose underlying communication networks exhibit loops, the algorithm with flexible routing is able to deliver approximately twice the quantity of information to the base station compared to that of the algorithm using fixed routing. However, this gain comes at a considerable communication and computational cost (increasing both by a factor of 100 times). Thus, while the algorithm with flexible routing is suitable for networks with a small numbers of nodes, it scales poorly, and as the size of the network increases, the algorithm with fixed routing should be favoured.

Specifically, these contributions have been published in:

- J. Kho, A. Rogers, and N. R. Jennings. Decentralised Adaptive Sampling of Wireless Sensor Networks. In *Proceedings of the 1st International Workshop on Agent Technology for Sensor Networks, a Workshop of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007.

  This workshop paper describes a distributed adaptive sampling algorithm (part of Contribution 2) which uses a simple information metric based on the uncertainty values of the regression line of actual FLOODNET sensor samples. The paper then evaluates the algorithm and shows that it collects information that has significantly less uncertainty error than a uniform non-adaptive approach (part of Contribution 3).

- J. Kho, A. Rogers, and N. R. Jennings. Decentralised Control of Adaptive Sampling in Wireless Sensor Networks. *ACM Transactions on Sensor Networks*, 5(3), 2009 (In Press).

  This article describes a novel principled information valuation of sensor readings based on Fisher information and GP regression (Contribution 1), and details the theoretical optimal, greedy, and an extended version of the decentralised heuristic adaptive sampling algorithm presented in Kho et al., (2007) (Contribution 2). The article then evaluates each algorithm in the FLOODNET setting and provides empirical results that the heuristic algorithm performs better than a number of benchmarks by offering the best trade-off between the computational cost and optimality (Contribution 3).

- J. Kho, L. Tran-Thanh, A. Rogers, and N. R. Jennings. Decentralised Control of Adaptive Sampling and Routing in Wireless Visual Sensor Networks. In *Pro-*

ceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems, May 2009 (In Press).

This short paper briefly highlights the research challenges in wireless visual sensor networks due to the inherent computational, storage, and power resources in their wireless smart camera nodes, and highlights the need of distributed algorithms in order to optimize the nodes' collected data at the base station by adapting their sampling and routing actions (part of Contributions 4 and 5).

- J. Kho, L. Tran-Thanh, A. Rogers, and N. R. Jennings. Distributed Adaptive Sampling, Forwarding, and Routing Algorithms for Wireless Visual Sensor Networks. In *Proceedings of the 3rd International Workshop on Agent Technology for Sensor Networks, a Workshop of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2009 (In Press).

  This paper details the adaptive sampling, transmitting, and forwarding algorithm for fixed routing in tree-structured networks (Contribution 4), and another algorithm for flexible routing in networks with any topologies (Contribution 5). The paper then empirically evaluates them and shows that albeit collecting more information value, the latter algorithm increases the communication and computational cost significantly compared to the former (Contribution 6).

1.5 Thesis Structure

The remainder of this thesis is structured as follows:

- In chapter 2, we provide a literature review of WSN applications in general and the chosen FLOODNET domain testbed in particular. We then describe a decentralised paradigm for controlling such networks using an agent-based approach. Following that, we conduct a thorough survey of current research in information metrics, adaptive sampling, adaptive routing, and inter-related adaptive sampling and routing algorithms that are available in the literature. We also discuss their limitations against the requirements that we placed earlier in this chapter and, thus, motivate the work that we present later in the proceeding chapters. Finally, we remark what models and approaches we can build upon and where further improvements are needed.

- In chapter 3, we describe the intuition of information-based adaptive sampling, and provide a formal model of a WSN system. We then show how we use a GP package to calculate our Fisher information value since this will allow the information content of a node's observations to be expressed. Following that, we formulate the three decentralised control algorithms that maximise the Fisher information metric for solving the adaptive sampling problem. In the same chapter, we conduct a thorough empirical evaluation and show that the heuristic algorithm collects an information value close to the optimal one and significantly more than those collected by existing approaches. More importantly, it runs in real-time and, hence, is scalable for large scale networks.

- In chapter 4, we begin by presenting an exemplar scenario where inter-related adaptive sampling, transmitting, forwarding, and routing problems typically occur. We then formalize the problem by extending our system model from the previous chapter. Following that, we detail the two distributed optimal algorithms for solving the problem. The first assumes fixed routing and works in tree-structured networks. The second works in networks with any topologies by using a flexible routing approach. Finally, we evaluate them and show that the algorithm with flexible routing is suitable for networks with a small numbers of nodes as it is scalable to only tens of nodes. For large-scaled networks, the algorithm with fixed routing is better.

- We conclude this thesis in chapter 5 by focusing on a summary of our research contributions, and finally outline future work that can be carried out to extend and enhance the proposed algorithms. This is followed by appendix A which lists a log of FLOODNET nodes' data and appendix B which enumerates the acronyms we use throughout this thesis.

Chapter 2

Literature Review

In this chapter, we discuss the existing literature in the area of adaptive sampling and routing algorithms, and we highlight the limitations of each of these algorithms and, thus, motivate the research in this thesis. To this end, in section 2.1, we begin by outlining the background of WSN applications in general and introducing the FLOODNET domain in particular. In section 2.2, we then describe, in general terms, decentralised approaches for controlling such networks using agent-based modelling. In section 2.3, we provide a background review on information valuation metrics as this is central to the development of information-based adaptive algorithms. In sections 2.4 and 2.5 respectively, we detail some of the most commonly used adaptive sampling and adaptive routing algorithms that have been developed for WSNs. In section 2.6, we then discuss existing inter-related adaptive sampling and routing algorithms. Finally, we conclude in section 2.7 by summarising our findings and relating them back to our original research requirements (as detailed in section 1.3).

2.1 Wireless Sensor Network Applications

WSNs, each of which consists of a set of wireless sensor nodes, are beginning to be deployed at an accelerated pace for an increasing array of applications. Many of them have been demonstrated in *environmental and habitat monitoring* applications (Mainwaring et al., 2002; Cardell-Oliver et al., 2005; De Roure, 2005; Padhy et al., 2005; Werner-Allen et al., 2006). Some have been installed in civil structures to continuously *monitor buildings' structural health* such that the networks can proactively determine the safety of the structures following a natural disaster for example (Chintalapudi et al., 2006). Additionally, WSNs are becoming increasingly popular for *monitoring spatial phenomena* such as the temperature distribution in a building and controlling heating, ventilation, and air conditioning (Reinisch et al., 2007; Krause et al., 2008). The military has also used WSN technology for *battlefield and area surveillance* or counter-sniper systems to detect and accurately *track sniper shooters' locations and movements* in urban environments (Simon et al., 2004; Ledeczi et al., 2005; Bramberger et al., 2006; He et al., 2006). Moreover, the emergency and healthcare services are starting to use pervasive *health monitoring* technology of intelligent wearable sensors, known as the Body Sensor Network, on patients under rehabilitation in order to improve the quality of the provided care (Kroc and Delic, 2003; Rahman and Shabana, 2006). Sensors could also be implanted in the human body to monitor medical problems like cancer (Lo and Yang, 2005).

Now, such networks of battery-powered sensor nodes that wirelessly communicate information sampled from the environment to a base station are becoming ever more popular because they have many important advantages over their wired counterparts, including:

- **Cost Effectiveness:** Pervasive and ubiquitous computing technologies allow cheap nodes in wireless networks to incur less costs in terms of setup, operation, and communication (Zachary, 2003). In a wired network, however, the cost of wiring sometimes greatly exceeds the cost of the nodes (Rabaey et al., 2000).
- **Extreme Deployment Environments:** Wireless nodes can be deployed in inhospitable, extreme outdoor conditions where humans are normally prevented from accessing such harmful sites (means that it would be impractical or often impossible to set up a wired network).
- **Scale, Size, and Availability:** Large numbers of cheap and failure-tolerant small wireless nodes can be scattered to cover a large geographical sensing area. Installing and configuring a few hundred nodes in a wired network, on the other hand, would be extremely time-consuming and error-prone.
- **Dynamic Coverage:** Wireless nodes can provide a greater level of flexibility by allowing the possibility of having mobile nodes that move around the environment being sensed in order to provide dynamic coverage. Such dynamism is typically impossible within a wired network.

Within this thesis, we focus in particular on *environmental wireless sensor networks* (EWSNs). We do this because the environment is a key application area for sensor networks and because there is considerable local expertise and access to deployed systems. Examples of relevant (non-local) EWSNs include a volcano monitoring project that has deployed sparse arrays of high spatial separation nodes in Volcan Tungurahua in central Ecuador to support volcanoes studies (Werner-Allen et al., 2006), a bird habitat monitoring system on several remote islands on the coast of Maine (Mainwaring et al., 2002), a soil moisture and temperature monitoring project in Australia (Cardell-Oliver et al., 2005), a hydrological environmental monitoring system in Tucson, Arizona (Delin et al., 2004), a habitat monitoring project in cryogenic environments in Antartica (Delin et al., 2003), and a soil monitoring operation that has developed an underground wireless network of soil sensors to improve farming while minimizing environmental impacts in central Iowa (Huang et al., 2008).

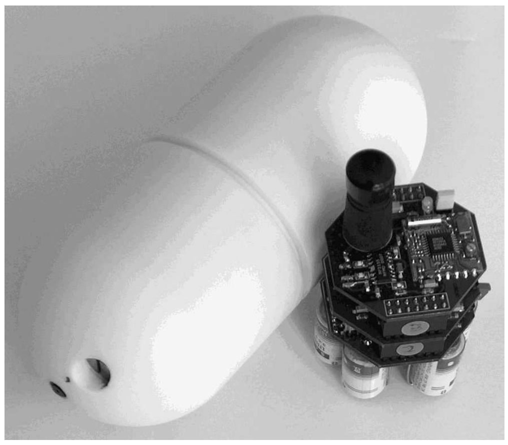
We now turn to two EWSNs of the University of Southampton deployments; namely (i) GLACSWEB (because it is considered to be the most relevant background work for which a similar adaptive sampling and communication algorithm to ours has been devised), and (ii) FLOODNET (because we are using it as our testbed domain to test our proposed mechanisms). The GLACSWEB project (Padhy et al., 2005) has produced and placed a few tens of nodes inside the Briksdalsbreen glacier in Norway in order to monitor the glacier movement and behaviour. The study site on the glacier was chosen as it was flat and crevasse-free with safe access. The system operated from August 2004 to August 2006. Due to the glacier's retreat, GLACSWEB researchers redeployed the nodes at Skalafellsjokull glacier, a part of Vatnajokull National Park in Iceland in 2008. The FLOODNET project (De Roure, 2005) has also developed and deployed several wireless sensor nodes to form a WSN system that monitors the water level in the River Crouch, East Essex in Eastern England. The site was chosen for its tidal behaviour such that, for test purposes, there are regular variations in the water level. These variations are particularly important for generating and testing an improved hydraulic model and flood simulator which enables environmental experts to predict floods in advance (Neal et al., 2007).

2.1.1 The GLACSWEB System

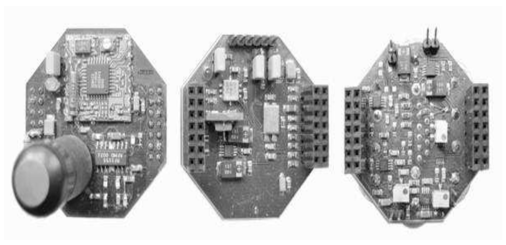
The aim of GLACSWEB is to develop a wireless network of autonomous probes (i.e. sensor nodes) that could be inserted into the Briksdalsbreen glacier's ice and till (or the subglacial sediment). These probes should be able to collect several environmental parameters related to the glacier's behaviour that are useful for subglacial dynamics and send them back to the surface via radio communications without disturbing the surrounding environment. The collected data will then be accessed world-wide in near real-time by researchers via the Internet in order to better understand climate change involving sea level change due to global warming and, eventually, to act as a vital environmental hazard warning system. Glaciologists believe it is important to monitor how glaciers contribute to releasing fresh water into the sea as this activity could cause rising sea levels and great disturbances to the thermohaline circulation of sea water (Braithwaite and Raper, 2002).

In more detail, each of GLACSWEB nodes (shown in figures 2.1(a) and 2.1(b)) is around 12cm in length. More specifically, it comprises of six main components; namely (i) *a micro-controller* (of a PIC18[1] family which controls the entire node's system including

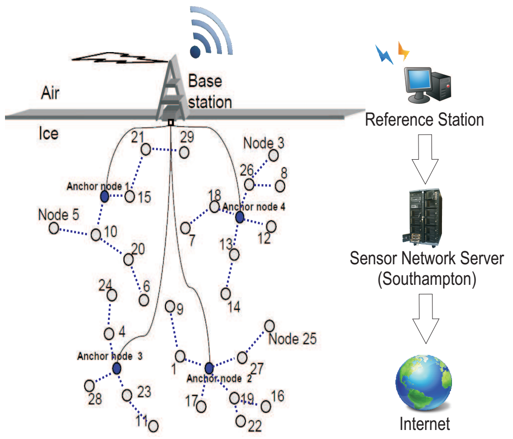
[1]`http://lis.epfl.ch/contest/flying07/docs/resources/PIC 18XXXX.pdf`, checked on 02/02/2009.



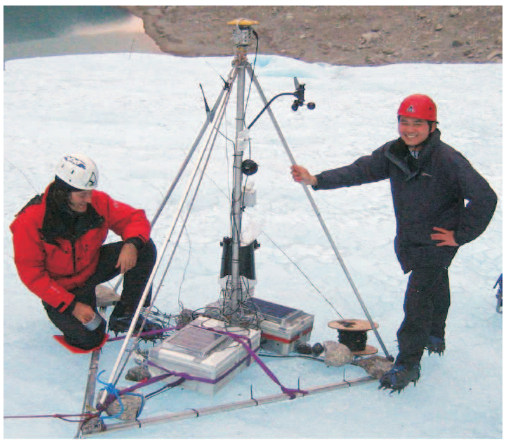
(a) The hardware and the shell (taken from (Martinez et al., 2006)).



(b) The hardware parts in modules (taken from (Martinez et al., 2006)).



(c) The deployed GLACSWEB system configured using a Time Division Multiple Access protocol (reproduced from (Elsaify et al., 2007)).



(d) The base station (taken from (Padhy et al., 2005)).

FIGURE 2.1: GLACSWEB WSN.

reading (i.e. sampling) and running the node's radio transceiver modules in order to transmit or receive collected readings), (ii) *a real-time clock*[2] (to control the node's wake up and sleep time), (iii) *storage* (with an on-board 128KByte memory that is used to store programs as well as the communication control and sensors' sampled data), (iv) *sensors* (that include an orientation, a water pressure, a temperature, a resistivity, a case stress, a humidity, and a light reflection sensor), (v) *a radio communication system* (with a 433MHz helical antenna and a power amplifier), and (vi) *a power controller and supply* (that is supported by three AA-sized lithium batteries of which each provides 2.25Ah). All these six hardware parts are bundled inside a specifically designed water-proof white tube-shaped polyester case.

The node runs in 5 different modes; namely (i) *sleep mode* (in which all its components are turned off apart from the clock; this requires only 105μJ/s of energy to operate), (ii)

[2]`http://www.maxim-ic.com/products/rtc/real-time-clocks.cfm?`, checked on 02/02/2009.

sampling mode (in which sensors are on to take samples, demanding 0.3J per sample), (iii) *transmission mode* (in which only the node's transmitter module is switched on to transmit samples sent in packets, taking 0.5J per packet), (iv) *receiving mode* (in which only the node's receiver module is on to receive samples, requiring 0.15J per packet), and (v) *idle transceiver on mode* (taking only 0.1J per second). Each packet is made up of 64Bytes.

Glaciologists from the University of Southampton have deployed these nodes within the glacier (see figure 2.1(c)). Now, due to the significant radio losses in the upper ice layer, the nodes in the glacier are unable to reliably communicate with the base station on the surface. Four anchor nodes are therefore connected directly to the base station via a serial cable (represented by solid lines in the figure). They act as cluster-heads responsible for communicating with the other nodes in their own sub-networks. The base station (as shown in figure 2.1(d)) is responsible for collecting data from the nodes. It also transmits the collected data via a radio modem down the valley to a reference station (with a PC on a main power), from which the data is sent to a web server in Southampton for daily back-up and made accessible from the Internet.

GLACSWEB originally adopted a centralised regime to control its system whereby each GLACSWEB node is centrally programmed (in its real-time clock) to have fixed wake up and sleep schedules and, hence, fixed data collection and transmission times. The clock dictates the node to take samples at four hour intervals and to communicate them to the base station every twelve hours. During the communication period, the nodes enable their radio transceivers for a fixed duration. The base station also powers up during this period, broadcasts the system time, and requests undelivered sensor readings from the nodes. The reference station then wakes up to receive data transfer from the base station. All the data that has been recorded is then transferred to the data server in Southampton during a transfer period (at nineteen hour intervals).

To combat the limitations introduced by controlling such a WSN in a centralised manner, Padhy et al. (2006) have recently developed a decentralised control to adaptively vary the sampling actions and routing behaviour of each GLACSWEB node (for more details, see sections 2.4 and 2.6 for the adaptive sampling and the adaptive routing aspect of the control respectively). The decentralised algorithms, however, are not complete and efficient since they are loosely coupled and this can sometimes result in no data collection at the base station (see section 2.6 for a more detailed discussion).

2.1.2 The FLOODNET System

Having described the existing work in GLACSWEB, we now review the FLOODNET project. In terms of background, flood damage in the UK rose from £1 billion lost in 1999, to nearly £1.3 billion two years later. In 2007, it was measured that flood losses approached nearly £3 billion (Harman, 2007; Pitt, 2008) and it could possibly hit £16 billion with the impact of climate change in 50 years time (Haddrill, 2007). Even worse, around 5 million people, in 2 million properties, currently live in flood risk areas in the UK. Now, this risk of flooding is increasing significantly for a number of reasons including land-use change, climate change, flood-prone investment, and the defect of timely flood warning for those people under threat.

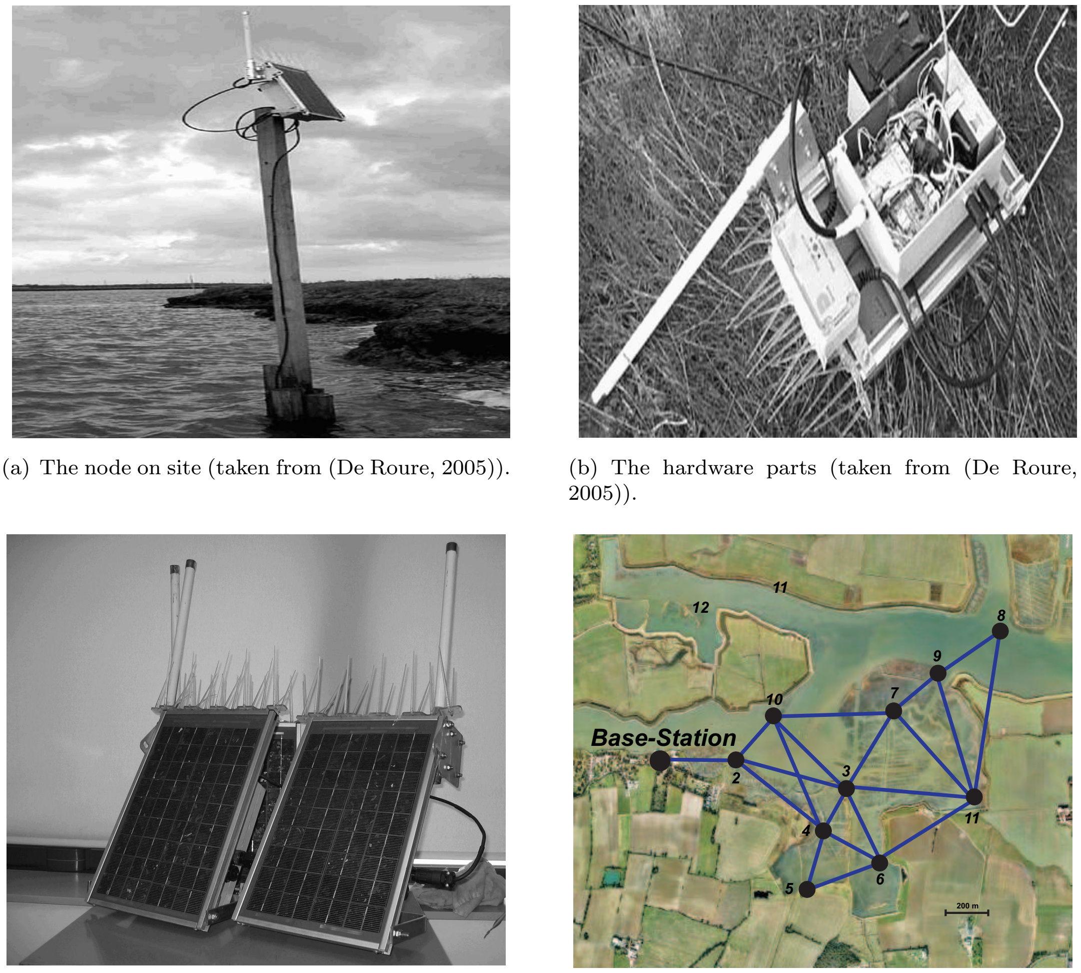
Given its importance, we choose flood detection as our target domain, and given its accessibility, we choose FLOODNET as the specific sensor network. Moreover, in comparison with GLACSWEB's, FLOODNET nodes are much more powerful in terms of computational and power resources. This, in turn, allows us to deploy an agent inside each node in order to compute all its own computational local actions using our mechanism.

The ultimate aim of FLOODNET is to provide early warning of flooding such that actions can be taken to alleviate risks to people and property. This early and precise warning is crucial as there is a clear correlation between the cost of damage and both the time in advance any warnings are given and the depth of the flooding. To this end, the FLOODNET system is currently deployed to gather precise tide height readings to enable a calibrated hydrological model of the deployment area to be constructed. The network must withstand long term unmanned operation without any significant human intervention as nodes are deployed at a number of hostile and not easily accessible locations where periodic data collections also might not be possible particularly in extreme environment conditions (for instance during floods). The network thus incurs less costs in terms of setup, operation, and communication compared to equivalent data logging devices.

Within the FLOODNET domain, the flood predictions can be simulated based upon the input sets of the water level data taken periodically from the site. However, there is a resource cost associated with collecting, storing, processing, and utilising sensed data. It is therefore important that the sampling and routing framework used to acquire the data maximises the delivery of information to the available forecasting models, given the limited energy resources available.

In more detail, FLOODNET consists of twelve nodes, each of which (shown in figures 2.2(a) and 2.2(b)) is based around a BitsyX[3] *Single Board Computer* (SBC) that uses an Intel 400MHz PXA255 RISC microprocessor. Since the SBC consumes a significant amount of power (1000mW) when providing field processing capabilities, it is in sleep mode for most of the time. The board provides support for PCMCIA where a wireless LAN PC card (Conexant NL-2511CD Plus Ext2) is installed to send and receive data wirelessly from the neighbourhood nodes (requiring an additional 910mW and 640mW of power respectively, however, in power safe mode, the card only consumes 90mW of

[3]`http://www.applieddata.net/products_bitsyX.asp`, checked on 02/02/2009.



(a) The node on site (taken from (De Roure, 2005)).

(b) The hardware parts (taken from (De Roure, 2005)).

(c) The node's solar panel.

(d) The deployed FLOODNET system.

Figure 2.2: FLOODNET WSN.

power). Each node is also installed with a sensor module which consists of a PIC16F88-I/P PIC (that is used for data processing), an AD8544 A/D, an AD8541 A/D (both are used as Analog-to-Digital converters), and a PDCR1830 water-depth transducer sensor (which is used to make tide height measurements, requiring another additional 50mW of power when activated). The PIC and converters are always turned on as they operate with tiny power consumption (requiring only 20mW of power). In addition, each node is equipped with a rechargeable lead-acid battery (FullRiver[4] HGL12-12 12V 12Ah/20hr) and a solar panel (BP[5] Solar BP SX 10U as shown in figure 2.2(c)), such that it can harvest solar energy to recharge the battery during the day.

Now, although the wireless transmission signal (under IEEE 802.11b 2.4GHz standard with up to 11Mbps bandwidth) of each FLOODNET node can cover a range of up to 1200 metres in an open space, the presence of obstructions such as sea walls, trees,

[4]`http://www.fullriver.com/products/admin/upfile/HGL12-12.pdf`, checked on 02/02/2009.

[5]`http://www.datataker.com/Library/Product_Data_Sheets_TS/Accessories/BP_Solar_SX10.pdf`, checked on 02/02/2009.

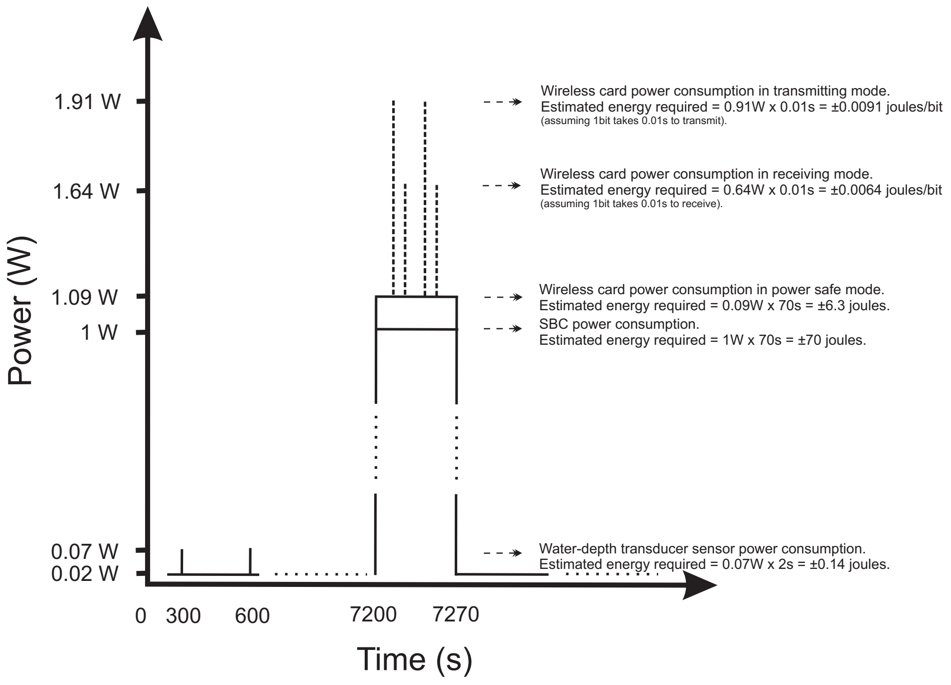


FIGURE 2.3: The battery consumption diagram of FLOODNET nodes.

and buildings at the FLOODNET field side, means that the nodes are only capable of communicating reliably over a maximum range of around 600 to 800 metres. Thus, most nodes do not have a direct link with the base station (see figure 2.2(d)). Nodes are numbered in the figure and able to directly communicate only with their close neighbourhood nodes (represented by lines). Node 1 is the base station while others are ordinary nodes. Each node that does not have any lines attached to it represents an old node that has already been removed. The nodes in the network have a single transmission level. Neighbourhood nodes within this transmission range can hear, receive, process, and re-broadcast the transmitted messages with the objective of transmitting the data back to the base station.

Having outlined the basics of FLOODNET system, we now consider the two types of regime that can be used to control it; the currently deployed centralised one, as well as our proposed decentralised one.

2.2 Agent-Based Decentralised Control

Many of the WSNs described in the previous section adopt a centralised control regime such that there is a central coordinator node, which is normally a base station or a node with much more computational, communication, and power resources. Such a node possesses detailed global knowledge about the state of the overall system (including the states of all the other nodes) and dictates which node in the network does which actions.

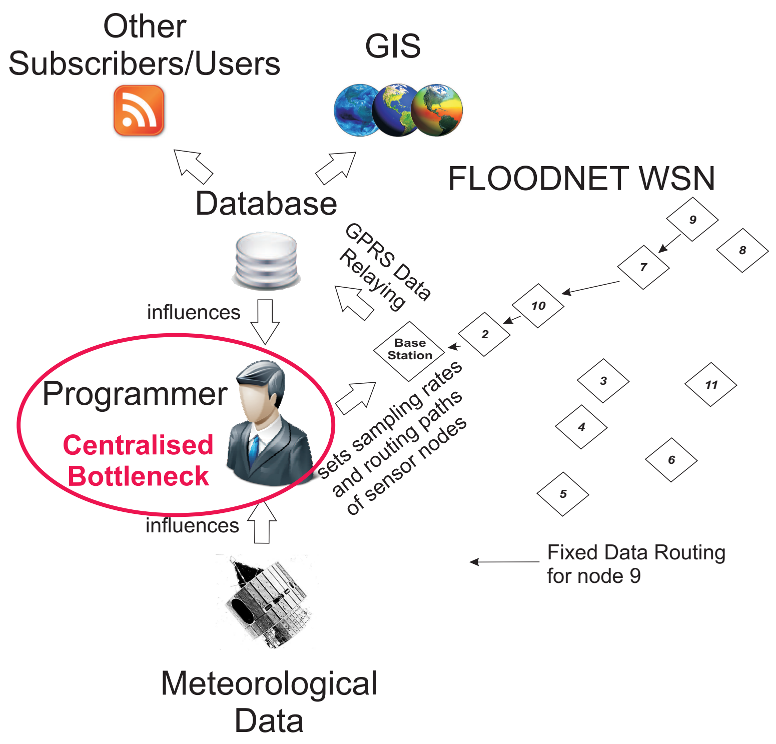


Figure 2.4: The centralised FLOODNET infrastructure.

Like many other similar applications, FLOODNET currently adopts a centralised regime to govern its system whereby each FLOODNET node is centrally programmed to have a controlled sampling rate and routing path. The node currently takes samples and stores them locally in its local memory at five minute intervals (in which the sensor is switched on for approximately two seconds). Each sample includes a mean water-depth measurement, a measurement variance estimate, the remaining battery power, and the time (see appendix A for a log of FLOODNET nodes' data). Following that, it activates its SBC with the corresponding transceiver modules every two hours for the purpose of transmitting the collected data to the base station via non-adaptive multi-hop routing. The measurements sampling process consumes a small amount of power (around 70mW) relative to that of the data transmission (around 1910mW), as shown in figure 2.3. The base station subsequently relays the data to a *Geographical Information System* (GIS) database using *General Packet Radio Service* (GPRS). Scientists from the Geography Department of the University of Southampton then use this incoming data within a hydraulic prediction model to make accurate and timely flood forecasts. At the same time, various parties who are interested in the data can as well subscribe to the incoming data stream (see figure 2.4).

The incoming sensed data (together with meteorological data) influences the programmer in setting the nodes' next sampling rates. Now, as outlined in section 1.2, this centralised approach to control has a number of drawbacks. This is because the existence of a central coordinator (i.e. the programmer) could slow down the performance of the system since it could be required to perform a large number of computations for determining the actions that should be taken by each node and, hence, could act as a catastrophic bottleneck (as shown in figure 2.4). Moreover, the centralised control regime could suffer from a lack of scalability and ability to cope with the network's dynamism and, therefore, is undesirable and often ineffective. On the other hand, if control and responsibilities are shared among nodes, the system can tolerate the failure of one or more of the nodes without overly affecting the overall performance (Marin-Perianu et al., 2007).

Now, there are occasions when the centralised control paradigm is appropriate (e.g. when the domain problem is reasonably static or when there are relatively few nodes in the network), but when the problem is dynamic then decentralised control systems are likely to be better. In these circumstances, decentralised (or distributed) strategies become attractive because they increase both the speed of the network's deployment and its robustness as communications no longer have to pass through a single point. It also reduces costs and places greater emphasis on smaller, cheaper computerised nodes, which are now more feasible because of the dramatic growth in computer power and resources (Zachary, 2003). Furthermore, the dynamic nature of future real-world WSN applications (including the node's failures or additions at any time and the limited knowledge of the topology issues) are likely to be better addressed by controlling the network in a decentralised manner.

Therefore, we remark that though FLOODNET is an exemplar of a small scale WSN and the currently implemented centralised control is still computationally tractable to adapt the actions of a small number of FLOODNET nodes in a timely manner, we would like to target our more robust and scalable decentralised approach to a wider range of WSNs, most of which will involve larger scale deployments. To illustrate the reason behind this, consider a scenario when the coordinator ceases and both nodes 8 and 9 in figure 2.4 currently sample at their highest rates and transmit their sampled readings according to a fixed path (i.e. both relay their messages through node 7 in order to get them to the base station). With no further instruction from the coordinator, both nodes will continue sampling at maximum rates and thus, rapidly deplete their energy resources. Node 9 will therefore be unlikely to be able to forward node 8's data readings in the near future. Moreover, node 8 will waste its energy by sampling data that will never get relayed to its destination. Under a decentralised regime, however, nodes 8 and 9 could autonomously adapt their sampling actions. Node 9 will also take into consideration the fact that its neighbour node 8 needs its help to relay data messages and hence, will pro-actively allocate a portion of its energy for this activity.

Our ultimate aim is therefore to remove the centralised point of control in FLOODNET system and deploy our decentralised information-based control of adaptive sampling, transmitting, forwarding, and routing algorithms within the FLOODNET nodes themselves. By doing so, the aim is to maximise the information value of the data collected at the base station, given the limited energy resources available (see the next section for

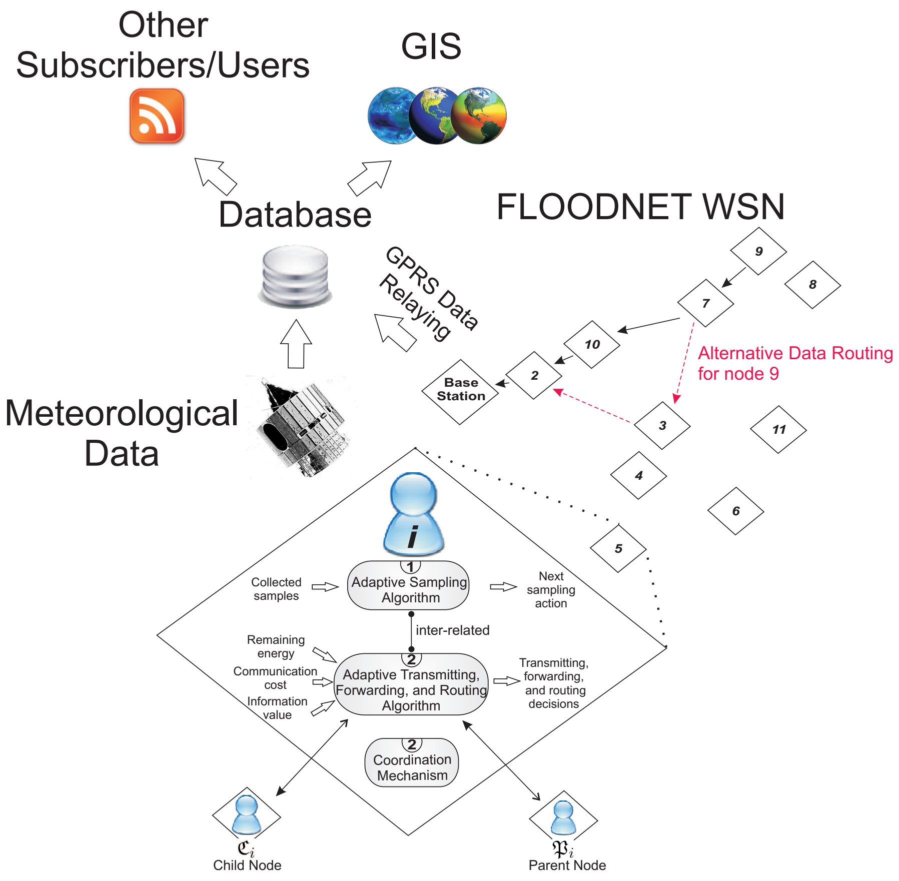


Figure 2.5: The decentralised FLOODNET infrastructure.

a number of information valuation metrics that are most commonly used to express the information content of sensor samples).

To this end, figure 2.5 shows the conversion of FLOODNET's centralised control regime to that of an agent-based decentralised one. Specifically, this is achieved by eliminating the central controller. We replace it by developing a distributed mechanism for *each node* in the network which provides simple local decision rules. The local decision rules give each node the ability to choose its decisions regarding its adaptive sampling rates adjustment (dictated by an adaptive sampling algorithm which is marked with number 1 in figure 2.5) and transmitting, forwarding, as well as next-hop routing decisions (dictated by an adaptive transmitting, forwarding, and routing algorithm together with the help of a coordination mechanism, both are marked with number 2 in the figure) based only on its local information and those of its parents and children. The coordination mechanism dictates how each agent (i.e. node) communicates with others and what minimum information is exchanged. Information provided to each node in order to enact the adaptive sampling policies includes only the node's past collected samples. Whilst that for the transmitting, forwarding, and routing policies includes its own and parents' residual battery power and communication costs, as well as the information value of the data its children and itself could potentially collect.

Having outlined how decentralised control will operate within the specific setting of FLOODNET, we now return to the more general setting of EWSNs. As discussed previously, in such settings, we consider each node within the WSN to be an autonomous intelligent *agent* because it has private information regarding its own state and it can decide for itself without any external coordinators, and the network itself as a multi-agent system because the individual agents need to coordinate their activities cooperatively towards achieving system-wide goals (that is, each agent is capable of reactive, social, and goal-directed behaviour) (Weiss, 2000; Wooldridge, 2002; Rogers et al., 2009). We believe this agent-based modelling is essential in controlling the nodes with their inherent distributed nature to select the best strategy to maximise the global objectives in accordance with other nodes. Within this context, each node must be able to decide for itself whether or not to perform an action on request from another node. This is different from the node being a passive *distributed object* which encapsulates some private states and has public methods corresponding to operations that other nodes are *always* allowed to perform on these states. There is, nevertheless, a similarity between distributed passive objects and reactive agents in a way that they both interact with others via message passing.

To illustrate this, consider the network scenario expected in figure 2.5 where node 7 is out of the base station's transmission range and hence, needs the help of either node 3, 9, or 10 to relay its recorded samples to the base station. As node 7 has social and proactive behaviour, it cannot attempt to achieve network goals by sending its data selfishly to the base station without taking others into account as these goals can typically be achieved only with the cooperation of others. Node 7 thus *coordinates* with others by exchanging coordination messages (more details of which can be found in section 4.4). Node 7 will consequently choose not to relay through node 9 which is one-hop further than itself to the base station. Assuming that node 10 has much energy remaining, it will be *willing* to cooperatively forward node 7's data. However, if later node 10 detects any dynamic events with high importance values in its readings, it will increase its sampling rate and this is likely to drain its energy budget rapidly. Following that, node 10 will reactively *stop* forwarding for others and *refuse* any further requests (as it is reactive to the changes in its environment). Node 7 will therefore change its routing path via node 3, which has its own *right* to refuse node 7's request if necessary in order to achieve global goals. In a more complicated setting (that we will consider as part of our future work), a node might compete with others in maximising its own utility and therefore, must be incentivised to take an action that benefits the overall system (see section 5.2 for more details).

Against this background, Corkill et al. (2007) have developed an agent-based, power-aware sensor network for atmospheric monitoring (named the *Collaborative Network for Atmospheric Sensing*). In their communication setting, data should be delivered within a certain period of time from the moment it is sensed, otherwise it will be useless. Agents (or nodes) must then coordinate with others to have their radios turned on when others are sending messages to them, otherwise some energy will be wasted due to dropped messages since most of the time, nodes are in their sleep state. In a WSN where energy is scarce, a clock synchronization protocol typically involves large message exchanges and induces extra processing, both of which can be very expensive (Sundararaman et al., 2005). Therefore, in our communication setting, we will assume that the transmission time and interval of nodes are fixed and pre-determined.

Horling et al. (2001) also adopt an agent-based paradigm to deal with the issues of coordination and reconfigurable sensors within their distributed sensor networks for real-time tracking in order to track one or more targets that are moving through the sensor environment. In their setting, the sensors are divided and placed into different sectors, each of which has an agent serving as the manager of the sector. The manager decides and organizes the scanning schedules of each node in its sector. In doing so, it must possess a complete information regarding the state of each node in its sector. The managers could therefore potentially act as bottlenecks as if any of them fails, this will overly affect the overall performance of the network. Their approach thus does not satisfy our Requirement 2 (as argued in section 1.3).

Within this agent-based approach, a useful technique that has emerged for solving multi-agent distributed coordination problems is that of *distributed constraint optimization*[6] (DCOP or DisCOP). To date, there has been a rich set of real-world distributed applications, such as in the area of networked systems, for which this technique has been used, particularly, in distributed applications where constraints exist among agents. Thus we explore this area further.

In more detail, a number of algorithms in the area of DCOP have been developed; these include *Synchronous Branch and Bound* (SBB) (Hirayama and Yokoo, 1997), *Distributed Breakout Algorithm* (DBA) (Yokoo and Hirayama, 1996), *Asynchronous Distributed Optimization* (ADOPT) (Modi et al., 2005), *Distributed Pseudotree Optimization Procedure* (DPOP) (Petcu and Faltings, 2005), *Asynchronous Partial Overlay* (APO) (Mailler and Lesser, 2006), and *Max-Sum* (Farinelli et al., 2008). Now, many of these algorithms are based on the *dynamic programming* method because it is often used to solve problems with distinguished properties of *overlapping subproblems*[7] and *optimal substructure*[8]. The method works by (i) breaking down the multi-agent problem into smaller subproblems, (ii) solving these subproblems optimally using this three-step process recursively, and (iii) constructing an optimal solution to the original problem by using these optimal

[6]DCOP is a distributed constraint optimization problem in which a group of agents (or nodes) must choose values (or actions) from a set of variables in a distributed manner (as each agent possesses its own private values), such that the actions maximises an objective function, while also satisfying the set of imposed constraints over the variables.

[7]A problem with the overlapping subproblems property means that it can be broken down into subproblems which are reused several times.

[8]A problem with the optimal substructure property means that the optimal solution to the problem can be constructed efficiently from optimal solutions to its subproblems.

solutions of the subproblems (Kellerer et al., 2004).

However, SBB is found to be ineffective due to its slow performance (and, hence, fails to meet Requirement 5). Moreover, it uses the notion of a virtual agent that acts as a central manager to deliver messages among agents. Now, due to the existence of this manager that is aware of all of the delivered messages' state, SBB does not meet Requirement 2 either. DBA is preferable when we want a near optimal solution much quicker than SBB, however, it may fail to return a solution even if one exists and also it cannot correctly determine if no solution exists (and, therefore, does not satisfy Requirement 3). To counter the limitations introduced by systematic, synchronous, and sequential message exchanges between nodes, ADOPT and DPOP are guaranteed to converge to the optimal solution while using only localized asynchronous communication and computation in the setting of a sensor resource allocation problem. However, since these algorithms are complete and asynchronous, they require an exponential increase in either the total message size being exchanged or the memory complexity on the agents. This is unrealistic for WSNs in which the nodes are typically installed with limited computational, storage, and memory resources (Requirements 5 and 6). Moreover, they are not specifically tailored to the specific problem that we address here. We will, however, adopt a similar message propagation method where nodes coordinate and exchange meta-data regarding the utility of the actual data with its neighbourhood nodes before sending the actual data itself. This is done in order to efficiently make use of the limited energy resources. In contrast, APO uses centralised mediator agents to compute the solution for portions of the original problem. It lacks distributed control and, therefore, does not satisfy Requirement 2. Finally, Max-Sum exchanges messages between conflicting low-power sensor nodes using a cyclic bipartite factor graph representation. It scales very well in tree-structured graphs and generates approximate solutions close to the global ones returned by ADOPT and DPOP. However, it is not a complete algorithm, and may fail to converge in cyclic graphs as there is no built-in termination algorithm to stop the message exchange process.

In a somewhat related setting, Pizzocaro et al. (2008) try to optimize a sensor mission assignment problem by modelling it as a NP-complete *multiple knapsack problem*[9] (MKP). They introduced the *sensor utility maximization* model in which each sensor-mission pair is associated with a utility offer. Each mission might require and compete for the same sensors, however, a sensor can only be assigned to serve only one mission at any point of time. A profit function is formulated and this represents the value that a sensor can bring to a mission. The goal is therefore to maximise the total profit, while ensuring that the total utility cumulated by each mission does not exceed its uncertain demand (otherwise, some utility offer will be wasted). A novel greedy algorithm was developed and showed to perform better compared to a number of benchmarks by offering

[9]MKP is an optimization problem by choosing a set of items, each with different value and weight, that can fit into a number of bags, each with a different capacity, such that the total value is maximised and the total weight on each bag does not exceed its given limit.

the best trade-off between the quality of the returned solution and the computational cost. However, the algorithm considered here follows a centralised approach, where all the intensive computation takes place at a central node which has all the information about the system's state and, consequently, it does not answer Requirements 2 and 5. The dynamism (Requirement 7) of the sensor-mission pair, where a mission may change or adapt its priority value over time if its demand is not met after a certain period of time, is also missing in this work.

Now, each of the message-exchanged type of coordination mechanism we have discussed so far involves a relatively small additional cost in communication. This is inevitable as each agent possesses only a limited knowledge of the network's global problem and so, has to communicate and exchange coordination information with others. Thus, such a mechanism might not be ideally suited for WSNs with small data sets (including FLOODNET) as the resources required to perform the coordination activity may overwhelm those for transmitting the actual data which is typically of simple scalar value that requires a small amount of energy to process (e.g. to sample, transmit, and receive) due to its small size. To deal with this point, we motivate our new adaptive algorithms, which are developed in chapter 4, by considering wireless visual sensor networks which capture data of a much larger size (more details of which can be found in section 4.2).

2.3 Valuing Information in a Wireless Sensor Network

To operate effectively, a mechanism needs a means of valuing the various resources that are exchanged and allocated. In our case, the nodes exchange information about their local states and observations, and so we require a means for the nodes to place a value on any piece of information or observation. These properties are encapsulated in an *information valuation function* (in our case, it is a metric to determine how good a particular piece of sensor observation is compared to another), which also defines the desired outcome for a node's sampling action. The observations deal with uncertainty and imprecision that might be caused by noise.

Within the data fusion and tracking literature, where spatially correlated sensor readings typically represent the estimated position of a target, there are a number of standard techniques for doing this. Most common, is the use of Fisher information, whereby the estimated position of the target is represented as a multi-dimensional probability distribution, and Fisher information is used to quantify the uncertainty represented by this distribution (Bar-Shalom et al., 2001; Chu et al., 2002; Frieden, 2004; Zhao and Guibas, 2004).

Specifically, Chu et al. (2002) describe *Information-Driven Sensor Querying* (IDSQ) for an array of sensing nodes that are used to estimate the position of a target being tracked. The algorithm is decentralised, and each cluster-head node selectively chooses to fuse its own information with that of other available nodes, in order to update its current belief about the target's position. It is the job of IDSQ to direct each cluster-head node to fuse the most valuable data (i.e. data that more accurately represents the target's position), and to do so, a Fisher information measure is used. In our work, we follow a similar procedure in order to determine the value of previously collected and temporally correlated sensor observations. The value can then be used to decide on the nodes' next sampling plans (more details of which will be given in section 3.3).

Krause et al. (2006) use a Gaussian process to model the spatial correlations between nodes, and then use these correlations to select the subset of node placements that is most informative. The mutual information metric that they use is also very similar to the Fisher information metric that we use here in our work. However, instead of using the metric to express the information content of spatially correlated sensor samples, we use it to model the temporal aspect of the correlated sensor samples collected from our FLOODNET WSN.

Several other techniques for valuing information include *Shannon entropy* and *Kullback-Leibler divergence*, both are mainly used in signal compression (or coding), target tracking, and information fusion techniques in WSNs (Hwang et al., 2004; Cover and Thomas, 2006). Specifically, Shannon entropy characterises the average amount of information which is gained from a certain set of events. The entropy is maximal when all the events outcomes are equally likely and, therefore, we are uncertain which event is going to happen. When one of the events has much higher chance to happen than the others, then the uncertainty (or entropy) decreases. Information value can thus be quantified as the difference between two probabilities of the random event.

In more detail, Wang et al. (2004) attempt to reduce the entropy of a target location distribution by repeatedly selecting an unused sensor with maximal expected information gain. The mutual information between the target location χ and the predicted sensor observation z_i ($i \in I$ is the sensor index from the set of sensors I), is given by:

$$\begin{aligned} I(\chi; z_i) &= H(\chi) - H(\chi|z_i) \\ &= \int p(\chi, z_i) \log \frac{p(\chi, z_i)}{p(\chi)p(z_i)} \, \partial\chi\partial z_i \end{aligned} \tag{2.1}$$

where

$$p(\chi, z_i) = p(z_i|\chi)p(\chi) \tag{2.2}$$

and

$$p(z_i) = \int p(\chi, z_i) \; \partial\chi \tag{2.3}$$

where $H(\chi|z_i)$ is the expected conditional entropy of the posterior target location distribution, $p(\chi)$ is the prior target location distribution, and $p(z_i|\chi)$ is the sensing models of candidate sensor i for selection. The observation of sensor i_{max} thus maximizes the mutual information:

$$i_{max} = \arg\max_{i \in S} I(\chi; z_i) \tag{2.4}$$

On other hand, Gulrez and Kavakli (2007) use Kullback-Leibler divergence as a measure of difference in information gain between two probability distributions, $p(\chi)$ and $q(\chi)$, and the equation is given by:

$$D_{KL}(Q||P) = \int q(\chi) \log \frac{q(\chi)}{p(\chi)} \, \partial\chi \tag{2.5}$$

where in the object tracking literature, $q(\chi)$ typically represents the probability distribution after a new measurement (with x and y coordinates of the target location) takes place, whereas $p(\chi)$ is the prior. In general, the relevant sensor readings would increase the information gain or the reduction of entropy from the previous knowledge about the state of the target, whereas the irrelevant sensor information would lead us towards an increase in entropy or would not affect the previous knowledge at all. As part of our future work, we will investigate these two alternative metrics when we consider the spatio-temporal correlations between FLOODNET sensor readings (see section 5.2 for more details).

2.4 Adaptive Sampling Algorithms

With reference to WSNs, sampling is a process of sensing an event or recording a sample generated from the monitored environment. Therefore, an adaptive sampling algorithm (which is numbered 1 in figure 2.5) is here defined as a protocol that is responsible for adaptively setting the sampling rate (i.e. *how often* a node is required to sample during a particular time interval) and schedule (i.e. *when* a node is required to sample) of each of the individual nodes in a network.

In most environmental WSNs, high spatial densities of sensor nodes are desirable for achieving high resolution and accurate estimates of the environmental conditions. These high densities, however, place heavy demands on the bandwidth and energy consumption for sampling and communication. Thus, an adaptive sampling approach that can significantly reduce energy consumption compared to the non-adaptive one, is needed so that nodes do not unnecessarily continue to sample at their maximum rates and thus deplete their energy resources rapidly. In our case, the key intuition is that the adaptive sampling algorithm needs to be able to detect correlations in the environment; meaning that many nodes may not need to sample at a given moment in order to achieve a desired level of accuracy (e.g. when the environmental parameter being observed is highly correlated and, thus, changing slowly, there is no need to sample frequently). The amount of communication required will also be reduced since fewer measurements are collected. A number of adaptive sampling algorithms within the literature use temporal or spatial correlations (or both) in order to make effective sampling decisions, and we review them here.

With respect to spatial correlations between nodes, Willett et al. (2004) have studied the backcasting adaptive sampling method in which multiple nodes that are spatially correlated form small subsets of nodes that then communicate their information to a fusion coordinator. Based upon this information, the coordinator then selectively activates additional nodes in order to achieve a target error level. Similarly, Makarenko and Durrant-Whyte (2004) describe a negotiation-based *Bayesian Decentralised Data Fusion* (BDDF) technique for an array of wireless nodes in a network. Their work accounts for the uncertainty inherent in such tracking applications, and a Bayesian non-linear filtering method is used to aggregate sensed data. The local filter of a node fuses the observations, and these fused observations are used to decide the node's next sampling plan. However, based on Requirements 2 and 6 that we placed in section 1.3, we need a decentralised solution with minimal additional communication between sensor nodes. Moreover, due to the use of centralised fusion, the scalability issue (Requirement 5) is likely to be problematic since the central coordinator is required to compute a large number of fusion decisions. We also do not attempt to fuse information from separate nodes as this technique is typically applied for applications that explicitly consider the spatial correlations of sensor nodes (not the temporal correlations as we outlined in Requirement 1). Although we acknowledge that these spatial correlations almost certainly do exist within FLOODNET domain, we choose to leave it as part of our future work as they are less important compared to those found in sensor networks for localization purposes. In these circumstances, nodes must be distributed and they must exchange readings to improve their local beliefs about the object's position being tracked as these spatially proximal sensor observations are highly correlated (Vuran et al., 2004). However, a large number of environmental WSNs in which nodes are required to periodically perform sampling on the event being observed mostly exploit temporal correlations between each consecutive observation of a sensor node.

In a somewhat different setting, but still concerned with the decentralised approaches to decision making within sensor networks, Mainland et al. (2005) present a market-based approach for determining efficient node resource allocations in WSNs. Rather than manually tuning node resource usage or providing specific algorithms as we do here, *Self Organising Resource Allocation* (SORA) defines a virtual market in which nodes sell goods (e.g. data sampling, data relaying, data listening, or data aggregation) in response to global price information that is established by the end-user. However, this approach again involves an external coordinator to determine and set the prices in order to induce the desired global behaviour for the network and, hence, does not meet Requirement 2 since the system has a single point of failure. Moreover, it is not clear how this price determination should actually be done in practice.

Most similar to our work, and the one that exploits temporal correlations between sensed data, is the *Utility Based Sensing and Communication* (USAC) algorithm proposed by Padhy et al. (2006). This is a decentralised control regime for adaptive sampling, designed for the GLACSWEB WSN (which is detailed in section 2.1.1). The adaptive sampling aspect of the algorithm models temporal variations in the environmental parameter being sensed as a piece-wise linear function, and then uses a pre-specified *confidence interval* parameter in order to make real-time decisions regarding the sampling rate of the sensor nodes. Linear regression is used to predict the value of future measurements, and if the actual sensor reading exceeds the confidence interval parameter, the node starts sampling at an increased rate. However, this parameter (and several others such as the window length and actual sampling rate) must be carefully selected, and a poorly chosen value can result in very poor performance in our setting (as we show later in section 3.5). For this reason, we require a more principled approach of valuing information, and this must be based on a non-parametric model (Requirement 1). Furthermore, since the algorithm does not explicitly perform any forward planning, the node can rapidly deplete its battery if the increased sampling rate is constantly re-triggered by data that is far from linear.

More recently, Dang et al. (2007) have proposed an adaptive sampling algorithm to find the optimal cruise path of a mobile sensor node in order to collect data that maximally reduces the uncertainty of a data assimilation mode, that is based on the Sigma Point Kalman Filter. Here, the environment being monitored is modelled as a set of grid points that are available for the node to sample at. The next sampling point is chosen to be the point that results in the lowest trace of the predicted covariance matrix indicating how uncertain the estimated environment state is. However, with this algorithm, there is an issue of scalability as the computation time of searching the next sampling point increases exponentially when extra nodes are introduced. Therefore, this algorithm is not suitable in our case since it does not satisfy Requirement 5.

Finally, Osborne et al. (2008) use a multi-output Gaussian process to explicitly model both temporal and spatial correlations between a small number of nodes. The Gaussian process is used for adaptive sampling whereby it can determine both the time, and the node from which the next sample should be taken, to ensure that the uncertainty regarding the environmental parameter being measured at each node location stays below a pre-specified threshold. However, as with the USAC algorithm, there is no forward planning. Moreover, contradicting with Requirement 2, the algorithm is centralised, since it requires information from all of the nodes in order to model the spatial correlations between them, and it is relatively computationally expensive; the novelty in the paper above, being a computationally efficient formalism of the Gaussian process. The computational cost (Requirement 5) precludes it being deployed on the current generation of sensor nodes and, furthermore, since it requires nodes to exchange data with one another, it would also incur additional communication cost that could possibly outweigh any saving achieved through more effective sampling (and, therefore, also fails to meet

Requirement 6).

Having described the existing work in adaptive sampling algorithms, in the next section, we consider those that exist in routing and more specifically at some of the most commonly used ones that have been developed for WSNs.

2.5 Adaptive Routing Algorithms

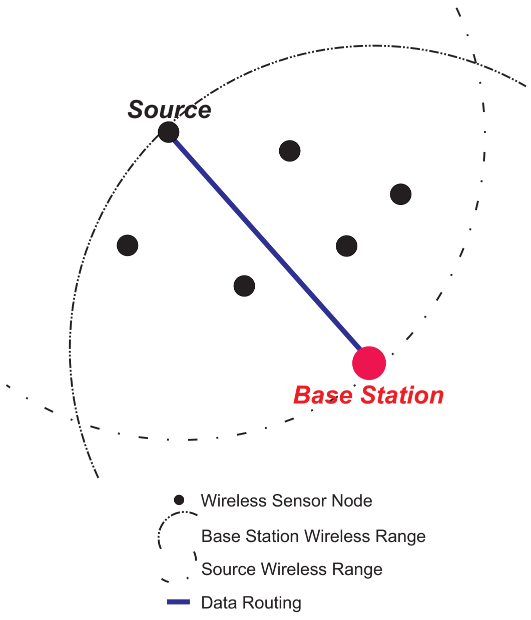
Routing is a process of transporting messages across a network by choosing a particular route depending on the protocol used. With particular reference to WSNs, routing is the process of delivering a message from a source node to a base station inside the same network. An adaptive routing algorithm (which is numbered 2 in figure 2.5) is therefore defined as an adaptive protocol that is in charge of determining the *transmitting* (i.e. *how many* samples of its own, a node is required to transmit during a transmission period), *forwarding* (i.e. *how many* samples of its neighbour, a node is required to to receive and forward during a transmission period), and *routing* (i.e. *which* route a node should transmit its own and forwarded samples through) decisions of each of the individual nodes in a network.

The most simplistic way to undertake this is via a single-hop communication (as shown in figure 2.6(a)), whereby a source node uses its radio transceiver to send a message directly to its destination base station since the transmission range of the source is within that of the base station. In contrast, multi-hop routing uses intermediate nodes between the source and the destination to forward messages (as can be seen in figure 2.6(b)). Based on the physics of radio transmission, it is known that a number of short range transmissions are typically more efficient than a single long range one because of the characteristics of wireless channel (Mhatre and Rosenberg, 2004). More specifically, this increase in inefficiency is due to the fact that as the separation of the nodes increases, the transmission power required increases geometrically[10]. Given this, this thesis adopts a multi-hop approach to routing.

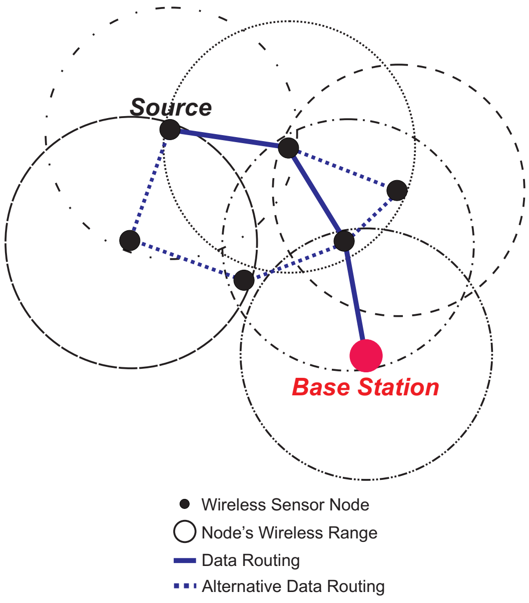
Moreover, there are a number of additional key issues in WSN routing, including:

- **Scalability:** The number of sensor nodes deployed in a sensing area maybe in the order of hundreds or more. Any routing schemes must be able to work (in real-time) with this large number of nodes.

[10]In practice, however, the energy efficiency in multi-hop routing depends on the network configuration including the topology and the relative distances of sensor nodes to the base station (Khan and Peng, 2005). In a number of occasions, a small number of large hops is more efficient than a large number of the smallest possible hops (Shelby et al., 2005). We remark that while in settings where the power required for multi-hop routing may be higher that that for single-hop, this cost is spread across a number of nodes instead of a single node and, thus, helping to evenly distribute the network degradation over the nodes.



(a) Single-hop.



(b) Multi-hop.

FIGURE 2.6: Basic routing methods.

- **Integration with Wake/Sleep Schedules:** To save power, many WSNs place their nodes into sleep states. Therefore, an awake node should not choose an asleep node as the next-hop node, unless somehow, it can first awaken that node. In our setting, however, the wake up and sleep schedules of each node in the network are pre-determined and synchronized.
- **Network Dynamism:** Some sensor nodes may fail or be added. These changes should not overly affect the overall routing performance of the sensor network.
- **Reliability:** Since messages need to travel multiple hops, it is important to have a high reliability on each link, otherwise the probability of a message arriving successfully at the destination will be low which, in turn, means that some energy resources will be wasted.
- **Mobility:** The routing process is made more complicated if either the source or destination or both nodes are not static in term of their positions. In our case, however, we assume that we have static nodes of which each has a 2-D spherical radio transmission range model (as can be seen in figure 2.6).
- **Security:** Many WSN routing algorithms have ignored their security factors. If the network is not owned by a single source (i.e. many adversaries exist), each node can initiate a wide variety of attacks on the routing algorithm for its own benefit. This selfish (or competitive) issue will be considered as part of our future work (see section 5.2 for more details).

A comprehensive overview of routing techniques in WSNs is given in (Al-Karaki and

Kamal, 2004; Akkaya and Younis, 2005). However, we here categorize the main example techniques into three broad approaches; namely (i) energy-based in which the protocol is designed to minimise energy consumption, (ii) information-based in which the protocol is designed to maximise the information value of data collected, and (iii) market-based in which the protocol is designed to maximise overall utility gained. We know consider each of these in turn.

2.5.1 Energy-Based Algorithms

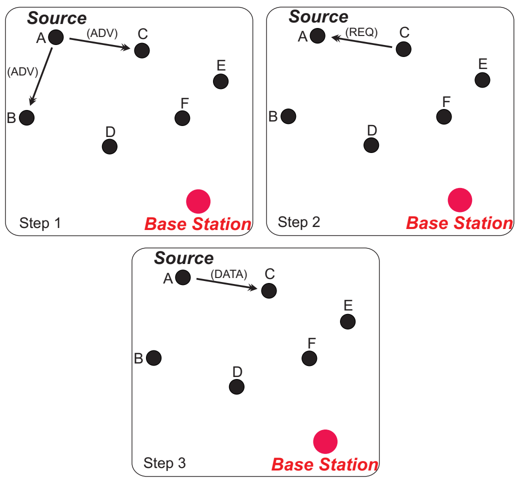
Energy-based routing protocols in WSNs were originally developed with the aim of finding the least power consumption route from the source to the destination base station (Youssef et al., 2002; Liu et al., 2005b; Gao and Zhang, 2006). Such approaches, however, typically result in a rapid reduction in the overall network lifetime as what often happens is that one particular shortest route is heavily used for data routing. Hence more recent research has focused on spreading the energy cost evenly over the entire network. Specific techniques include choosing the nodes with the most battery power remaining, using a combination of weighted routing factors, and electing a sub-base station (or cluster-head) to be in charge of data re-routing for a subset of nodes. We will discuss each of them in turn.

Among the energy-based protocols, *Flooding* (Akyildiz et al., 2002) is the simplest. It involves each of the nodes broadcasting every received message to all it neighbours, unless the maximum number of hops for the message is reached or the destination of the message is the node itself. The major advantages of flooding, being a reactive protocol (i.e. one that responds appropriately to prevailing circumstances in dynamic and unpredictable environments), are simplicity and minimal overheads since it does not require costly topology maintenance or complex route discovery protocols. However, it can result in *data implosion* (where a node receives multiple copies of the same message from its neighbours), *message overlap* (where an event is detected by multiple nodes, which all report the same stimuli at the same time, resulting in unnecessary wastage in communication power), and *resource wasteness* (as it does not take resources such as the remaining energy into account). To improve upon this basic approach, a number of extensions have been proposed. Specifically, *Gossiping* (Medidi et al., 2005) is an extension in which nodes do not broadcast, but instead forward the incoming messages to a randomly selected neighbour. This approach thus reduces the first two limitations found in Flooding. *Flossiping* (Zhang and Cheng, 2004) is an enhancement to Flooding and Gossiping approaches by using a single branch gossiping with low-probability random selective relaying in order to achieve a better overall performance. Whilst *Directional Flooding* (DF) (Ko et al., 2004) aims to reduce the number of hops, by targeting the flood towards the destination with the assumption that all nodes know both their own geographic location and the location of the destination for the current message. However, all the four protocols do not meet Requirement 7 (as outlined in section 1.3) since we require an adaptive routing algorithm that can react accordingly to the network's dynamism.

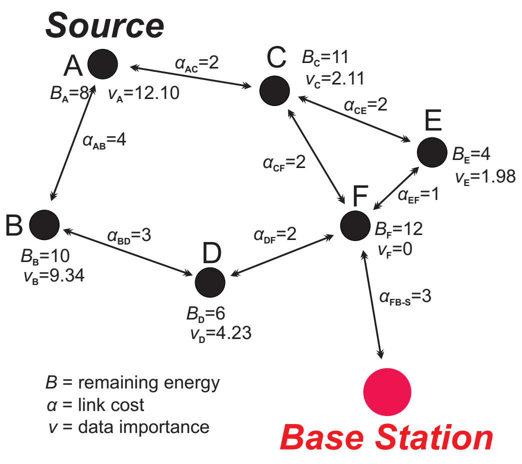
We observed that though Flooding, Gossiping, Flossiping, and DF techniques are still widely used in WSNs, a number of researchers are developing new methods because of their inherent power hungry nature. Kulik et al. (2002) develop *Sensor Protocols for Information via Negotiation* (SPIN) that adopts a publish-subscribe approach in which a source node broadcasts an advertisement (ADV) containing meta-data (which is the data that describes the sensor data) before sending the actual data itself which requires comparatively a bigger communication cost due to its larger size. As can be seen in figure 2.7(a), interested listeners then respond with a request (REQ). The transmitting (or source) node subsequently responds by sending the actual sensor data (DATA). This protocol introduces a slight extra message overheads due to its publish-subscribe approach and may only be relevant for large data sets. However, SPIN's data advertisement mechanism cannot guarantee the delivery of data (and, hence, does not satisfy Requirement 3). Consider a simple scenario expected in figure 2.7(a) where the base station is interested in the data but all the intermediate nodes between the source and the base station (nodes B, C, D, E, and F) are not, such data will not be delivered to the destination base station at all.

In FLOODNET, an *Adaptive Routing Algorithm* (ARA) (Zhou and De Roure, 2007) has been developed. ARA is a protocol designed with the aim of optimising the power consumption to extend the network's lifetime, while also satisfying the FLOODNET domain application specific requirement of the need for timely data. It consists of an adaptive sampling and an adaptive routing algorithms. The adaptive sampling aspect of the protocol uses a similar approach to the Backcasting adaptive sampling algorithm and, hence, inherits the same limitations of having a single point of failure (Requirement 2). This is due to the existence of a central coordinator that dictates the data importance and sampling plan of each node. The routing protocol, on other hand, is determined by several other factors including the remaining battery power of the communicating nodes, the importance of the data being transmitted, and the link cost (proportional to the distance) between the two nodes (as shown in figure 2.7(b)). In their setting, however, the adaptive sampling and adaptive routing algorithms are loosely coupled (and, hence fail to meet Requirement 4). Consider a scenario when intermediate nodes are required to sample at their maximum rates by the central coordinator, they will therefore deplete their energy resources rapidly. Other nodes, particularly those at the edge of the network, will not be able to successfully transmit their messages to the base station at a later stage since the intermediate nodes will have insufficient power to relay messages.

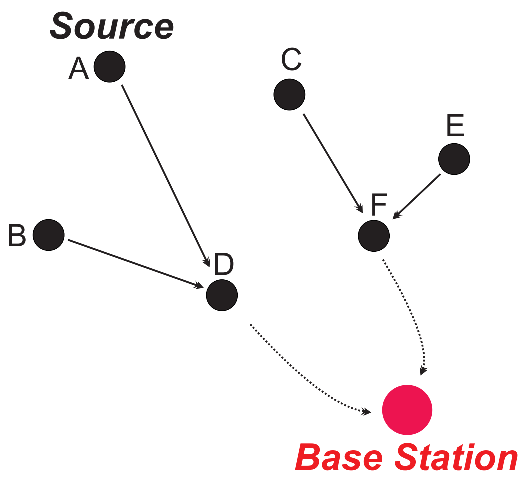
The final set of protocols in this class are *Low Energy Adaptive Clustering Hierarchy* (LEACH) (Veyseh et al., 2005) and *Power Efficient Gathering in Sensor Information*



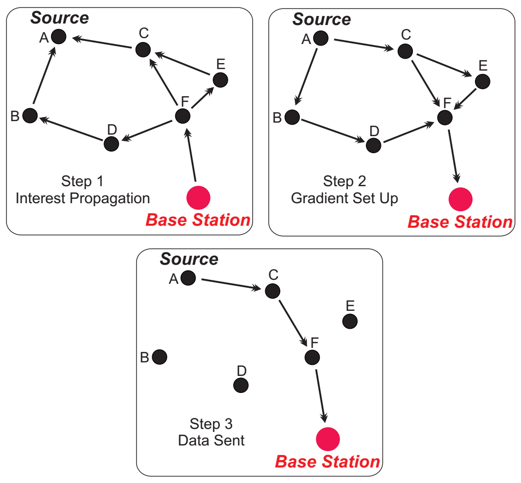
(a) SPIN protocol.



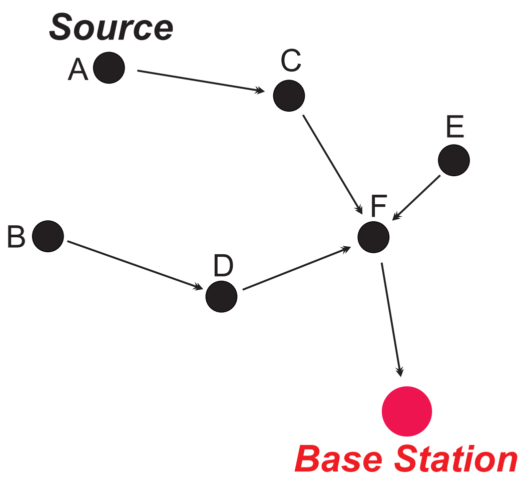
(b) ARA protocol.



(c) Clustering-based protocol. Nodes D and F are the cluster-heads which communicate directly to the base station.



(d) Data centric protocol.



(e) Data aggregation protocol.

FIGURE 2.7: Examples of routing protocols.

Systems (PEGASIS) (Lindsey and Raghavendra, 2002). They are clustering-based protocols that minimise energy consumption in sensor networks through the rotation of cluster-heads such that the high energy consumption in communicating with the base station is spread across all nodes. In general, the methods in this class make good attempts to try to balance the energy consumption by electing cluster-heads of which each is responsible for relaying the data from a subset of nodes back to the base station. However, these cluster-heads all need to be placed inside the base station's radio range as they communicate directly to it (using a single-hop routing approach as can be seen in figure 2.7(c)). Therefore, these single cluster-leaders can become a bottleneck (which clearly does not satisfy Requirement 2). Additionally, overhead and complexity associated with applying reorganization phase to form clusters could increase if there are many cuts in the network (for instance in a highly dynamic network as required in Requirements 6 and 7). Both protocols also assume that each node has location information about all other nodes in the network so that it knows where to route data. This is seemingly unrealistic in the current stage of WSN unless nodes are individually equipped with a *Global Positioning System* (GPS) module.

2.5.2 Information-Based Algorithms

An alternative class of protocols are those that seek to gain maximum information value across the network. An example of this is *Dynamic Source Routing* (DSR) (Qin and Lee, 2004) protocol that FLOODNET once adopted for multi-hop routing. DSR is a reactive protocol that adapts quickly to routing changes, specifically, when node failures or additions are frequent. It is a source-routing-based protocol where the path to the destination is known only to the source nodes. A source node determines routes by constantly flooding *route request* packets into the network during a *route discovery* phase, whenever it want to transmit its readings. Therefore, this protocol is undesirable since it is not scalable and involves large communication cost due to its flooding approach (failing to meet Requirements 5 and 6). DSR performs efficiently only for a network with less than a few tens of nodes. And as we require an algorithm that works efficiently on large-scale networks in general (as outlined in section 1.3), DSR is undoubtedly not suitable.

On other other hand, *Information Managed Energy Aware Algorithm for Sensor Networks with Rule Managed Reporting* (IDEALS) (Merrett et al., 2005) protocol aims to extend the network lifetime of WSNs. IDEALS is an application specific heuristic protocol as it includes every sensor node deciding its individual network involvement based on the information importance contained in each message. The protocol lacks of a principled approach to value the importance of sensor readings (Requirement 1) since the information importance is basically divided into discrete levels. The action space of each node considered here only includes routing, while based on Requirement 4, we also need to consider the inter-dependent sampling, transmitting, forwarding, and routing actions.

Other information-based protocols methods includes data-centric and data-aggregation protocols. *Directed Diffusion* (DD) (Intanagonwiwat et al., 2003) is a data-centric protocol (meaning that the base station is the one that sends out an interest stating query description by flooding the query to the entire network, as can be seen in figure 2.7(d)). Therefore, directed diffusion is not suitable to be applied to applications (such as environmental monitoring WSNs) that require continuous data delivery to the base station. To avoid flooding, *Rumor Routing* (RR) (Braginsky and Estri, 2002) routes the queries to the nodes that have observed a particular event to retrieve information about the occurring events and, thus, reduces the total communication cost. However, rumor routing performs well only when the number of events is small. For a large number of events, the algorithm becomes infeasible (and, therefore, fails to satisfy Requirement 5) due to increase in the cost of maintaining node-event tables in each node. Moreover, all data-centric protocols do not take into account the network's dynamism (Requirement 7) in routing data messages.

Data Combining Entities (DCE) (Schurgers and Srivastava, 2001), *Gradient Based Routing* (GBR) (Xia et al., 2005), and *Information Directed Routing* (IDR) (Liu et al., 2005a) family of methods permits individual nodes to process and aggregate their sensor data, before relaying it to the others. As shown in figure 2.7(e), node F aggregates the incoming data (from nodes C, D, and E) and its own before sending the result to the base station. However, as explained before, the data fusion technique, is typically applied for applications that explicitly consider the spatial correlations of sensor nodes (not the temporal correlations as we have in Requirement 1).

2.5.3 Market-Based Algorithms

The last class of protocols are those that use a *market-based control* (MBC) paradigm. The usage of MBC in WSN allows the use of tools from general equilibrium theory to analyse the behaviour and correctness of a decentralised system. The main market-based protocol includes *Self Organised Routing* (SOR) (Rogers et al., 2005) and SORA which has already been discussed in section 2.4.

In more detail, SOR is a mechanism-design-based protocol proposed for the GLACSWEB domain. It is a distributed protocol that aims to maximise the network's lifetime. Each node is designed to follow locally selfish strategies which, in turn, result in the self-organization of a routing network with desirable global properties. The protocol consists of a communication protocol, equipping nodes with the ability to find and select a node that is willing to act as a mediator for data relaying, and a payment scheme that guides the nodes to make local decisions that ensure good global performance whereby a node is rewarded for forwarding messages to the destination. Specifically, the communication protocol calculates the implications using any found mediators and selects the one that appears to provide the greater increase in the lifetime of the originating node. The payment scheme has the effect of reducing the area where mediators may relay data for other nodes. The action space of each node does not meet Requirement 4 since in a more realistic scenario, the protocol should take into account the node's actions of sampling, transmitting, and routing its own samples, as well as forwarding and routing neighbour's data samples.

This type of incentive-compatible mechanism will however be considered as one of our avenues of future work as it presents an incentive for nodes (e.g. to relay data for other nodes) in a non-cooperative (i.e. competitive) setting where each of the nodes is modelled as a rational and selfish entity trying towards achieving global system-wide goals (see section 5.2).

2.6 Inter-Related Adaptive Sampling and Routing Algorithms

In an inter-dependent sampling and routing setting, efficient coordination is typically computationally expensive since each node's energy consumption must be optimally allocated between *sampling* and *transmitting* its own data, *receiving* and *forwarding* the data of other nodes, and *routing* any data. In particular, this means that the choices of one node can potentially affect all other nodes in the network. Consider a case in which intermediate nodes (i.e. nodes that are relatively closer to the base station) naïvely sample and transmit their own samples at their maximum frequency. By doing so, they would deplete their energy rapidly and would not be able to relay (or forward) their neighbours' data messages (particularly those coming from nodes far away from the base station). To date, very few approaches have attempted such an integration approach. Noticeable exceptions, however, are the work of Padhy et al. (2006) and Mainland et al. (2005) who developed USAC and SORA respectively. Both protocols have already been discussed in section 2.4. We will therefore only highlight the integration aspect of each of them in this section.

USAC consists of a decentralised adaptive sampling and routing protocol. Within this mechanism, each node adjusts its sampling rate depending on a valuation function that assigns a value to newly sampled data (recall section 2.4). This protocol is intended for low power, computationally constrained devices, and as such, relies on a heuristic approach to estimate the opportunity energy cost used by each sensor for sampling, forwarding, and routing data. The protocol is not efficient and the integration of the node's actions (Requirement 4) is very limited since there is no guarantee that the transmitted data will actually be forwarded to the base station. For instance, there might be cases where nodes with data of a high value are unable to send their data to the base station because intermediate nodes have depleted their energy. The protocol could thus result in no data collection at the base station (and, hence, does not meet Requirement 3).

Another approach in this area is that of SORA. Here, an external coordinator (or agent) determines and sets the prices of nodes' actions in a virtual market to induce the desired global behaviour for a network. If the coordinator wants to capture more samples (due to, for instance, the rapid changes in the environment being monitored), it raises the price of data sampling and decreases those of data listening and transmitting. Later, in order to guarantee the successful delivery of collected data to the base station, the coordinator will increase the price of data listening and relaying so that nodes will help others to relay messages. This is because nodes are designed to selfishly maximise their utility function and, therefore, are incentivised to take this particular action. Given their limited energy resources, they will thus reduce their sampling frequency since a portion of energy has been highly drained for forwarding activities.

2.7 Summary

In this chapter, we have described some widely used WSN applications, with a detailed introduction into the FLOODNET domain, since this forms the exemplar application of this thesis. We then reviewed centralised and agent-based decentralised control regimes for WSNs, particularly, in the area of DCOP, and explicitly discussed their advantages and disadvantages. Following that, we compared and contrasted the most common adaptive sampling, adaptive routing, and inter-related adaptive sampling and routing algorithms in WSNs (see table 2.1 for a summary). We have shown that they all suffer from at least one of the following major limitations:

- Using a centralised paradigm (e.g. SBB, APO, MKP, SORA, ARA, and BDDF protocols) that has a partial or complete knowledge regarding the state of the network, but could catastrophically act as a bottleneck and reduce the network robustness (Requirement 2).
- Requiring significant amounts of communications among the nodes, raising the issues of scalability and reliability (failing to meet Requirements 5 and 6) when applied for large-scale networks with hundreds of nodes of which each is installed with limited computational, storage, memory, and communication resources. A list of algorithms that suffer from these drawbacks include SBB, ADOPT, DPOP, BDDF, DSR, and DD protocols.
- Adopting an approach that is not reactive and proactive to the dynamic and rapid changes of the complex and inter-connected WSN systems. MKP, LEACH,

PEGASIS, Flooding, Gossiping, Flossiping, DF, DD, and RR protocols therefore fail to meet Requirement 7.

- A few algorithms (e.g. DBA, SPIN, and USAC protocols) do not give a warranty that data samples will eventually arrive at the base station since intermediate nodes might deplete their energy much faster than those nodes on the edge of the network. This clearly does not satisfy our Requirement 3.
- All algorithms in the literature do not show a tight integration among the nodes' actions. We believe nodes in WSNs should have an option to either act simply for their own good (by sampling and transmitting their own samples at maximum rate) or to cooperate and help others (by receiving and forwarding their samples to the destination), as outlined in Requirement 4.

Against this background, in chapter 3, we design three novel decentralised control algorithms for information-based adaptive sampling which represent a trade-off in computational cost and optimality, using the combination of a Fisher information metric and GP regression as a measure of the information content of sensor samples. This work is similar to that of Krause et al. (2006), but we consider temporal, rather than spatial, correlations (see section 3.3.1 for more details).

In chapter 4, we continue to develop a novel optimal decentralised adaptive sampling, transmitting, and forwarding algorithm which assumes fixed routing and varies each node's sampling, transmitting, and forwarding rates in order to ensure all nodes use their limited resources on maximising the information content of the data collected at the base station. We then extend it to deal with flexible routing such that the extended algorithm will work with any arbitrary network topologies and we optimise both the route and the integration of actions of each node. To this end, we will use the technique of DCOP and meta-data message exchange before sending the actual data (in a similar way to those of DPOP and Max-Sum). However, in our case, we use localized synchronous communication to reduce the exponential increase in total messages being exchanged due to the conflicts of decisions between nodes.

Table 2.1: Protocols summary (sorted alphabetically).

Protocol	**Method**	**Additional Remarks**
ADOPT	Asynchronous message passing.	Uses a distributed asynchronous opportunistic depth-first search strategy and an efficient termination detection.
APO	Cooperative mediation process.	Creates centralised mediator agents to solve subproblems of the distributed constraint problem. The agents then increase the size of the subproblems over time.
ARA	Adaptive sampling and energy-aware adaptive routing algorithm.	Centralised adaptive sampling and routing protocol driven by an external coordinator.
Backcasting	Adaptive sampling algorithm.	Adopts a centralised approach for adapting the sensor nodes' sampling behaviour to achieve a target error level.
BDDF	Data aggregation algorithm.	Uses the Bayesian non-linear filtering method to aggregate sensed data in order to decide the next sampling plan of each node.
DBA	Constraint satisfaction algorithm.	Incomplete distributed algorithm that uses the combination of the hill-climbing technique in parallel with the breakout method as a strategy to escape from local minima.
DCE	Data aggregation algorithm.	Aggregates sensed events before transmitting them to significantly reduce energy consumption for communication.
DD	Data-centric protocol with interest flooding method.	Base station sends out interest (stating its query description) which is flooded within the network. Listening nodes that satisfy the interest's requirements react by sending their data back to the base station.
DF	Rebroadcasts every received messages towards the destination.	Each node must be aware of its own and others' geographic locations, as well as the location of the destination for the current message.
DPOP	Asynchronous communication.	Agents (or nodes) construct the pseudotree structure of the network and exchange asynchronous messages with neighbourhood nodes to converge into an optimal and non-conflicting solution.
DSR	Source-based path discovery.	Was used to control FLOODNET system but lacks of scalability as nodes constantly flood route request packets into the network whenever they want to transmit readings.
Flooding	Rebroadcasts every received message.	Simple and reactive protocol but is power hungry as it can result in massive message redundancy.
Flossiping	Single branch gossiping with low-probability random selective relaying.	Enhancement to Flooding and Gossiping approaches.
GBR	Data aggregation algorithm.	Protocol that employs data aggregation to save transmission energy.

Protocol	**Method**	**Additional Remarks**
Gossiping	Forwards every received message to a randomly selected neighbourhood node.	Reduces redundancy in Flooding, however, is still power hungry in nature.
GP	Most informative sensor placements algorithm.	Uses the GP to model spatial correlations between nodes.
IDEALS	Information importance rule managed reporting algorithm.	Each node decides its individual network involvement based on the information importance contained in its message.
IDR	Data aggregation algorithm.	Centralised protocol that is formulated as a joint optimisation of data transport and information aggregation with the objective of minimising communication cost while maximising information gain.
IDSQ	Data fusion algorithm.	Decentralised protocol to update each node's current belief about the target's position being tracked, by fusing the most valuable data from other nodes.
LEACH	Clustering- (or hierarchy-) based routing algorithm.	Cluster head aggregates the data of its cluster member nodes.
Max-Sum	Variant of sum-product algorithm.	Uses a simple message passing scheme to efficiently find approximate solutions to the distributed constraint optimization problem.
MKP	Centralised greedy sensor-mission assignment algorithm.	Maximises the total profit that sensors can bring to missions, imposed by a set of constraints (modelled as a multiple knapsack problem).
Multi-Output GP	Adaptive sampling algorithm.	Uses the GP to model both spatial and temporal correlations between a small number of nodes.
PEGASIS	Clustering- (or hierarchy-) based routing algorithm.	Assumes that each node has location information about all other nodes (i.e. has complete knowledge about the network).
RR	Data-centric algorithm.	Uses the same method as that of DD. However, instead of flooding the whole network with query or event messages, it creates paths leading to each event when the event happens and, thus, reduces the total communication cost.
SBB	Synchronous message passing.	Complete and centralised constraint optimization algorithm in multi-agent system coordination.
SOR	Mechanism design approach.	Decentralised control approach that utilizes a payment scheme to ensure global goals achievement.
SORA	Market-based algorithm for node's resource allocation.	Protocol with a payment scheme for nodes to act upon in order to determine their ideal behaviours (by taking actions that maximise their own utilities).
SPIN	Publish-subscribe approach.	Involves extra message overheads and, thus, only efficient for large data sets.
USAC	Inter-dependent adaptive sampling and routing algorithm.	Its information valuation is calculated based on the confidence interval that is resulted from a simple linear regression over the actual sensor readings. The routing protocol relies on a heuristic approach to estimate the opportunity energy cost to transmit.

Chapter 3

Decentralised Control of Adaptive Sampling

This chapter outlines the work undertaken towards addressing the problem of decentralised control of adaptive sampling in WSNs in general and FLOODNET in particular. Specifically, here we describe a principled information measure based upon Fisher information and *Gaussian process* (GP) regression (Contribution 1), and we present three decentralised algorithms that allow individual nodes to maximise this information measure given their individual energy constraints (Contribution 2). We then show how the algorithms permit nodes to collect more valuable information compared to that of a number of benchmarks (Contribution 3).

To this end, in section 3.1, we highlight the intuition of information-based adaptive sampling by considering an exemplar scenario. In section 3.2, we then detail the sampling problem we face in a general manner. In section 3.3, we show how we use a GP package to calculate our information metric that allows the information content of a node's observations to be expressed. In section 3.4, we formulate the three decentralised control algorithms for solving the problem. We start with an exponential algorithm that maximises the Fisher information metric by performing GP regression on the individual nodes, and progress to a more computationally tractable algorithm that performs a greedy optimisation. We further reduce the computational cost by using an efficient algorithm that uses a heuristic approach, rather than the GP regression, in order to determine the allocation of sampling actions each day. Their performances are then empirically evaluated against a number of benchmarks in section 3.5. Finally, conclusions are discussed in section 3.6.

3.1 Information-Based Adaptive Sampling

Due to their constrained energy resources, a key requirement within WSN applications is an effective energy management policy. Now, this is often addressed through adapting the sampling policies of the nodes. Sampling policies generally describe a node's *sampling rate* (i.e. *how often* a node is required to sample during a particular time interval) and *schedule* (i.e. *when* a node is required to sample). In most of these WSNs, the nodes are typically not able to sample at their maximum rate because doing so would deplete their energy in a number of months or even weeks. Therefore, there exists a trade-off associated with wanting to gain as much information as possible by sampling at the highest frequency, with the energy constraints available to accomplish these activities. To address this, our mechanism (more details of which are given in section 3.4) seeks to dictate that each node should conserve battery energy by taking more samples during the most dynamic (unpredictable or rapidly changing) events, while sacrificing some energy by sampling less during the static ones.

To illustrate this point, consider the exemplar scenario expected in figure 3.1. If both nodes had not sampled as frequently as they did during the static event (i.e. if we remove some samples that are represented by small filled dots, randomly or at fixed intervals), we would still be very certain about the water level pattern at that period of time. But if the nodes had not taken samples during the dynamic event (those that are represented with bigger dots), the value of uncertainty about the water level would increase dramatically (as we would have expected the water level in figure 3.1(a) to remain constant within regions 2.2 and 2.4, to continue to rise within regions 1.1 and 1.2, and to continue falling within regions 2.1 and 2.3, and the water level in figure 3.1(b) to remain constant within regions 1.2 and 2.1, to continue to rise within regions 1.1 and 1.3, and to continue falling within region 2.2). Thus, sampled data within dynamic events is more valuable than that during static ones.

Nodes are therefore required to make optimal use of their energy resources to sample data of top importance. In doing so, they should place higher priority on those samples that have a higher information content (i.e. those within regions 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, 2.4, and 3.1). Predictable samples that are less important (i.e. those within regions 4.1, 4.2, 4.3, and 4.4, and all other smaller dots) are sensed only if there is sufficient remaining energy to do so.

To this end, some recent work has explored decentralised algorithms that enable the nodes to autonomously adapt and adjust their own sampling policies (recall section 2.4). Such solutions are attractive in our context since they remove the bottleneck of a central decision maker (and the need to inform this decision maker of the energy state of each node), and they fully exploit the ever increasing computational capacity of the nodes themselves. Furthermore, they are also more robust than centralised alternatives since there is no single point of failure, and even in the case that communication with

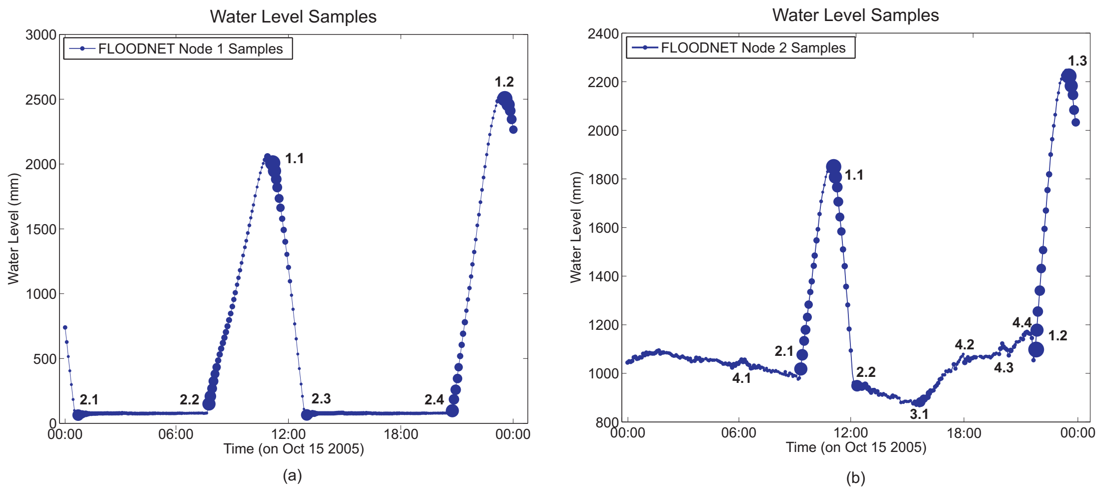


FIGURE 3.1: Information-based adaptive sampling scenario for (a) FLOODNET node 1 and (b) node 2. Filled dots represent the collected samples. The bigger the dot the higher the level of importance of this particular sample. In these cases, the biggest ones are located within regions 1.1, 1.2, and 1.3. Smaller ones are located in other regions identified by numbers sorted increasingly according to their levels of importance.

the base station fails (perhaps due to the failure of a node on a multi-hop route to the base station), the nodes are able to continue to autonomously operate in the absence of any external direction until communication is restored.

To date, such decentralised algorithms have typically been applied to WSNs deployed for environmental monitoring, and they have specifically considered networks composed of battery powered nodes that exhibit finite lifetimes. Since a node sampling at its maximum rate would deplete its battery in a short period of time, effective sampling policies in this context seek to balance the lifetime of the sensor network as a whole against the value of the information that it collects. To do so, they typically invoke domain specific heuristics that depend upon one or more user specified parameters. For example, the USAC algorithm of Padhy et al. (see section 2.4), which is representative of the state-of-the-art in this area, models temporal variations in the environmental parameter being sensed as a piece-wise linear function, and uses a pre-specified *confidence interval* parameter in order to make real-time decisions regarding the sampling rate of the nodes.

However, in many applications, nodes are also capable of harvesting energy from their local environment through different sources (e.g. solar power (Li et al., 2008; Alippi and Galperti, 2008), wind (Weimer et al., 2006; Park and Chou, 2006), or vibration energy (Zhang and He, 2008; Torah et al., 2008)). In such cases, additional operating modes become possible, and a common alternative to that described above is to require that the nodes maintain *energy neutral* operation; balancing energy harvesting against energy consumption, in order that they exhibit an indefinite lifetime (Kansal et al., 2007). In this context, an effective sampling policy must maximise the information that a node collects over a particular time interval subject to energy constraints, and this typically involves planning exactly when, within the specified time interval, to take a constrained number of samples. To actually achieve this within a general setting without resorting to domain specific heuristic requires that (i) we can predict the information content of a node's future samples, given a particular sampling schedule, and (ii) we can then optimise this sampling schedule, subject to energy constraints, in order to maximise the information that will be collected by a node over a particular time interval.

Thus, against this background, in this chapter we address these two complementary challenges. In particular, we develop a principled generic information metric for sensor networks based upon Fisher information and GP regression. We then present three decentralised algorithms for information-based adaptive sampling which represent a tradeoff in computational cost and optimality. These algorithms allow individual nodes to maximise this information measure by adapting and adjusting their sampling policies.

3.2 Problem Description

Here, we formalise a description of the generic sampling problem that we face (of which FLOODNET is but one specific instance). To this end, let n be the number of sensor nodes within a WSN system and the set of all nodes be $I = \{1, \ldots, n\}$. The sensor network is tasked with monitoring some environmental parameter over multiple days. We divide the day into a fixed number of time slots and denote these time slots by the set $H = \{1, \ldots, w\}$.

Each node $i \in I$ can sample at s_i different rates over a period of time. Its set of possible sampling rates is denoted by $C_i = \{c_i^1, \ldots, c_i^{s_i}\}$ where $C_i \subseteq \mathbb{Z}^+$ and $c_i^j < c_i^{j+1}$. Specifically, each element of this set, c_i^j, is a positive integer that describes the number of times that the node samples during a time slot.

Each of the algorithms (that we will devise within this chapter) determines the actual sampling rate that each node should use within any specific time slot. Thus, each node $i \in I$ has an allocated set of sampling actions (i.e. sampling schedules) for each day denoted by $Alloc_i = \{a_i^1, \ldots, a_i^w\}$, where $a_i^k \in C_i$, $\forall k \in H$. Any element, $a_i^k \in Alloc_i$ therefore represents the number of times that the node should sample within any specific time slot within the day. Hence, at the end of a day, node i will have collected a set of g_i observations, Y_i, at a corresponding set of sampling points, X_i, such that $|X_i| = |Y_i| = g_i = \sum_{k=1}^{w} a_i^k$.

In general, the nodes within the network will deplete their energy resources at different rates since they will have different sampling schedules. Assuming that the remaining battery power available for sampling of node i at the beginning of a day is B_i, and it requires a certain amount of energy, e_i^{s}, to sample an event, we must ensure that any set of sampling actions satisfies:

$$\sum_{k=1}^{w} a_i^k \cdot e_i^{\mathrm{s}} \leq B_i \tag{3.1}$$

such that the sum of all the energy required to do the sampling actions on that day must not exceed the remaining battery power. Note that our choice of imposing the energy constraint over a 24 hour period is a natural one since it represents a daily cycle in which the node recharges its battery during daylight and gradually depletes it during the night. Furthermore, note that we do not include the constant transmitting and receiving variables into the equation since each node transmits its recorded readings in every two hour period to the base station using a non-adaptive multi-hop routing method (with a routing table created at system start up time)[1].

The algorithms that we shall consider are thus constrained to take a maximum number of samples within a 24 hour period. The details of how this maximum number is calculated, however, are not restricted in any way. For example, figure 3.2 illustrates this by comparing (a,c) the case where $H = \{1, \ldots, 12\}$, in which one time slot represents a two hour interval, and (b,d) where $H = \{1, \ldots, 24\}$, in which one time slot represents a one hour interval. Given the environmental parameter they are observing, both FLOODNET nodes 1 and 2 must optimize their energy to take samples of greater importance by adapting their sampling actions as those indicated inside round brackets. In this case, node 1 should increase its sampling rate to maximum within regions 1.1 and 1.2 in order to capture the incoming and outgoing tides, and reducing it to either a moderate rate or a minimum during other regions. On the other hand, node 2 should adjust its sampling rate by: (i) increasing it to its highest frequency within regions 1.1, 1.2, 1.3, 2.1, and 2.2, (ii) decreasing it to its moderate rate within regions 3.1, 4.1, 4.2, 4.3, and 4.4, and (iii) decreasing it further till its lowest value within other regions.

The bigger the number of the time slots, the more flexibly the nodes can adapt their sampling schedules. Therefore, in order to find the optimal sampling actions, the set of time slots must be set to $H = \{1, \ldots, |Y_i|\}$, where Y_i is node 1's or 2's set of observations and $|Y_i|$ is the number of observations (or samples). However, the optimal solution within this setting is only computationally feasible if the size of the adaptive sampling problem is very small (more details of which are given in section 3.4.1).

A node's preferences express the satisfaction of any particular action when faced with a choice between different alternatives. In our case, the actions correspond to the different sampling rates that a node may choose to perform within any particular time slot, and the preferences express the information content of the data collected by performing the corresponding actions. A preference structure brings together all the alternatives, $\mathcal{V}$, and represents a node's preferences over the set of possible outcomes. Now, there are

[1]Note that we also assume that the communication costs do not depend on the quantity of samples taken. However, more complex relationships will be modeled in the next chapter for the inter-related adaptive sampling, transmitting, forwarding, and routing problem.

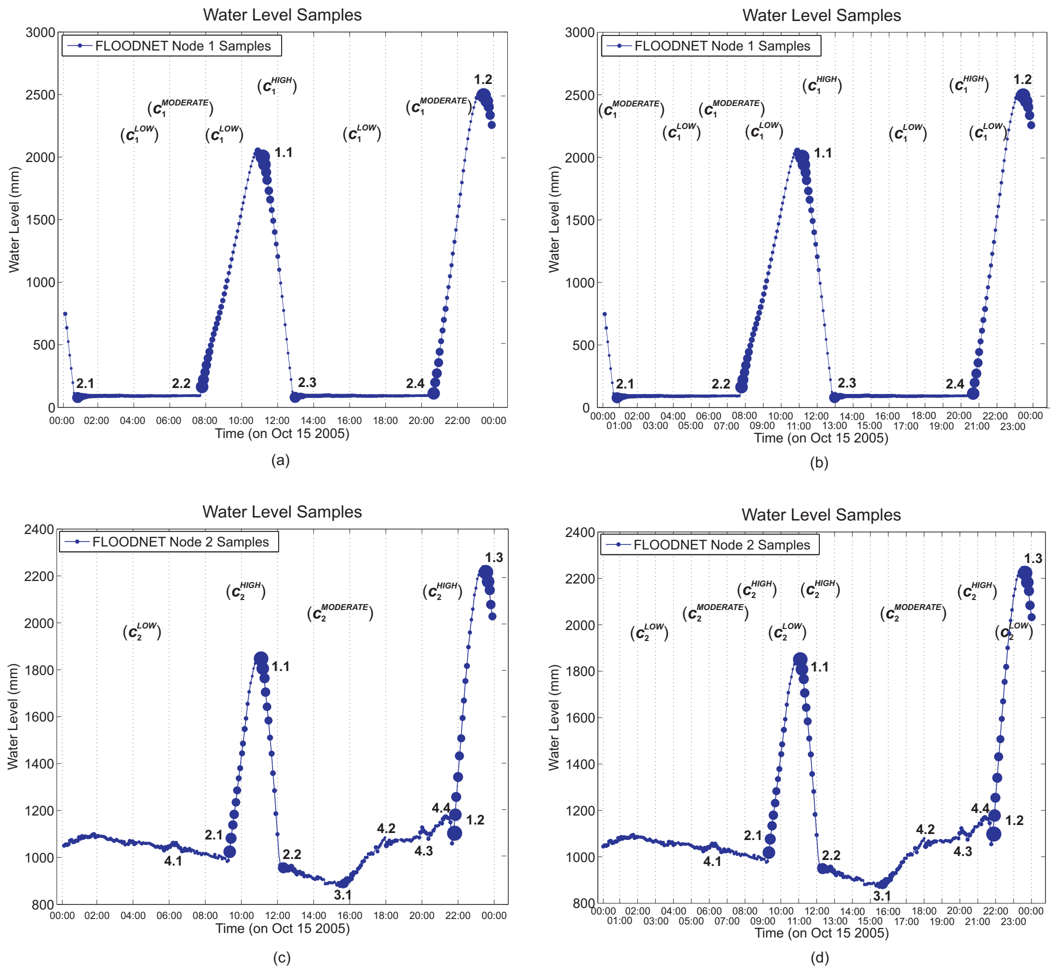


FIGURE 3.2: The set of adaptive sampling actions that (a,b) FLOODNET node 1 and (c,d) FLOODNET node 2 should use within any specific time slot on Oct 15th 2005.

several choices that can be made regarding the definition of a mathematical model for preference structures (see Chevaleyre et al. (2006) for a review), but here we choose a simple cardinal structure since it allows a node to make individual comparisons between its sampling actions (i.e. whether node i is obtaining greater information value by sampling at rate c_i^j rather than c_i^k). Furthermore, in the next chapter, it will also allow comparisons between multiple nodes (i.e. whether node i is obtaining a greater information value by sampling at rate c_i^j than node l operating at c_l^j).

In more detail, a cardinal preference structure consists of a valuation function (i.e. a utility function or a mathematical function used to calculate the value or goodness of a certain action taken by nodes) given by $v : \mathcal{V} \rightarrow \mathcal{Val}$, where $\mathcal{Val}$ is a set of numerical values (typically, $\mathbb{N}, \Re, [0, 1]$, or $\Re^+$).

3.3 Information Metric

Building on the problem description above, an algorithm needs a way to value the various observations that the nodes may make. As mentioned in section 2.3, we use Fisher information to determine the value of previously collected and temporally correlated observations and this is then used to decide on the next sampling plans of each node. If at any point in time, we are able to calculate an estimate of the value of the environmental parameter being sensed, and this estimate is represented by a predictive distribution with mean, $\widehat{\mu}(t)$, and variance, $\widehat{\sigma}^2(t)$, then the mean Fisher information over any period of time between t_1 and t_2 is given by:

$$FI = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} \frac{1}{\widehat{\sigma}^2(t)} \partial t \tag{3.2}$$

The estimated value of the environmental parameter between times t_1 and t_2 is informed by the samples that the node actually takes, and in the next section we specifically describe how we can perform this estimation in a principled Bayesian framework using GP regression.

Finally, we remark that we consider the value of the information collected by the sensor network as a whole to simply be the sum of the information collected by each individual node and, thus, we are explicitly not considering correlations between different nodes. Relaxing this assumption is a focus of our future work, and we discuss it in more detail in section 5.2.

3.3.1 Gaussian Process Regression

As described above, in order to calculate the mean Fisher information, we must use the actual (and possibly noisy) samples taken by the node to estimate the value of the environmental parameter being sensed over a continuous period of time (including times between those at which samples were actually taken). Furthermore, this estimate must represent a full predictive distribution with both a mean and a variance. Hence, we use GP regression to generate this estimate. This principled approach allows us to perform Bayesian inference about functions; in our case, the function representing the value of the environmental parameter over time (Rasmussen and Williams, 2006). Such techniques have a long history in geospatial statistics (Cressie, 1993), and more recently have been used as a generic non-parametric probabilistic model for spatially correlated phenomena (Guestrin et al., 2005; Ertin, 2007). In contrast, here we use it with temporally correlated node's samples.

In more detail, a GP regression takes as inputs a set of g sampling points, $X = \{x^1, \ldots, x^g\}$, and a set of g noisy observations or target values, $Y = \{y^1, \ldots, y^g\}$ (both are termed the training set) (Mackay, 1998; Seeger, 2004). Given a covariance function that describes the correlation between sensor readings at any two times (we shall discuss this function in more detail later), the GP is able to infer the posterior predictive distribution (i.e. the conditional distribution available after the GP has observed the training set and the covariance function) of the environmental parameter at any other set of m test points, $\mathfrak{X} = \{\mathfrak{x}^1, \ldots, \mathfrak{x}^m\}$. This predictive distribution is represented by a mean, $\widehat{\mu}(\mathfrak{X}) = \{\widehat{\mu}(\mathfrak{x}^1), \ldots, \widehat{\mu}(\mathfrak{x}^m)\}$, and a variance, $\widehat{\sigma}^2(\mathfrak{X}) = \{\widehat{\sigma}^2(\mathfrak{x}^1), \ldots, \widehat{\sigma}^2(\mathfrak{x}^m)\}$, given by:

$$\widehat{\mu}(\mathfrak{x}^j) = \mathcal{c}^{\mathrm{T}}(\mathcal{C} + \sigma^2 I_g)^{-1} Y \tag{3.3}$$

$$\widehat{\sigma}^2(\mathfrak{x}^j) = C(\mathfrak{x}^j, \mathfrak{x}^j) - \mathcal{c}^{\mathrm{T}}(\mathcal{C} + \sigma^2 I_g)^{-1} \mathcal{c} \tag{3.4}$$

where $\mathcal{C}$ is a $g \times g$ matrix for the training set covariances, $\mathcal{c}$ is a $g \times 1$ vector identifying the training-test set covariances (i.e. a row vector of the covariances of $\mathfrak{x}^j$ with all variables in the training set), $C(\mathfrak{x}^j, \mathfrak{x}^j)$ is the covariance of $\mathfrak{x}^j$, I_g is a $g \times g$ identity matrix, and σ^2 is an added Gaussian noise of the training set accordingly.

The variance is then used to calculate the mean Fisher information of the interval spanned by $\mathfrak{X}$ of m test points. Thus, the mean Fisher information, over the interval $\mathfrak{X}$, conditioned on the set of observations represented by the sensor readings, Y, taken at times X, is given by:

$$FI(X) = \frac{1}{m} \sum_{j=1}^{m} \frac{1}{\widehat{\sigma}^2(\mathfrak{x}^j)} \tag{3.5}$$

Note that this is a discretization of equation 3.2, whose resolution is determined by the number of prediction points that cover the period of interest (i.e. by the value of m). Furthermore, note that the value of Fisher information calculated above does not depend directly on the actual samples (since there is no dependence on Y in the expression for $\widehat{\sigma}^2(\mathfrak{x}^j)$ in equation 3.4).

3.3.2 Covariance Functions

A key assumption of the GP regression technique described above is that points in time within $\mathfrak{X}$ that are close together are likely to have similar predicted values within $\widehat{\mu}(\mathfrak{X})$. Furthermore, training points in X which are close to estimation points in $\mathfrak{X}$ are those that are most informative. This notion of closeness or similarity is defined by a covariance function. The covariance function is a crucial ingredient within GP regression. It allows prior information concerning the domain problem to be incorporated into the inference (for instance that the environmental parameter being sensed varies smoothly over time and/or is periodic), and thus, it influences the quality of the predictions made. While much empirical guidance for the choice of covariance functions does exist, there is no formal methodology for determining this choice automatically (Rasmussen, 2004).

In our case, we choose a commonly used covariance function termed *squared exponential* or *Gaussian* covariance function:

$$C_{sqe}(x, \mathfrak{x}) = v_{sqe} \cdot \exp\left[-\frac{(x-\mathfrak{x})^2}{\lambda_{sqe}^2}\right] \tag{3.6}$$

where v_{sqe} is the weighting of this term, and λ_{sqe} is the length scale or correlation length, that represents the length along which successive target values are strongly correlated. We choose this function because it is infinitely differentiable and, thus, capable of modelling smoothly varying environmental parameters. Moreover, Girard (2004) shows that this covariance function has good general modelling abilities and predictive performance comparable with that of neural networks. However, this choice is not fundamental to our algorithms and alternatives such as *rational quadratic*, *linear*, or *exponential* could also be used in other cases. Furthermore, these alternative forms can also be combined together (by summation or multiplication) to derive a rich family of possible covariance functions[2].

Since many WSNs monitoring environmental phenomena show a periodical pattern between days in their readings (as we have with FLOODNET tide data), we also use a *periodic* covariance function:

$$C_{per}(x, \mathfrak{x}) = v_{per} \cdot \exp\left[-\frac{2\sin^2\left(\frac{x-\mathfrak{x}}{\mathfrak{p}}\right)}{\lambda_{per}^2}\right] \tag{3.7}$$

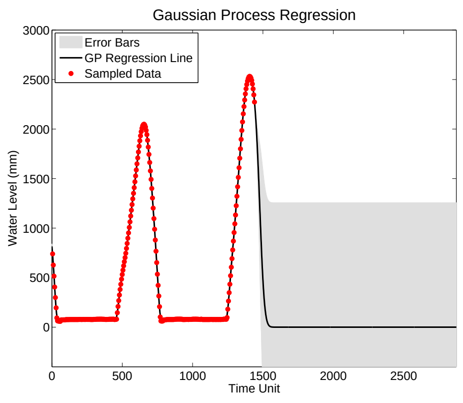
where v_{per} is the weighting of this term, $\mathfrak{p}$ is the periodicity of the data, and λ_{per} is the length scale. In addition, an *independent* covariance function with weighting v_{noi} is used to represent Gaussian distributed noise in sensor readings:

$$C_{noi}(x, \mathfrak{x}) = v_{noi} \cdot \kappa, \text{ where } \kappa = \begin{cases} 1 & \text{if } x = \mathfrak{x} \\ 0 & \text{if } x \neq \mathfrak{x} \end{cases} \tag{3.8}$$

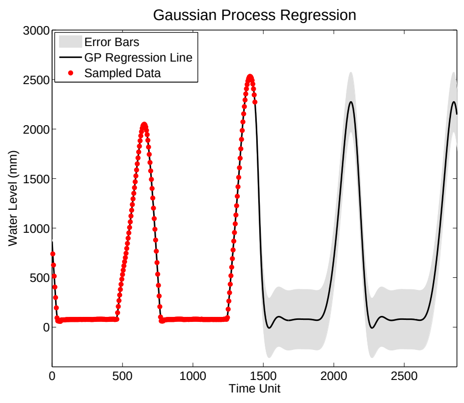
These three separate terms are combined by simply adding them together, and this combination is shown to provide accurate water level estimates for the FLOODNET domain. For example, figure 3.3 illustrates this by comparing (a,c) the case where $\mathcal{C} = C_{sqe} + C_{noi}$ which excludes the periodic term, and (b,d) where $\mathcal{C} = C_{sqe} + C_{per} + C_{noi}$ which includes it. In both cases, the markers represent the nodes' samples, the solid line indicates the mean of the predictive distribution, and the shaded area represents its variance.

The weightings, v_{sqe}, v_{per}, and v_{noi}, characteristic lengths, λ_{sqe} and λ_{per}, periodicity, $\mathfrak{p}$, and sensor noise, σ^2, are collectively termed hyperparameters, and in general, we do not

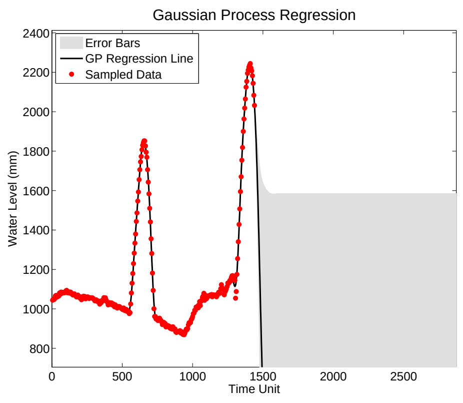
[2]In general, if we use a covariance function with sufficient flexibility, then we can be confident that the machinery of Bayesian inference will automatically reduce the weighting of terms that are not necessary to describe the data that has been observed.



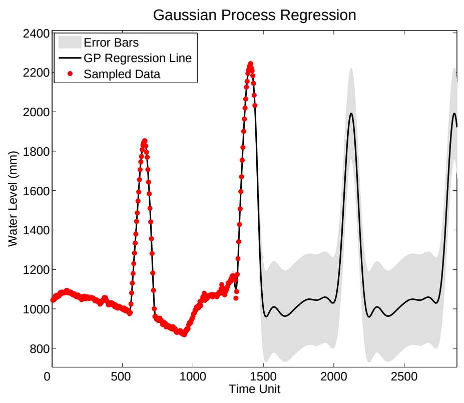
(a) GP regression on FLOODNET node 1 using a covariance function without period terms such that $\mathcal{C} = C_{sqe} + C_{noi}$.



(b) GP regression regression on FLOODNET node 1 using a covariance function with period terms such that $\mathcal{C} = C_{sqe} + C_{per} + C_{noi}$.



(c) GP regression on FLOODNET node 2 using a covariance function without period terms such that $\mathcal{C} = C_{sqe} + C_{noi}$.



(d) GP regression regression on FLOODNET node 2 using a covariance function with period terms such that $\mathcal{C} = C_{sqe} + C_{per} + C_{noi}$.

FIGURE 3.3: Gaussian process regression applied to FLOODNET nodes' water level data on Oct 15th 2005. One time unit represents a minute interval.

TABLE 3.1: Example hyperparameters for the covariance functions learned from FLOODNET data.

Parameter	**Value**
Variance for C_{sqe} (v_{sqe})	$\log 112.2733$
Variance for C_{per} (v_{per})	$\log 956.3368$
Variance for C_{noi} (v_{noi})	$\log 3.3753$
Correlation length scale for C_{sqe} (λ_{sqe})	$\log 19.4994$
Correlation length scale for C_{per} (λ_{per})	$\log 0.7882$
Unit period scale for C_{per} $(\mathfrak{p})$	$\log 736.3052$

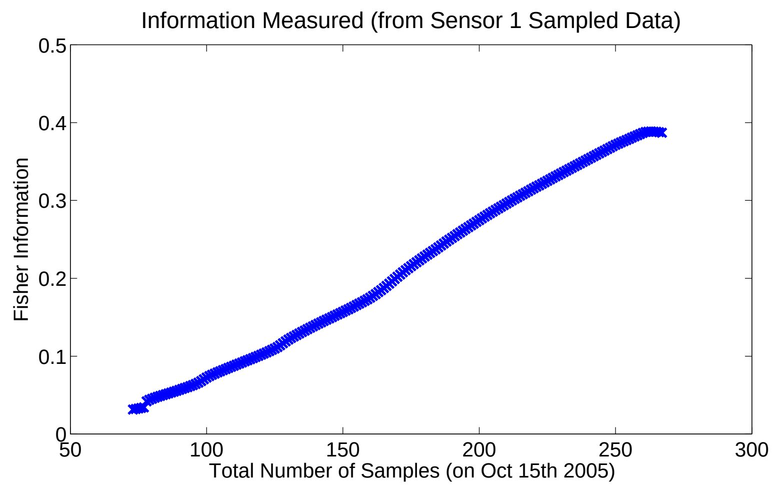


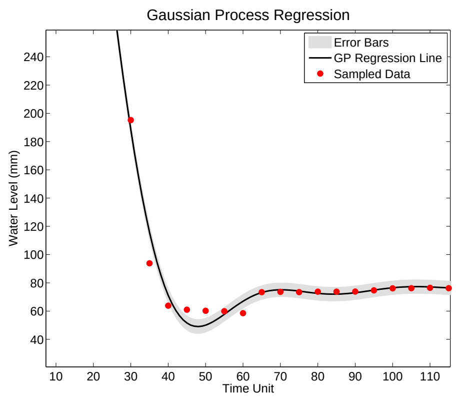
FIGURE 3.4: Fisher information value (FI) gathered over one day period plotted against the number of samples. As expected, the value of information generally increases (i.e. we are more certain in the value of the environmental parameter being sensed) as the sensor nodes takes more samples. Note that in the case of a GP with fixed hyperparameters, we would expect to observe a concave function since the observations are correlated.

know their values *a priori*. However, a number of techniques can be used to infer their values from the sensor readings themselves. Within the GPML[3] package that we use here, techniques for learning the hyperparameters are based on maximisation of the log likelihood function using an efficient conjugate gradient-based optimization algorithm (Bishop, 2006). However, there is no guarantee that the marginal log likelihood does not suffer from multiple local optima. Thus, we use a multi-start process for setting good initial hyperparameters whereby we restart the maximisation of log likelihood from a number of different starting points, and select the one that results in the maximum log likelihood[4]. We perform this learning prior to performing regression whenever new data is present, and table 3.1 shows an example of these hyperparameters.

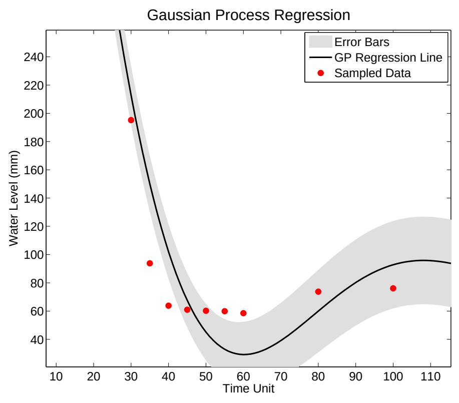
Within the FLOODNET domain, whenever the water level raw data points are closely related (i.e. they have a small covariance matrix or they are more frequently sampled), the variances of the estimated values, $\widehat{\sigma}^2(\mathfrak{X})$, will decrease. The Fisher information value will, on the other hand, increase as it is the inverse uncertainty of the estimate (see figure 3.4). To illustrate this, consider the scenario illustrated in figure 3.5. Assume FLOODNET nodes 1 and 2, each of which has a set of twelve noisy measurements per hour for a given day. Thus, each of the nodes samples at five minute intervals such that the total number of samples (g) on that day is 288 (12 samples/hour x 24 hours). Now, in order to find out the value of this set of data, the node performs the GP regression utilizing the sampling and target sets (sets X and Y correspondingly) to produce the predictive distribution with mean $\widehat{\mu}(\mathfrak{X})$ and variance $\widehat{\sigma}^2(\mathfrak{X})$. Given this, equation 3.5

[3]`http://www.gaussianprocess.org/gpml/`, checked on 02/02/2009.

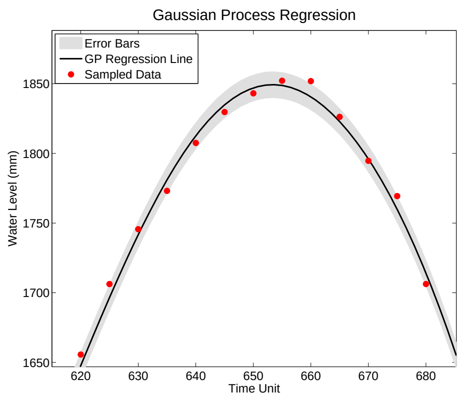
[4]In future work we intend to investigate the use of fully Bayesian approaches to maintain a distribution over possible hyperparameter values (Osborne et al., 2008).



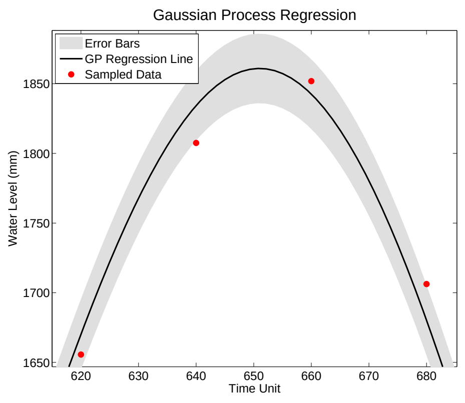
(a) Regression on sampled data taken between time unit 30 and 115, with 18 samples. FLOODNET node 1 collects Fisher information value of $FI = 0.20290$.



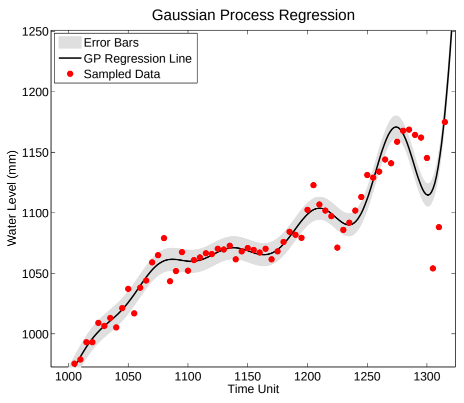
(b) Regression on sampled data taken between time unit 30 and 115, with 9 samples. FLOODNET node 1 collects Fisher information value of $FI = 0.40650 \cdot 10^{-1}$.



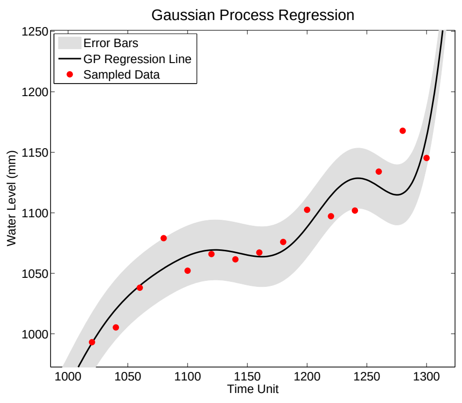
(c) Regression on sampled data taken between time unit 620 and 680, with 13 samples. FLOODNET node 2 collects Fisher information value of $FI = 0.10683$.



(d) Regression on sampled data taken between time unit 620 and 680, with 4 samples. FLOODNET node 2 collects Fisher information value of $FI = 0.40357 \cdot 10^{-1}$.



(e) Regression on sampled data taken between time unit 1005 and 1315, with 63 samples. FLOODNET node 2 collects Fisher information value of $FI = 0.10680$.



(f) Regression on sampled data taken between time unit 1005 and 1315, with 15 samples. FLOODNET node 2 collects Fisher information value of $FI = 0.40256 \cdot 10^{-1}$.

FIGURE 3.5: GP regression example using sampled data collected from FLOODNET nodes operating on Oct 15th 2005. One time unit represents a minute interval.

can then be used to determine the information content of the samples.

3.4 Decentralised Information-Based Adaptive Sampling

Given the problem description and information metric, the objective in this work is to now derive an algorithm that can automatically determine the allocation of actions each day, $Alloc$, that will maximise the total mean Fisher information collected by the nodes (i.e. the utilitarian social welfare), subject to the energy constraint (in equation 3.1). To this end, we now present three novel decentralised control algorithms for information-based adaptive sampling that achieve this. Each algorithm represents a different trade-off between computational cost and optimality. We start with an exponential algorithm that maximises the Fisher information metric by performing GP regression on the individual nodes, and progress to a more computationally efficient algorithm that uses a heuristic approach, rather than the GP regression, in order to determine the allocation of actions each day.

Each of these algorithms follows the same broad pattern. On any specific day, each node $i \in I$ may be in one of two modes: (i) *an updating mode* in which the node samples at a predefined maximum rate throughout all time slots, or (ii) *a standard sampling mode* in which it samples according to the allocation, $Alloc_i$. The algorithms use the samples taken whilst the node is in its updating mode to calculate the allocation of actions, $Alloc_i$, to be used whilst the node is in its standard sampling mode. The frequency with which the node enters the updating mode is determined manually by the system designer, and depends on the variability of the environment. In relatively static environments the allocation will remain valid for sometime and, thus, updating can occur less frequently. In more dynamic settings the allocation must be updated more often[5]. Note that since the node samples at its maximum rate whilst in the updating mode, then the more often updating is performed, the less samples can be taken during any day whilst the node is in its standard sampling mode (in order to maintain energy neutral operation). In the experiments that we present in section 3.5.4 the nodes update once every three days.

3.4.1 The Optimal Algorithm

A first approach to providing a decentralised algorithm is to simply deploy the GP regression algorithm on each node, and then to find the subset of the samples that were taken whilst the node was in its updating mode that maximises our Fisher information metric (whilst also satisfying the energy constraints of the node when it is in its standard

[5]We remark that setting this parameter automatically is part of our future work. We expect that this can be achieved by measuring the information content of samples collected each day, and re-entering the updating mode when this shows a significant departure from that which is expected.

sampling mode). Thus, in more detail, if X_i is the set of sampling points taken whilst node i was in its updating mode, we wish to solve:

$$X_i^* = \arg\max_{X_i} FI(X_i) \tag{3.9}$$

subject to the constraint that $X_i^* \subseteq X_i$ and $|X_i^*| \cdot e_i^{\mathrm{s}} \leq B_i$. Given this subset of sampling points, we then calculate the allocation of sampling actions of node i, $Alloc_i = \{a_i^1, \ldots, a_i^w\}$, by simply counting the number of sample points in each time slot such that:

$$a_i^k = \sum_{j=1}^{|X_i|} \begin{cases} 1 & \text{if } x_i^j \in X_i^* \text{ and in time slot } k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in H \tag{3.10}$$

Now, a naïve approach to finding this optimal subset is to simply enumerate all possible combinations. This approach, however, is too computationally intensive and works only for very small problems as it very rapidly becomes intractable. For instance, in the case of FLOODNET in which a node takes 288 samples a day whilst in its updating mode, but can only take 144 samples a day in its standard sampling mode, this algorithm would need to evaluate more than 10^{100} (C_{144}^{288}) solutions. This is clearly impossible to compute in a reasonable amount of time regardless of processor speed (for more details, see section 3.5.4.2 for the run time performance of the algorithms that we present in this section).

3.4.2 The Greedy Algorithm

Since the naïve enumeration approach is infeasible, we need a smarter means of tackling this problem. Thus, we devise a greedy adaptive sampling algorithm that again deploys the GP regression algorithm on the sensor node, but then works by allocating one additional sampling point at a time until there are no more samples to add. The allocated sampling points cannot be altered in subsequent iterations (i.e. they are fixed). This significantly reduces the number of possibilities to compute compared to the naïve optimal algorithm which considers the whole set of combinations of sampling points as possible solutions. For example, in the case discussed above where 144 out of the possible 288 sampling points must be selected, we need only evaluate 31104 solutions (as compared to C_{144}^{288} solutions above). Nevertheless, this method is still reasonably slow to be run on nodes with the type of limited computational power found in WSNs in general and FLOODNET in particular.

In more detail, this algorithm works as follows (see algorithm 1). At setup time, the vector variable *gpFI* that temporarily records all the evaluated information values (FI as in equation 3.5), is initialized to a null set (line 1). Each node $i \in I$ then presets a number of samples (*preSamp*) and equally distributes them into its time slots. Following

Algorithm 1 Greedy adaptive sampling.

```
1: $gpFI \leftarrow \{\}$
2: while $addSamp > 0$ do                                ▷ While there is an additional sample to allocate
3:     for each $sampToTry \in remSP$ do                 ▷ Iterates each remaining sampling point
4:         $preSampTemp \leftarrow preSamp \cup sampToTry$
5:         $gpFI \leftarrow gpFI \cup \textsc{CalcFIUsingGP}(preSampTemp)$   ▷ Calculates the GP information value
6:     end for
7:     $[maxFI, indexOfNextSP] \leftarrow \textsc{Max}(gpFI)$
8:     $nextSP \leftarrow \textsc{NextSamplingPoint}(indexOfNextSP)$     ▷ Finds $nextSP$
9:     $preSamp \leftarrow preSamp \cup nextSP$          ▷ $nextSP$ is included into $preSamp$
10:    $remSP \leftarrow remSP \backslash nextSP$        ▷ $nextSP$ is excluded from the remaining sampling point, $remSP$
11:    $gpFI \leftarrow \{\}$
12:    $addSamp \leftarrow addSamp - 1$
13: end while
```

this initialization phase, the node then uses its energy resources to iteratively sample *addSamp* times more from the possible remaining sampling points *remSP* (lines 2 and 3). On each iteration, the node evaluates the information value of each remaining sampling point (line 5). It then allocates one sample at the sampling point *nextSP* where the information value is increased the most (lines 7 and 8). At the end, the chosen *nextSP* is included into the vector variable *preSamp* (line 9). It is then excluded from the vector variable *remSP* and the variable *gpFI* is cleared (lines 10 and 11 respectively). This repeats until there are no more samples to add.

The vector variable *preSamp* eventually contains the greedy selection of sampling points and, thus, the allocation of sampling actions of node i, $Alloc_i = \{a_i^1, \dots, a_i^w\}$, can again be found by simply counting the number of sample points in each time slot such that:

$$a_i^k = \sum_{j=1}^{|X_i|} \begin{cases} 1 & \text{if } x_i^j \in preSamp \text{ and in time slot } k \\ 0 & \text{otherwise} \end{cases} \qquad \forall k \in H \tag{3.11}$$

Finally, we remark that the GP regression algorithm itself is relatively computationally expensive and, thus, we next present a heuristic algorithm that enables the sensor nodes to use a simpler means of valuing information in order to gain a faster performance.

3.4.3 The Heuristic Algorithm

We first describe a simplified valuation function that avoids the need to perform GP regression on the node, and we then present the algorithm that we use to select sampling points in order to maximise it.

3.4.3.1 The Valuation Function

In this algorithm, we value information heuristically rather than computing it using equation 3.5. We do this because the iterative process of calculating the information value using the GP regression technique is computationally expensive, and both the algorithms presented above require that this process is performed repeatedly. Specifically, we use simple linear regression and develop an information function that is based on the standard deviation of the best-fit regression line. This is appropriate, since given a small enough time window, the relationship between the time and environmental observations data can be approximated as a piecewise linear function.

Using this method, the uncertainty in a set of sensor readings is expressed in *confidence bands* about the linear regression line. The confidence band has the same interpretation as the standard deviation of the residuals (termed Sd in equation 3.13, where g represents the number of data points and $\hat{y}$ is the new value of y calculated from the newly found slope, b^1, and intercept, b^0, variables), except that it varies according to the location along the regression line. The distance of the confidence bands from the regression line, $\tau_i^t(X_i)$, at point x_i^t conditioned on the set of sensor readings, Y_i, that is collected by node i at times X_i, is:

$$\tau_i^t(X_i) = Sd(X_i)\sqrt{\frac{1}{g_i} + \frac{(x_i^t - \bar{x}_i)^2}{\sum_{j=1}^{g_i}\left(x_i^j - \bar{x}_i\right)^2}} \tag{3.12}$$

where x_i^t is the location along the x-axis data points where the distance is being calculated. $\bar{x}_i$ and $\bar{y}_i$ are the mean values of X_i and Y_i respectively.

$$Sd(X_i) = \sqrt{\frac{\sum_{j=1}^{g_i}\left(y_i^j - \hat{y}_i^j\right)^2}{g_i - 2}} \tag{3.13}$$

where

$$\hat{y}_i^j = b^0(X_i) + b^1(X_i) \cdot x_i^j \tag{3.14}$$

and

$$b^0(X_i) = \bar{y}_i - b^1(X_i) \cdot \bar{x}_i \tag{3.15}$$

and

$$b^1(X_i) = \frac{\sum_{j=1}^{g_i}\left\{\left(x_i^j - \bar{x}_i\right)\left(y_i^j - \bar{y}_i\right)\right\}}{\sum_{j=1}^{g_i}\left(x_i^j - \bar{x}_i\right)^2} \tag{3.16}$$

In order to perform this simple linear regression properly, the input must consist of at least three data points. This is because if there are only two data points they will produce a smooth linear regression line (with no standard deviation), while anything less than that will result in invalid inputs. For these reasons, we enforce the fact that a node must at least sample once in every time slot (defined as the minimum sampling rate and, therefore, $B_i \geq w \cdot e_i^{\mathrm{s}}$, $\forall i \in I$, where as defined in section 3.2, w is the number of time slots and e_i^{s} is the energy required by node i to perform one sample). In this way, given the standard deviation and the confidence bands, we are able to tell whether one set of observations is more valuable than another which, in turn, allows us to define a value associated to every action.

Given the expressions above, we can now derive the *total deviation*, Td^{g_i}, for a set of g_i data points, X_i, collected in time slot k, by calculating the area between the confidence bands, and this total deviation represents our uncertainty over this period. Specifically, we use a trapezoidal numerical integration method for this (Rabinowitz and Davis, 2006). The trapezoid approximation (or trapezoid sum, Ts) of $\int_a^b f(x)\ \partial x$, that is associated with the partition $a = x^1 < x^2 < ... < x^{g_i} = b$ is given by:

$$\begin{aligned} Ts &= \frac{1}{2}[f(x^1) + 2f(x^2) + ... + 2f(x^{t-1}) + f(x^{g_i})]\Delta x \\ &= \frac{1}{2}[f(x^1) + 2\sum_{t=2}^{g_i-1} f(x^t) + f(x^{g_i})]\Delta x \end{aligned} \tag{3.17}$$

and, thus, we are now able to derive:

$$Td^{g_i}(k) = \frac{1}{2}[2\tau_i^1(X_i) + 4\sum_{t=2}^{g_i-1} \tau_i^t(X_i) + 2\tau_i^{g_i}(X_i)]\Delta x_i \tag{3.18}$$

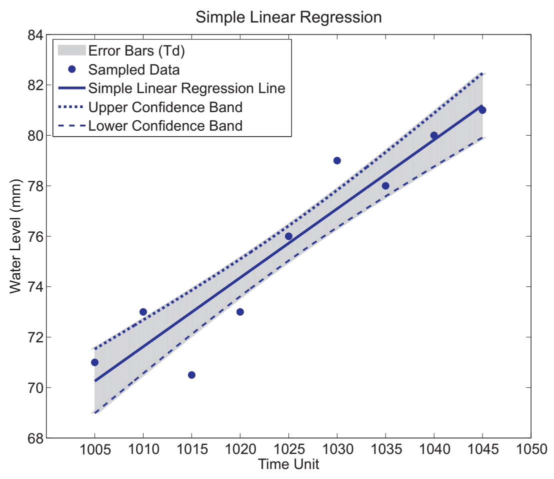
For example, consider the case shown in figure 3.6 that uses real data collected from FLOODNET node 1 on Oct 15th 2005 between time unit 1005 and 1045 (where one time unit represents a minute interval). Figure 3.6(a) shows a case where 9 samples are taken and figure 3.6(b) shows another case where only 5 are taken. The X axis represents the time unit, whilst the Y axis represents the water level. The solid line denotes the simple linear regression line, whilst the curved dashed lines demarcate the confidence bands (it represents the boundaries of all possible straight lines). In the case where 9 samples are taken, the procedure just described allows us to calculate the total deviation, Td, as 14.6050, while in the case of 5 samples, it is 35.5968. In both cases, the total deviation is represented by the shaded area between the confidence bands; less uncertainty is denoted by a smaller area.

Given this total deviation, we can now simply derive the gain in information value (or the decrease in uncertainty) when different sampling decisions are made. More formally, $Gain_i^j(k)$ is defined as the reduction in total deviation that node i can achieve by taking samples at rate c_i^j (and, hence, collecting g_i samples) rather than the minimum sampling rate, c_i^1, in time slot k, and is given by:

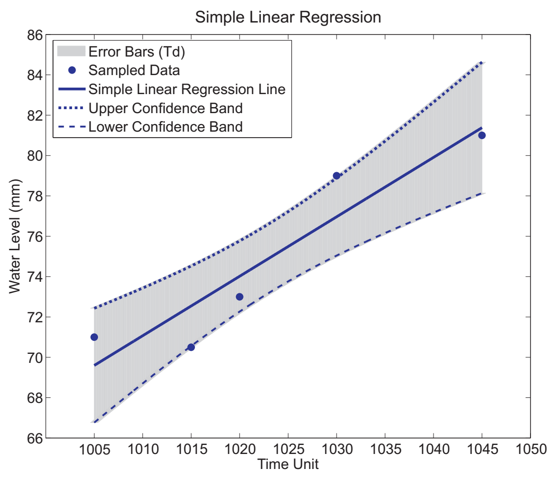
$$Gain_i^j(k) = Td^{c_i^1}(k) - Td^{c_i^j}(k) \quad \text{where } c_i^j \in C_i, i \in I, k \in H \tag{3.19}$$

This minimum sampling rate is applied as a basis where a node gains zero value.

The data values for each node are often best represented in a table format. To this end, let G_i be a table with s_i rows numbered from 1 to s_i, and w columns, where s_i is the



(a)



(b)

FIGURE 3.6: Example application of the linear regression based valuation metric applied to data collected from FLOODNET node 1 on Oct 15th 2005 between time unit 1005 and 1045. Two cases are shown; (a) where 9 samples are taken and (b) where only 5 samples are taken.

different sampling rates of node i and w is the number of time slots. The element of the table that is in the j^{th} row and the column with label k is thus $Gain_i^j(k)$. This value indicates the decrease in uncertainty (or the reduction in total deviation) of node i if it chooses to perform c_i^j sampling action in time slot k rather than its minimum sampling rate, c_i^1.

As described earlier, when in its updating mode, each node $i \in I$ samples at its maximum rate, $c_i^{s_i}$. Now, by taking subsets of samples (corresponding to the set of actions specified in table G_i) from the full set and performing the linear regression on these subsets, we obtain a new total deviation for each subset. The values that will be assigned to the table are the total deviation difference between sampling at the minimum rate and at other rates. For instance where $column = k$, if the total deviation that is produced with a subset of samples c_i^j taken during time slot k has a value of $Td^{c_i^j}(k)$, while that of a minimum sampling rate c_i^1 taken between the same period is $Td^{c_i^1}(k)$, then the value inside $column = k$ and $row = j$; that is $Gain_i^j(k)$, will be $Td^{c_i^1}(k) - Td^{c_i^j}(k)$.

3.4.3.2 The Algorithm

We now focus on how to search for an allocation of a node's actions that maximises the information metric described above. For this purpose, we introduce V_i as a $s_i \times w$

matrix with s_i number of actions of node i and w number of time slots:

$$V_i = \begin{bmatrix} v_i^{11} & v_i^{12} & \dots & v_i^{1w} \\ \vdots & \vdots & \ddots & \vdots \\ v_i^{s_i1} & v_i^{s_i2} & \dots & v_i^{s_iw} \end{bmatrix} \quad D_i = \begin{bmatrix} d_i^{11} & d_i^{12} & \dots & d_i^{1w} \\ \vdots & \vdots & \ddots & \vdots \\ d_i^{s_i1} & d_i^{s_i2} & \dots & d_i^{s_iw} \end{bmatrix}$$

such that v_i^{jk} represents the value that node i will get if it chooses to perform action c_i^j in time slot k (i.e. $Gain_i^j(k)$). D_i is a matrix of binary values and each of the elements corresponds to a decision variable (a "1" represents a state where node i carries out the corresponding c_i^j action in time slot k, whilst a "0" represents another state where the node does not carry out the corresponding c_i^j action). For instance, when $d_i^{11} = 1$ then this node i chooses to perform action c_i^1 in time slot 1. This also means that $d_i^{j1} = 0, \forall j \in C_i \backslash c_i^1$.

In more detail, the objective function to be maximised is defined in equation 3.20. The constraint in equation 3.21 states that every node $i \in I$ can only elect one action at any particular point of time, whereas that in equation 3.22 states that the total number of samples taken by it must not exceed the maximum number of samples it can take on that day:

$$D_i^* = \arg \max_{\{V_i,D_i\}} \sum_{c_i^j \in C_i} \sum_{k=1}^{w} v_i^{jk} \cdot d_i^{jk} \tag{3.20}$$

subject to:

$$\sum_{j=1}^{s_i} d_i^{jk} = 1 \quad \forall k \in H \tag{3.21}$$

$$\sum_{c_i^j \in C_i} \sum_{k=1}^{w} c_i^j \cdot d_i^{jk} \leq N_i \tag{3.22}$$

where N_i is calculated such that $N_i \cdot e_i^{\mathrm{s}} \leq B_i$ as described in equation 3.1.

This problem, as formulated above, can be cast as a person-task assignment problem[6] (Yong et al., 1993). Given this insight, we can solve the problem using *binary integer programming* (BIP) (Chen et al., 2000), which is a subset of linear programming. A popular method to solve this numerically is the *simplex* algorithm and in this case we exploit the *GNU Linear Programing Kit*[7] (GLPK) to do so.

Having described the techniques that we use, we now seek to present the rest of the heuristic information-based adaptive sampling algorithm (see algorithm 2 and figure 3.7). Specifically, the algorithm, which is distributed and installed on each node in the network, provides a means for the individual nodes to adjust their own sampling rates

[6]In the assignment problem, we want to assign a set of people to do a set of tasks. Each person takes a certain number of minutes to do a certain task, or cannot do a particular task at all, and each person can be assigned to exactly one task. The ultimate aim, here, is to minimize the total time taken to do all of the tasks.

[7]`http://www.gnu.org/software/glpk/`, checked on 02/02/2009.

Algorithm 2 Heuristic information-based adaptive sampling.

```
1: $updSSched \leftarrow TRUE$                                ▷ Node in updating mode
2: $sRate \leftarrow MAX\_S\_RATE$                            ▷ Samples at maximum rate, $c_i^{s_i}$
3: $readings \leftarrow \{\}$
4: loop
5:     if $sTime = NOW$ then                                  ▷ Time to sample
6:         $readings \leftarrow$ PerformSampling($sTime$)
7:         if $\neg updSSched$ then
8:             $timeSlot \leftarrow$ TimeSlot($sTime$)
9:             $sRate \leftarrow$ GetSRate($timeSlot$)        ▷ Changes sampling rate according to $timeSlot$
10:        end if
11:        SetSTime($sTime + sRate$)                          ▷ Node sets its next sampling time
12:    end if
13:    if $tTime = NOW$ then                                  ▷ Time to transmit
14:        $uError \leftarrow$ CalcUError($readings$)         ▷ Calculates information uncertainty in current readings using (3.18)
15:        if $dateChanged$ then
16:            $daysCount \leftarrow daysCount + 1$
17:            if $updSSched$ then
18:                CalcTdReduction($readings$)                ▷ Computes the gain in information value or the decrease in uncertainty (i.e. the reduction in total deviation for table $G_i$) using (3.19)
19:                FindSSchedule()                            ▷ Optimizes (3.20) using the BIP solver, subject to constraints (3.21) and (3.22) to determine $D_i^*$
20:                $updSSched \leftarrow FALSE$
21:            end if
22:        end if
23:        if $\neg updSSched \wedge$ HasEnoughEnergy() $\wedge (daysCount \geq CONST)$ then
24:            $updSSched \leftarrow TRUE$                    ▷ Node in updating mode
25:            $daysCount \leftarrow 0$
26:            $sRate \leftarrow MAX\_S\_RATE$                ▷ Sets sampling rate to maximum
27:        end if
28:        PerformTransmit($readings$)
29:        SetTTime($tTime + tRate$)                          ▷ Node sets its next transmitting time
30:        $readings \leftarrow \{\}$
31:    end if
32: end loop
```

based only upon their local historical data and remaining energy resources. Now, within the initialization phase, some required variables are set. These include the boolean variable *updSSched* which is set to $TRUE$ to indicate that the node starts in its updating mode, and then having calculated an allocation of sampling actions for subsequent days, enters its standard sampling mode.

Following the initial updating phase, each node $i \in I$ enters an infinite loop state. On each iteration, it checks its sampling and transmitting time. Whenever the current loop represents the time that it needs to sample (line 5 or state 1), the function PerformReading instantiates a new *reading* and attaches it to the end of the variable *readings*. Subsequently, if the node is not in updating mode, its *sRate* is assigned a value equal to the sampling rate in its schedule, corresponding to the appropriate time slot (line 9 or state 2). The node then sets its next sampling time variable, $sTime$.

Inside the same loop iteration, whenever the node is also required to transmit its current readings (line 13 or state 3), it first calculates the total deviation in this set of readings by using the simple linear regression method described earlier (i.e. calling function CalcUError with equation 3.18). Later, if the node detects that it has entered the following day and it is also in updating mode, it will call the function CalcTdReduction (with equation 3.19, see line 18) to compute the reduction in total deviation that the

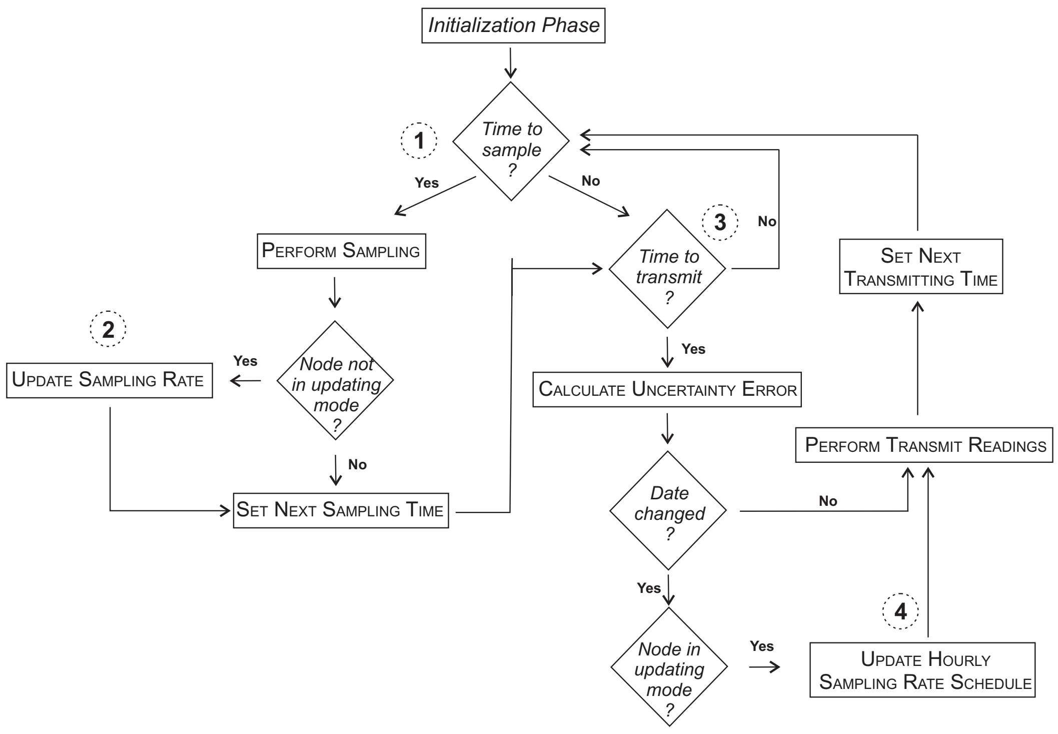


Figure 3.7: State diagram of the algorithm.

node can achieve by taking more samples than the minimum sampling rate. Function FindSSchedule then uses the BIP GLPK solver to evaluate (in real-time) the best allocations of node i's schedule and resources that maximise the total deviation reduction (i.e. equation 3.20), given the node's current energy constraints of equations 3.21 and 3.22 (line 19 or state 4). The allocation of sampling actions, $Alloc_i = \{a_i^1, \ldots, a_i^w\}$, is thus determined by:

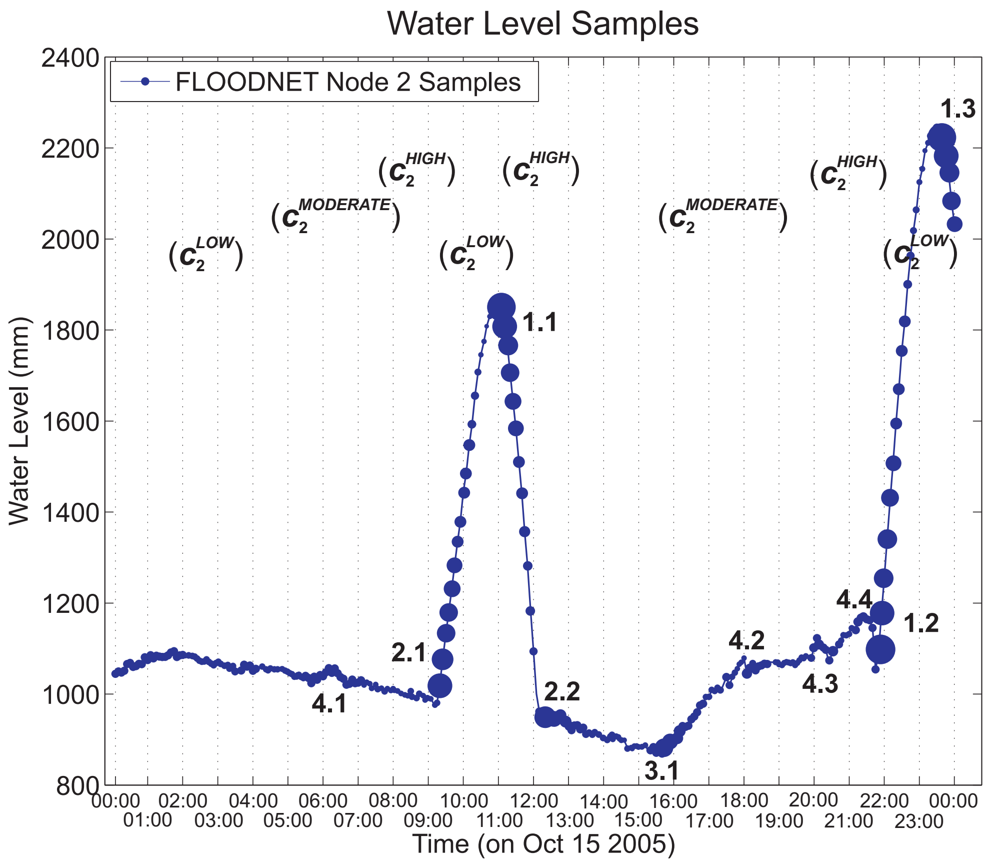
$$a_i^k = c_i^j, \text{ where } j \in \{1..s_i\} | d_i^{jk} = 1, d_i^{jk} \in D_i^* \quad \forall k \in H \tag{3.23}$$

The node then transmits its collected sensor readings and sets its next transmission time variable, $tTime$.

3.4.3.3 Illustrative Example

To illustrate how the algorithm works in practice, reconsider the scenario expected in figure 3.8. FLOODNET node 2 samples at its maximum sampling rate, $c_2^{s_2}$, on Oct 15^{th} 2005 and, thus, collects 288 samples at the end of the day. The node has as well its energy budget, $B_2 = 720$, to perform sampling on the next day, Oct 16^{th} 2005 (requiring $e_2^{\mathrm{s}} = 8$ units of energy per sample). In this case, the day is divided into 24 time slots, where $H = \{1, \ldots, 24\}$. Each time slot thus represents a one hour interval.

In this case, the reduction values in total deviation for table G_2 are chosen arbitrarily for illustrative purposes. However, these values are practically calculated using equation



G_2		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$c_2^{1\,(LOW)}$	= 1 sample	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$c_2^{2\,(MODERATE)}$	= 3 samples	0.126	0.239	0.191	0.200	0.093	0.439	0.706	0.162	0.137	3.114	0.726	4.989	4.551	0.203	0.319	1.992	0.216	0.326	0.845	0.912	1.709	5.776	0.426	4.109
c_2^{j}	= ...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
$c_2^{s_2\,(HIGH)}$	= 12 samples	0.266	0.398	0.326	0.315	0.113	0.719	1.033	0.281	0.306	5.126	1.769	6.777	7.910	0.381	0.590	2.918	0.333	0.420	1.360	1.126	2.911	9.315	0.916	7.911

FIGURE 3.8: The action-value table G_2 of FLOODNET node 2 with $H = \{1, \ldots, 24\}$.

3.19. In this table, the rows represent the different sampling rates (or actions) and the columns represent the time slots. For instance where *column* = 10, if node 2 chooses to sense at sampling rate c_2^2 or $c_2^{s_2}$, it will sample at its moderate or maximum rate (three or twelve times) respectively (assuming $c_2^2 = 3$ and $c_2^{s_2} = 12$), and, in return, will gain a corresponding reduction in total deviation of 3.114 or 5.126 (that is $Gain_2^2(10)$ or $Gain_2^{s_2}(10)$ respectively) compared to if it had only taken samples at its minimum sampling rate, c_2^1, during the same period.

The heuristic information-based adaptive sampling algorithm dictates the node to maximise the reduction in total deviation (or the gain in information value) and, thus, with its remaining energy budget, node 2 will:

- Increase its sampling rate to its maximum, $c_2^{s_2}$, by sampling twelve times during each of time slots 10, 12, 13, 22, and 24 (gaining the reduction of 5.126, 6.777, 7.910, 9.315, and 7.911 respectively),

- Increase its sampling rate to its moderate value, c_2^2, by sampling three times during each of time slots 7, 16, 19, 20, and 21 (gaining the reduction of 0.706, 1.992, 0.845, 0.912, and 1.709 respectively),
- Maintain its minimum sampling rate, c_2^1, by sampling only once during each of the other time slots (gaining no additional reduction).

3.5 Empirical Evaluation

Having detailed the three decentralised algorithms for information-based adaptive sampling, we now turn to their evaluation in order to examine their performance and effectiveness. Specifically, we are interested in comparing the total Fisher information value gathered at the base station and the run time performance (i.e. the computational cost) from each of the individual nodes for the various decentralised control regimes. We chose these measures because they enable us to determine whether by using the same amount of battery energy, our adaptive sampling algorithms permit the nodes to collect more valuable information compared to that of a number of benchmarks.

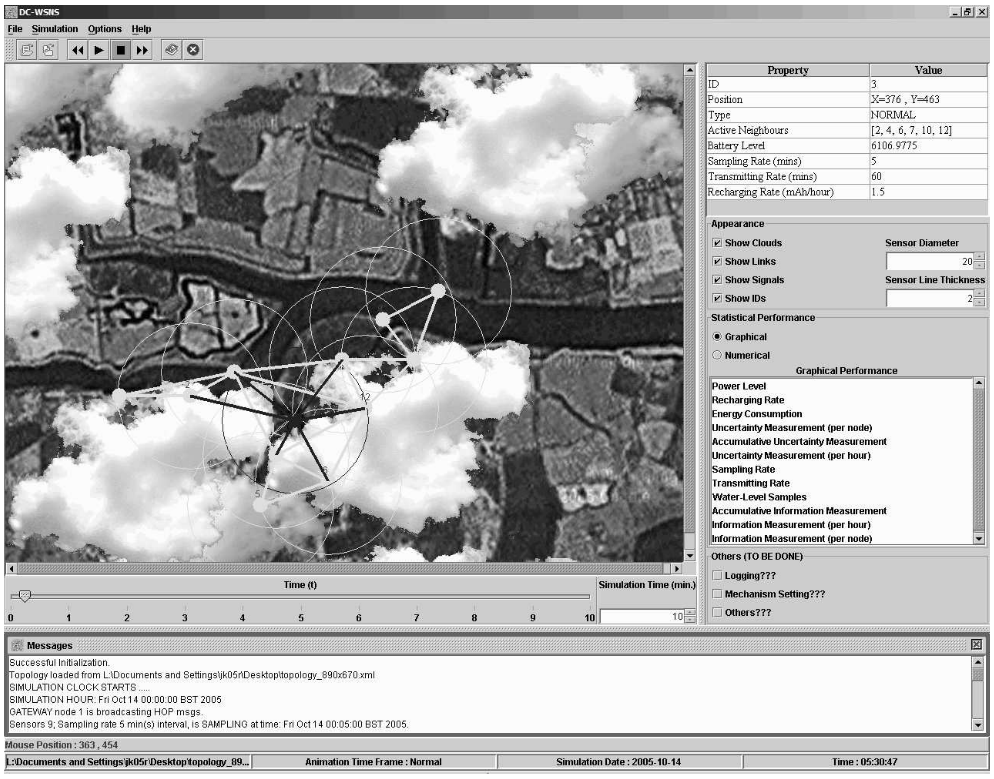
To this end, we first describe the wireless sensor network simulator that was developed in the course of this research. We then detail the experimental setup and the benchmarks, and go on to the actual evaluation.

3.5.1 The Simulator

Simulation is the most widely adopted method for analysing WSNs as it allows algorithms (or protocols) to be properly tested prior to the actual development on-site, and offers the possibility of performing evaluations, with different parameter values and settings, in a rapid and cost-effective way (Khan et al., 2008). Results are, however, highly dependent of the fidelity of the models within the simulation[8].

Due to the absence of a high-fidelity FLOODNET domain simulator, we built a wireless sensor network simulator (called DC-WSNS). This is implemented in the Java programming language and was specifically developed to provide a virtual environment in which sensor nodes can either be scattered randomly or situated at specific locations. The simulator's main parameters can be varied at set up time (e.g. the sampling actions and

[8]Some open source WSN simulators include *ns-2* (Chen, 2007), *SensorSIM* (Park et al., 2000), *J-Sim* (Sobeih et al., 2005), *OBIWAN* (Egea-Lopez et al., 2006), and *OMNeT++* (Colesanti et al., 2007). However, we chose not to use any of them due to many restrictions and difficulties in integrating and tailoring our algorithms into any of these simulations' environment. Moreover, none of them deals with the model of node's renewable energy resources as we have in our case with the FLOODNET nodes.



Property	Value
ID	3
Position	X=376 , Y=463
Type	NORMAL
Active Neighbours	[2, 4, 6, 7, 10, 12]
Battery Level	6106.9775
Sampling Rate (mins)	5
Transmitting Rate (mins)	60
Recharging Rate (mAh/hour)	1.5

[✓ Show Clouds]
[✓ Show Links]
[✓ Show Signals]
[✓ Show IDs]
Sensor Diameter: 20
Sensor Line Thickness: 2
[✓ Graphical] Numerical
Logging???
Mechanism Setting???
Others???
Simulation Time (min.): 10

```
Successful Initialization.
Topology loaded from L:\Documents and Settings\jk05r\Desktop\topology_890x670.xml
SIMULATION CLOCK STARTS .....
SIMULATION HOUR: Fri Oct 14 00:00:00 BST 2005
GATEWAY node 1 is broadcasting HOP msgs.
Sensors 9; Sampling rate 5 min(s) interval, is SAMPLING at time: Fri Oct 14 00:05:00 BST 2005.
```

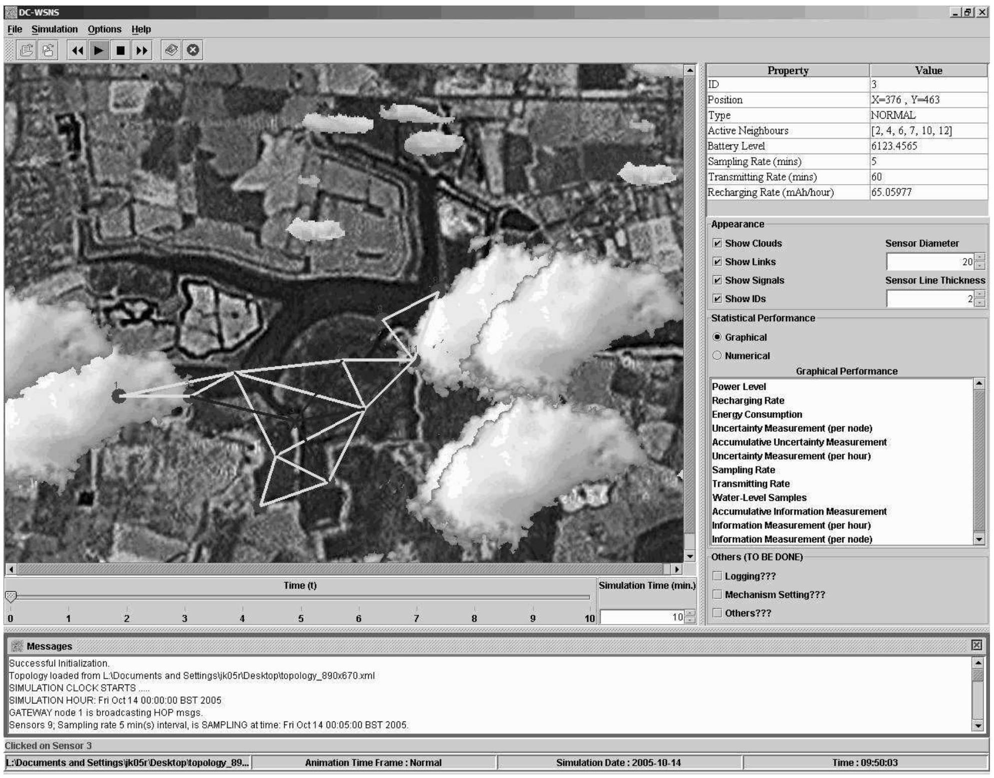
```
Mouse Position : 363 , 454
L:\Documents and Settings\jk05r\Desktop\topology_89... | Animation Time Frame : Normal | Simulation Date : 2005-10-14 | Time : 05:30:47
```

FIGURE 3.9: FLOODNET's nodes (represented by the filled dots) transmit their collected samples back to the base station node 1 (most left hand side node, using multi-hop routing). On the top and right of the screen are the toolboxes and controls correspondingly. Whilst on the left is the graphical display of the network. The message transmissions are represented by the circles that surround the nodes.

power consumptions of each node). At this stage, however, they can't be varied at run time.

In more detail, individual nodes in the network can be identified by selecting a particular node on the graphical network display (as shown in figures 3.9 and 3.10). Once selected, information such as the node's location (i.e. the relative location on the display panel of the simulator), type (whether it is an ordinary node or a base station), residual energy level, energy recharging rate, and sampling rate, can be viewed on the simulator's right panel. The simulation toolbox available on the toolbar is used to drive and simulate the network. This allows us to step through the simulation in single time steps (which is the fundamental period of time that is implemented in DC-WSNS and that can be adjusted) or to run the simulation until a particular time step. At each time step, all the nodes in the network are inspected to ascertain whether or not they are scheduled to sample or transmit data.

Additionally, during a simulation run, DC-WSNS records a range of real-time temporal network statistics. These include the remaining energy level, the battery recharging rate, the information value measurement, and the water level sampled data of each node. Each of these sets of information can be shown in a graph- or text-based format.



Property	Value
ID	3
Position	X=376 , Y=463
Type	NORMAL
Active Neighbours	[2, 4, 6, 7, 10, 12]
Battery Level	6123.4565
Sampling Rate (mins)	5
Transmitting Rate (mins)	60
Recharging Rate (mAh/hour)	65.05977

[✓ Show Clouds] [✓ Show Links] [✓ Show Signals] [✓ Show IDs]
Sensor Diameter: 20
Sensor Line Thickness: 2
[✓ Graphical] Numerical
Simulation Time (min.): 10

```
Successful Initialization.
Topology loaded from L:\Documents and Settings\jk05r\Desktop\topology_890x670.xml
SIMULATION CLOCK STARTS .....
SIMULATION HOUR: Fri Oct 14 00:00:00 BST 2005
GATEWAY node 1 is broadcasting HOP msgs.
Sensors 9; Sampling rate 5 min(s) interval, is SAMPLING at time: Fri Oct 14 00:05:00 BST 2005.
```

Clicked on Sensor 3

L:\Documents and Settings\jk05r\Desktop\topology_89... | Animation Time Frame : Normal | Simulation Date : 2005-10-14 | Time : 09:50:03

Figure 3.10: FLOODNET's nodes sample from the environment.

Besides, the simulation's workflow is logged and displayed inside the simulator's internal frame at the bottom of the screen.

Having outlined the basic features of DC-WSNS, we now consider the domain models upon which DC-WSNS is built. These models include the wireless communication, the battery, and the energy harvesting capabilities (specifically a solar panel model in combination with a cloud cover model). A network stack model (including the routing table of each node) is adapted from those in the ARA simulator (Zhou and De Roure, 2007). With all these models, DC-WSNS provides a platform in which objective observations can be made at any time.

3.5.1.1 The Wireless Communication Model

DC-WSNS is not currently designed to accurately model the wireless communication channel. Rather, we assume that a node has a single transmission level such that only neighbourhood nodes within its transmission range can hear, receive, and process its broadcasted messages. In DC-WSNS, a message is either a sampled data collected from the environment or a "HELLO" typed message packet originated by the base station for the node initialization process. Messages have a fixed size and, hence, require the same amount of time and energy to transmit.

We further assume that every propagated message is received by the receiving node without any failures and that there is no difference between overhearing and receiving in term of power consumption. Moreover, the nodes are pre-programmed to ignore and drop packets that are not destined for them (for the purpose of energy savings). Therefore, the communication model implemented is reasonably primitive[9].

3.5.1.2 The Battery Model

In our case, the nodes in the network deplete their energy resources at different rates (as the nodes might have different sampling rates at any time slots). The total energy consumed, $Bc_i(k)$, per time slot k for node i is given by the equation:

$$Bc_i(k) = a_i^k \cdot e_i^{\mathrm{s}} \qquad \text{where } a_i^k \in C_i \text{ and } i \in I \tag{3.24}$$

where a_i^k is the sampling action (or schedule) of node i at time slot k. We remark that as described in section 3.2, we do not include the constant transmitting and receiving variables into the equation since each node transmits its recorded readings in every two hour period to the base station using a non-adaptive multi-hop routing method.

Moreover, DC-WSNS reserves a fraction β_i (labelled as the cut-off threshold) of the energy supply (i.e. the battery) of each node $i \in I$. This reservation is normally applied to preserve the longevity of the battery's lifetime and, here, the cut-off threshold, β_i, is expressed as a fraction of the maximum energy supply, $Bmax_i$ (i.e. the full battery capacity). If a node's residual energy drops below this threshold, its radio communications (transmitting or receiving) and sensing capabilities cease. The energy budget available to each node at the outset is thus given by $Bmax_i \cdot (1 - \beta_i)$. Whilst the energy remaining at time slot k for node i, $B_i(k)$, is given as:

$$B_i(k) = Bmax_i \cdot (1 - \beta_i) - \sum_{j=1}^{k-1} Bc_i(j) \tag{3.25}$$

We note that this model is identical to the *linear model* proposed in (Park et al., 2001) where the battery is treated as a linear storage of current regardless of what the discharge rate is. The remaining battery capacity after operation duration of time t^d is thus expressed by the following equation:

$$B = B' - \int_{t^0}^{t^0+t^d} I(t)\ \partial t \tag{3.26}$$

[9]See section 4.5.1 for a more comprehensive model implemented for the inter-related adaptive sampling, transmitting, forwarding, and routing problem. The development of an algorithm for WSNs is typically targeted at analysing specific network's performance and, thus, realistic models should be used in the areas that significantly affect this particular criteria. In our case, the information-based algorithms require realistic energy and information models in order to ensure close correlation with practical results.

where B' is the previous capacity and $I(t)$ is the instantaneous current consumed by the node at time t. The linear model assumes that $I(t)$ will stay the same for the duration t^d. With this assumption, equation 3.26 simply becomes:

$$\begin{aligned} B = B' - \int_{t_0}^{t_0+t_d} I(t)\ \partial t &= B' - I \cdot t|_{t^0}^{t^0+t^d} \\ &= B' - I \cdot t^d \end{aligned} \tag{3.27}$$

Being the most simplistic model, our linear model falls short of portraying the behavior of a real life battery with characteristics such as *discharge rate dependent capacity*[10] and *relaxation effect*[11] (Park et al., 2001; Pop et al., 2005). However, we believe that our results presented in the next section would not be significantly affected by these types of battery model.

3.5.1.3 The Battery Recharging Model

While the battery model we have discussed so far is reasonably standard, it is not adequate for FLOODNET nodes because they have solar panels as one of their energy producing components. Thus a model of battery charging needs to be incorporated in addition to that of the static battery. Specifically, this involve an energy recharging model. For a such solar powered system found in FLOODNET nodes, a key issue is that of the amount of sunlight available; this is given by the time of day (i.e. there is no sunlight at night) and the cloud cover (i.e. if it is very cloudy then the panels will receive little sunlight). In modelling the clouds as realistically as possible, we consider their shape, their size, their thickness, their speed, and their movements in variable directions (recall figures 3.9 and 3.10).

With the cloud model, one form of energy recharging using a solar panel is to assume that if the solar panel of node i does not lie under a cloud, it will gain a preset energy increase defined as Bh_i each day. However, this energy will be reduced depending on the number of layers or the thickness of the cloud above the solar panel (represented by δ_i). The thicker or the more layers the cloud has (i.e. the bigger δ_i is), the more of the sun's energy is trapped inside, hence, the less energy is regained.

During night time (defined from time ns to time nf), due to the absence of the sunlight, the node's energy supply recharges at a small rate of $Bn_i(t)$ at time t due to stored energy in the solar cells. This rate is chosen arbitrarily and remains constant during

[10]A dependent capacity model considers the effect of battery discharge rate on the maximum battery capacity.

[11]A relaxation model considers the effect of high discharge rates that causes the battery to reach its end of life even though active materials inside are still available. This model is currently the most comprehensive model but there exists considerable difficulty in implementing it since the relaxation effect involves many electrochemical and physical properties of the battery.

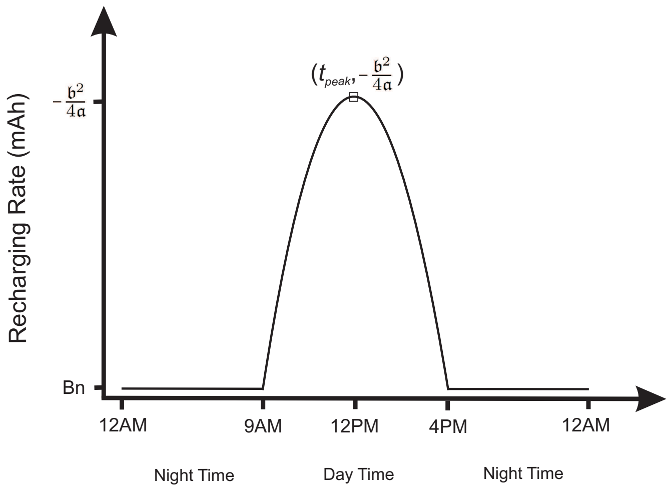


FIGURE 3.11: Recharging rate model (during winter season) where $nf = ds$ and $df = ns$.

the night time. Whilst during day time (defined from time ds to time df), the energy recharging rate is modelled as a quadratic function which peaks (i.e. recharges the most) at midday (see figure 3.11). This approach is based on the model used in (Kansal et al., 2004). The total energy recharging during day time, Bd_i, for node i can be found by deducting the total energy recharging during night time from that of each day:

$$\begin{aligned} Bd_i &= Bh_i - \int_{ns}^{nf} Bn_i(t)\ \partial t \\ &= Bh_i - Bn_i \cdot t|_{ns}^{nf} \end{aligned} \tag{3.28}$$

$$\begin{aligned} Bd_i &= \int_{ds}^{df} (\mathfrak{a}_i t^2 + \mathfrak{b}_i t + \mathfrak{c}_i)\ \partial t \\ &= \left[\frac{1}{3}\mathfrak{a}_i t^3 + \frac{1}{2}\mathfrak{b}_i t^2\right]_{ds}^{df} \qquad \text{where } \mathfrak{c}_i = 0 \\ &= \frac{1}{3}\mathfrak{a}_i(df^3 - ds^3) + \frac{1}{2}\mathfrak{b}_i(df^2 - ds^2) \end{aligned} \tag{3.29}$$

where we assume $\mathfrak{c}_i = 0$ because $Bn_i(t)$ is negligible compared to Bd_i. Now, because t_{peak} is exactly at the half way point of the quadratic curve:

$$t_{peak} = -\frac{\mathfrak{b}_i}{2\mathfrak{a}_i} \qquad \text{and} \qquad t_{peak} = \frac{(df - ds)}{2}$$

$$\mathfrak{b}_i = -\mathfrak{a}_i(df - ds) \tag{3.30}$$

The combination of equations 3.28, 3.29, and 3.30 is further derived as:

$$Bh_i - Bn_i \cdot t|_{ns}^{nf} = \frac{1}{3}\mathfrak{a}_i(df^3 - ds^3) - \frac{1}{2}\mathfrak{a}_i(df^2 - ds^2)(df - ds) \tag{3.31}$$

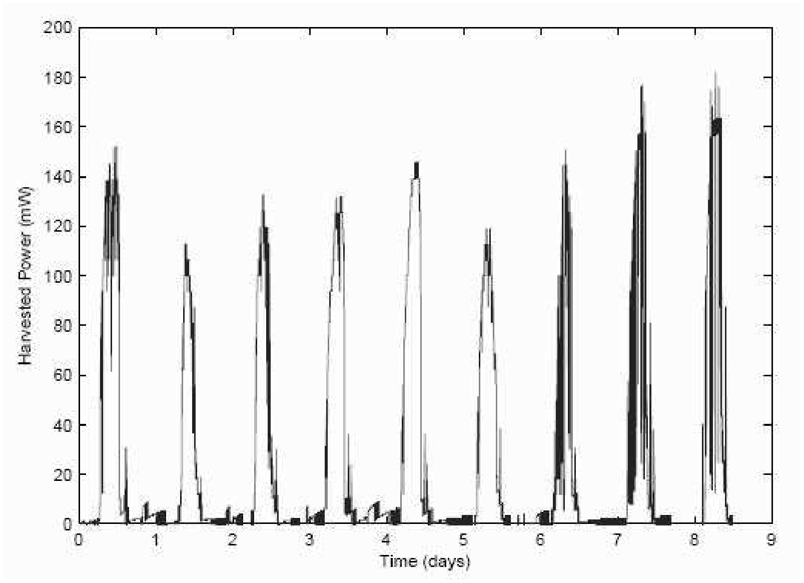


FIGURE 3.12: Heliomote solar energy-based charging power recorded for nine days (taken from (Kansal et al., 2007)).

Given equation 3.31, we are able to obtain the specific $\mathfrak{a}_i$ and $\mathfrak{b}_i$ values for node i. In turn, we are now able to calculate the energy recharging rate, $Bp_i(k)$, during a particular time slot k for node i:

$$Bp_i(k) = \begin{cases} Bn_i \cdot t|_{\mathfrak{t}(k)}^{\mathfrak{t}'(k)} & \text{if } ns \leq t \leq nf \\ \left[\frac{1}{3}\mathfrak{a}_i t^3 + \frac{1}{2}\mathfrak{b}_i t^2\right]_{\mathfrak{t}(k)}^{\mathfrak{t}'(k)} & \text{if } ds \leq t \leq df \end{cases} \tag{3.32}$$

where $\mathfrak{t}(k)$ and $\mathfrak{t}'(k)$ are the beginning and end of time slot k respectively.

From equations 3.25 and 3.32, the energy remaining, $B_i(k)$, at time slot k for node i becomes:

$$\begin{aligned} B_i(k) &= Bmax_i \cdot (1 - \beta_i) - \sum_{j=1}^{k-1} Bc_i(j) + \sum_{j=1}^{k-1} [Bp_i(j) \cdot (1 - \delta_i(j))] \\ &= Bmax_i \cdot (1 - \beta_i) - \sum_{j=1}^{k-1} [Bc_i(j) - Bp_i(j) \cdot (1 - \delta_i(j)] \end{aligned} \tag{3.33}$$

where the energy remaining never exceeds its maximum:

$$B_i(k) \leq Bmax_i \cdot (1 - \beta_i) \tag{3.34}$$

We believe this energy recharging model is adequate enough for our current purposes. This view is based on an experiment in which a test Heliomote was installed with a solar panel and was placed in a location where sunlight is available for part of the day (Kansal et al., 2007). The power output from the solar cell was recorded for a nine day period that vary among cloudy, hazy, and sunny. Figure 3.12 shows the results of this

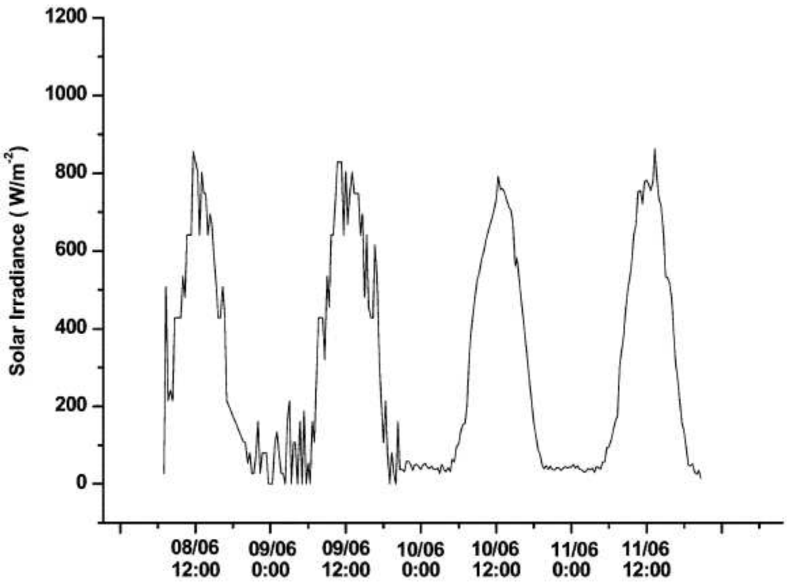


Figure 3.13: The solar irradiation of a wireless sensor node with an intelligent hybrid power source system, plotted against time (taken from (Li et al., 2008)).

experiment. Note that it shows a similar global pattern to that of our energy recharging model. The fluctuations are due to cloud covers and possibly energy leakage. We also note that the development of wireless sensor nodes with a hybrid power system to realize a self-sustaining WSN has also produced a similar solar irradiation plot to that of our recharging model (see figure 3.13).

3.5.2 Network and Parameters Setup

In our experiments, we use a simulation of the FLOODNET network, driven by real data for batteries, tide readings, and cloud cover (used to model solar energy harvesting). The experiments are run using FLOODNET's actual topology with a fixed number of nodes (twelve) at fixed locations (i.e. the nodes are immobile). The sampled data model (worth approximately eight days of measured data starting from Oct 14^{th} 2005 $00.00AM$) for each node was fixed for each instance of the experiments. All the cloud parameters including the cloud coverage, thickness, and wind speed are initialized with realistic data (at FLOODNET's site) available in METAR[12] format. These are all done in order to reproduce the FLOODNET scenario as realistically as possible. The remaining battery energy of each node and its recharging rate are set to be low so that it cannot continuously sample at its maximum rate. Given these constraints, nodes must therefore allocate resources and schedule themselves to determine how often and when to sample efficiently in order to maximise their collected information value.

In a simulation run, nodes can fail due to their battery depletion, but they cannot be added or removed. For the sake of simplicity and in order to exploit all the possible

[12]`http://weather.noaa.gov/weather/metar.shtml`, checked on 02/02/2009.

Table 3.2: Parameters used for the simulations.

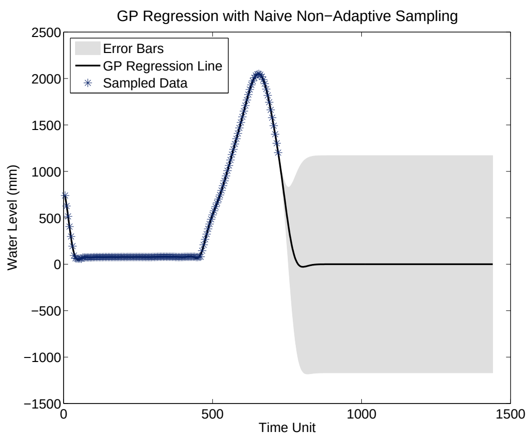
Parameter	**Value**
Number of nodes (n)	12
Number of time slots (w)	24
Maximum energy capacity ($Bmax_i, \forall i \in I$)	51840 joules
Energy supply cut-off threshold ($\beta_i, \forall i \in I$)	0.5
Energy to sample ($e_i^s, \forall i \in I$)	0.14 joules/sample
Transmission rate	1 transmission/2 hours
Number of different sampling rates ($s_i, \forall i \in I$)	4
Set of possible sampling rates ($C_i, \forall i \in I$)	{1, 3, 6, 12} samples
Maximum sampling rate ($c_i^{s_i}, \forall i \in I$)	12 samples/time slot
Minimum sampling rate ($c_i^1, \forall i \in I$)	1 sample/time slot
Energy recharging rate at night time ($Bn_i, \forall i \in I$)	18 mJ/hour
Night start time (ns, df) during summer	19:00 pm
Day start time (ds, nf) during summer	07:00 am
Night start time (ns, df) during winter	16:00 pm
Day start time (ds, nf) during winter	08:00 am
Energy recharging rate in month $1 \leq month \leq 12$ ($Bh_i, \forall i \in I$)	{8.4, 13.2, 16.8, 21.6, 25.2, 22.8, 24.0, 22.8, 19.2, 14.4, 7.2, 6.0} J/day
Average temperature in month $1 \leq month \leq 12$	{1.6, 2.3, 5.4, 7.8, 11.7, 14.6, 17.5, 17.1, 14.1, 10.0, 5.1, 2.8} Celsius

changes in the system, at this point of time, we only consider four different actions ($s_i = 4$, $\forall i \in I$) describing each node's sampling rate. Thus, it can either sample one, three, six, or twelve times per hour (i.e. $C_i = \{1, 3, 6, 12\}$, $\forall i \in I$). The other simulator parameters are initialized as described in table 3.2.

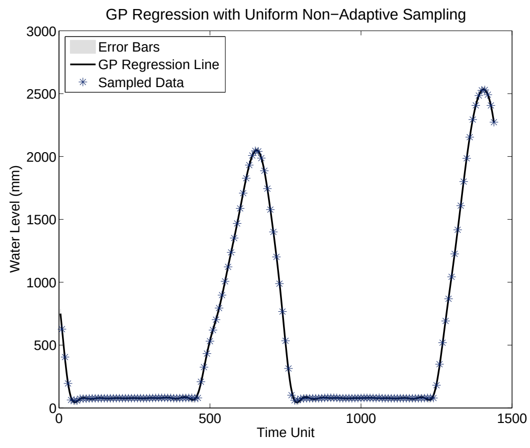
3.5.3 Benchmark Algorithms

In our experiments, the benchmark algorithms include:

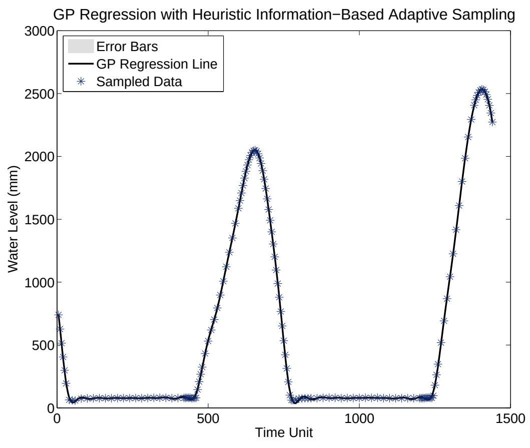
- **A Naïve Non-Adaptive Sampling Algorithm:** This dictates that each node $i \in I$ should sample at its maximum rate, $c_i^{s_i}$, whenever there is enough battery energy to do so. The node's sampling behaviour is therefore non-adaptive and can be described as:
$$a_i^k = c_i^{s_i} \quad \forall i \in I, \forall k \in H \tag{3.35}$$
- **A Uniform Non-Adaptive Sampling Algorithm:** This dictates that each node $i \in I$ in the network should simply choose to divide the total number of samples it can perform in a day, N_i, where $N_i = \frac{B_i}{e_i^s}$, equally into its time slots, such that:
$$a_i^k = \arg\max_u u \quad \text{where } u \in \left\{ c_i^j \in C_i | c_i^j \leq \frac{N_i}{w} \right\} \quad \forall i \in I, \forall k \in H \tag{3.36}$$
- **A Utility-based Sensing And Communication (USAC) Algorithm:** This is a state-of-the-art algorithm that lets each node adjust its sampling rate depending on the rate of change of its observations (recall section 2.4 for more details).



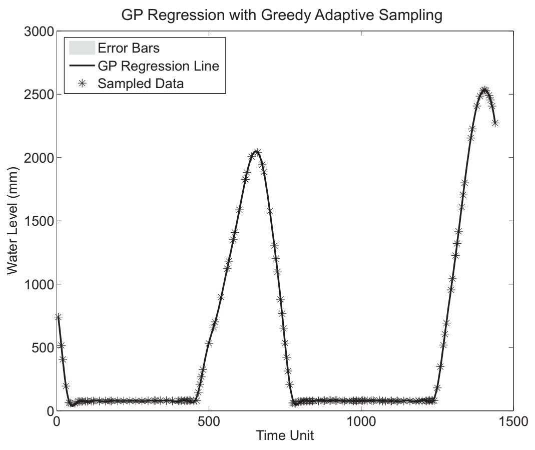
(a) GP Regression with Naïve Non-Adaptive Sampling Algorithm collecting Fisher information value of $FI = 1.804 \cdot 10^{-3}$.



(b) GP Regression with Uniform Non-Adaptive Sampling Algorithm collecting Fisher information value of $FI = 3.278 \cdot 10^{-2}$.



(c) GP Regression with Heuristic Information-Based Adaptive Sampling Algorithm collecting Fisher information value of $FI = 7.396 \cdot 10^{-2}$.



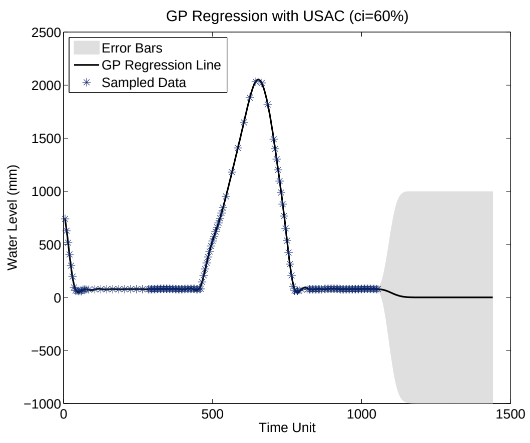
(d) GP Regression with Greedy Adaptive Sampling Algorithm collecting Fisher information value of $FI = 1.045 \cdot 10^{-1}$.

FIGURE 3.14: GP Regression evaluated on daily basis for FLOODNET node 1 collecting samples on Oct 15th 2005. One time unit represents a one minute interval.

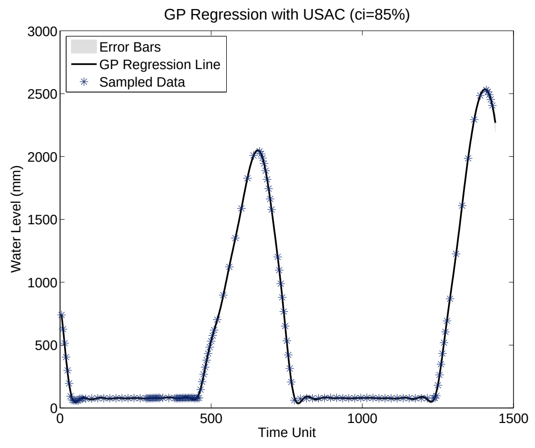
Specifically, the algorithm uses a linear regression method which is run to determine the next predicted data, $dat(t+1)$, with some bounded error (termed its *confidence interval*, ci). If the next observed data falls outside ci, the node sets its sampling rate to the maximum rate in order to incorporate this phase change. However, if data falls within the ci, it implies that the node is allowed to reduce its sampling rate for energy efficiency due to the presence of predictable information that has a low value. The USAC algorithm does not have a notion of time slot and therefore, each node has the flexibility to change its own sampling rate at any point of time[13].

Now, assuming that the sampling rate of node $i \in I$ at t point of time, $Sr_i(t)$, is

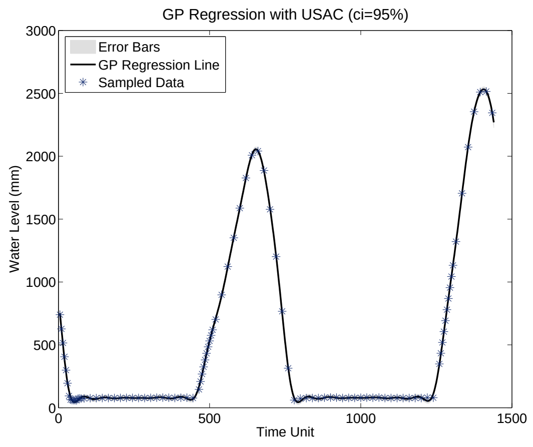
[13]All other algorithms, on the other hand, dictate that each node should only change its sampling rate at subsequent time slots.



(a) GP Regression with USAC (ci=60%) collecting Fisher information value of $FI = 4.058 \cdot 10^{-3}$.



(b) GP Regression with USAC (ci=85%) collecting Fisher information value of $FI = 6.142 \cdot 10^{-2}$.



(c) GP Regression with USAC (ci=95%) collecting Fisher information value of $FI = 5.559 \cdot 10^{-2}$.

FIGURE 3.15: GP Regression with USAC using different values of ci evaluated on daily basis for FLOODNET node 1 collecting samples on Oct 15th 2005. One time unit represents a one minute interval.

equal to c_i^j, where $c_i^j \in C_i$, then its sampling rate at $t+1$ is defined as:

$$Sr_i(t+1) = \begin{cases} c_i^{j-1} & \text{if } Lo(ci) \leq dat(t+1) \leq Up(ci) \\ c_i^{s_i} & \text{otherwise} \end{cases} \tag{3.37}$$

where $Lo(ci)$ and $Up(ci)$ are the lower and upper bound of ci respectively. As the setting of ci is central to USAC's operation, and because no guidelines are given about what values to use, here we use the following range of values: $60\%, 85\%$, and 95%. This is, we believe, sufficient to fully examine USAC's performance in this domain.

- **Unconstrained Sampling:** This ignores the constrained energy of the node, and allows the node to sample its maximum rate for the entire trial period. This represents an absolute upper bound on the value of information that can be collected, but clearly cannot actually be implemented in practice since the nodes will deplete their batteries before the end of the trial period.

3.5.4 Results

We now present the results of the simulation process described above. Figures 3.14 and 3.15 show the comparison of Fisher information values that are evaluated (on a daily basis using the GP regression technique as per equation 3.5) with different sets of FLOODNET node 1 observations collected using the different algorithms. It also shows that the node obtains the lowest uncertainty in its set of readings on that day and, hence, the highest information value (FI), when it collects samples using the greedy adaptive sampling algorithm. With the heuristic information-based adaptive sampling algorithm, on the other hand, it collects slightly less information value. However, this value is significantly higher compared to those collected using the two non-adaptive and USAC benchmarks.

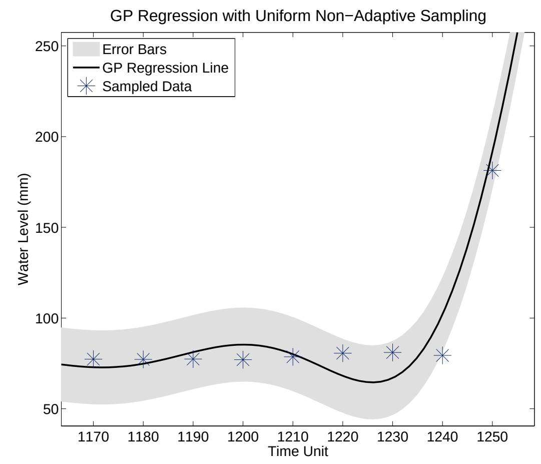
The actual $\widehat{\sigma}^2(\mathfrak{x}^j)$ (defined in equation 3.4) in figures 3.14 and 3.15 are too small to be visible on such a scale (for enlarged versions and to show more clearly how the heuristic adaptive sampling algorithms outperform the non-adaptive and USAC algorithms, see figures 3.16 and 3.17[14]). Moreover, we also compare the computational time performance of the various algorithms in order to provide the other side of the optimality and computational time trade-off.

3.5.4.1 Information Value Analysis

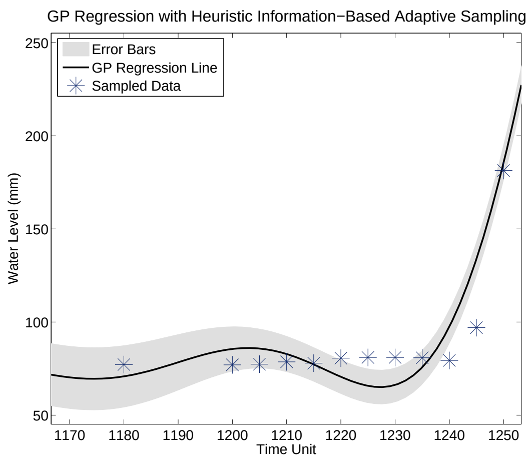
As can be seen in figure 3.18(a), the heuristic information-based adaptive sampling algorithm performs well. Compared to the naïve and uniform non-adaptive approaches respectively, this algorithm consistently increases the total Fisher information collected by about 83% and 27% per day over the trial period. The plot clearly shows the superiority of the heuristic information-based adaptive sampling algorithm and that the information value of the data collected found by it is approximately 75% of the greedy's.

Furthermore, in comparison with case of nodes that can sample unconstrained by power requirements (i.e. they can sample at the maximum rate throughout the trial period) our heuristic information-based adaptive sampling algorithm is approximately 66% of this upper bound. This upper bound corresponds to the peaks in figure 3.18 and, thus, we do not show it as an additional line in this plot.

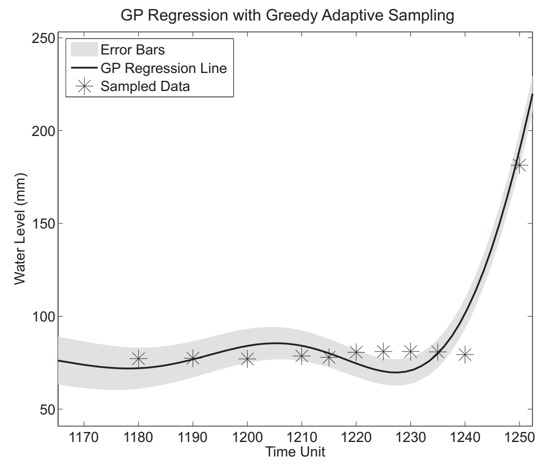
[14]In these figures, the GP regression using the naïve non-adaptive and USAC ($ci = 60\%$) sampling algorithms are not shown because the plots would simply be covered by the error bars as can be seen from figures 3.14(a) and 3.15(a) respectively.



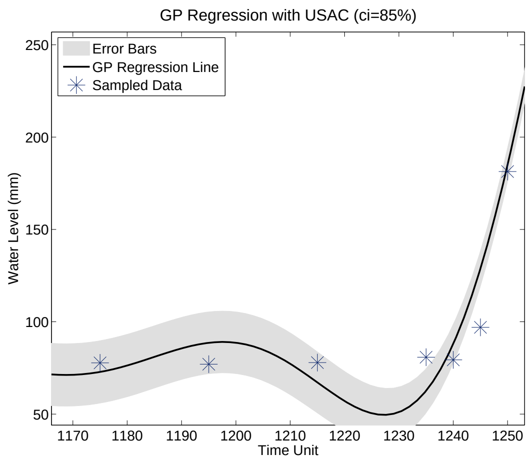
(a) GP Regression with Uniform Non-Adaptive Sampling Algorithm.



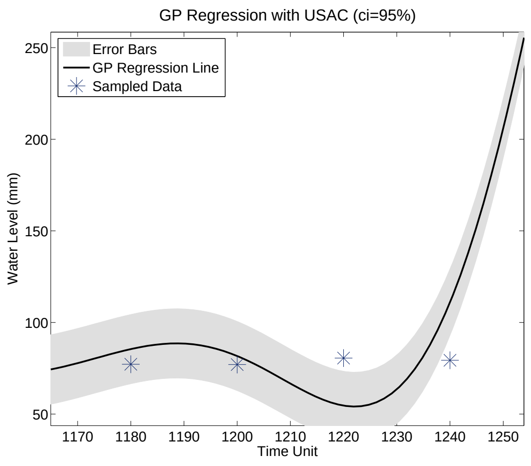
(b) GP Regression with Heuristic Information-Based Adaptive Sampling Algorithm.



(c) GP Regression with Greedy Adaptive Sampling Algorithm.

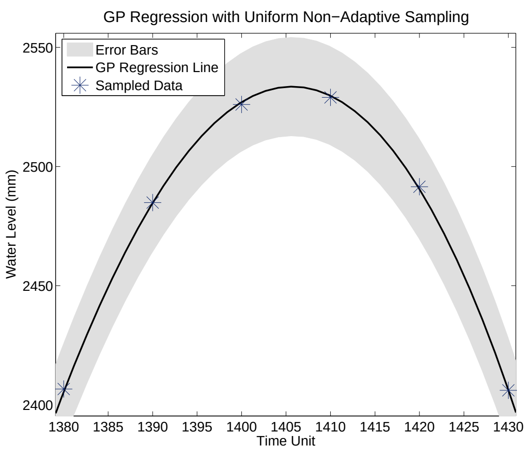


(d) GP Regression with USAC (*ci*=85%).

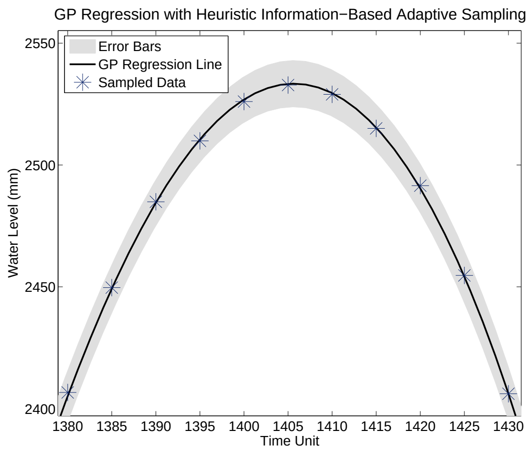


(e) GP Regression with USAC (*ci*=95%).

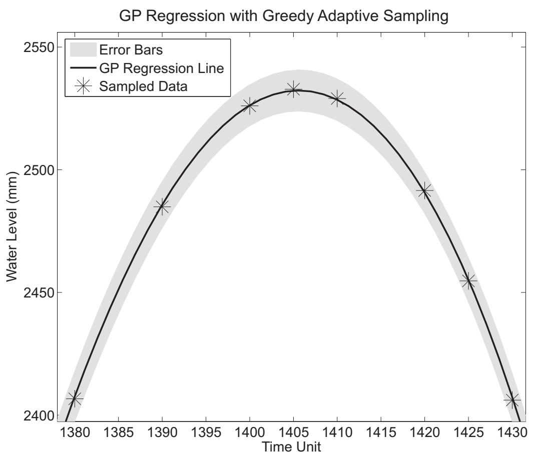
FIGURE 3.16: GP Regression on sampled data taken between time unit 1170 and 1250 from FLOODNET node 1 operating on Oct 15th 2005. One time unit represents a one minute interval.



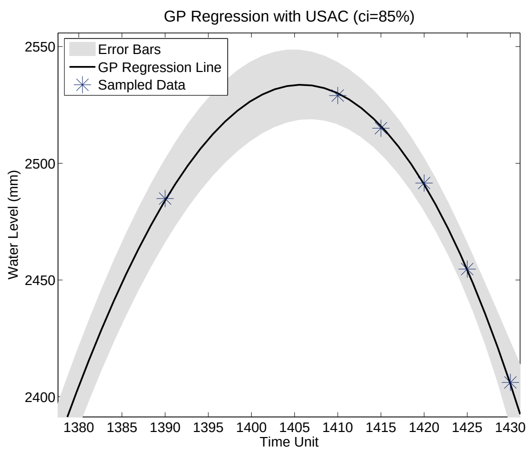
(a) GP Regression with Uniform Non-Adaptive Sampling Algorithm.



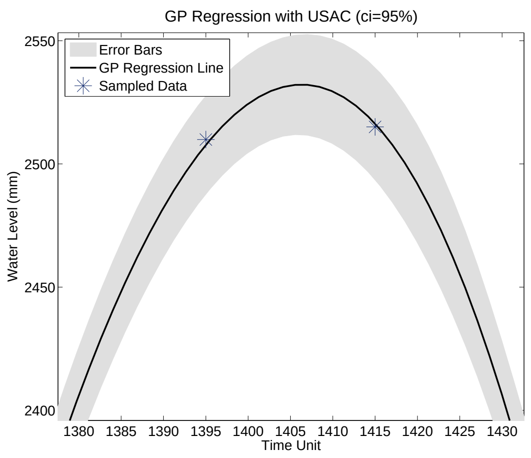
(b) GP Regression with Heuristic Information-Based Adaptive Sampling Algorithm.



(c) GP Regression with Greedy Adaptive Sampling Algorithm.

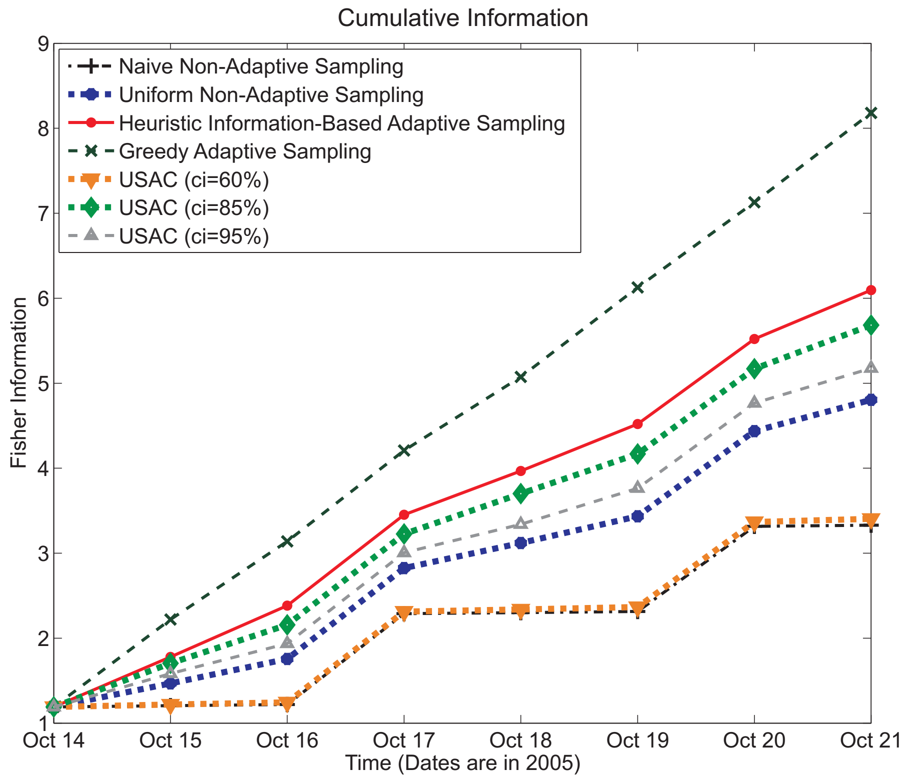


(d) GP Regression with USAC (*ci*=85%).

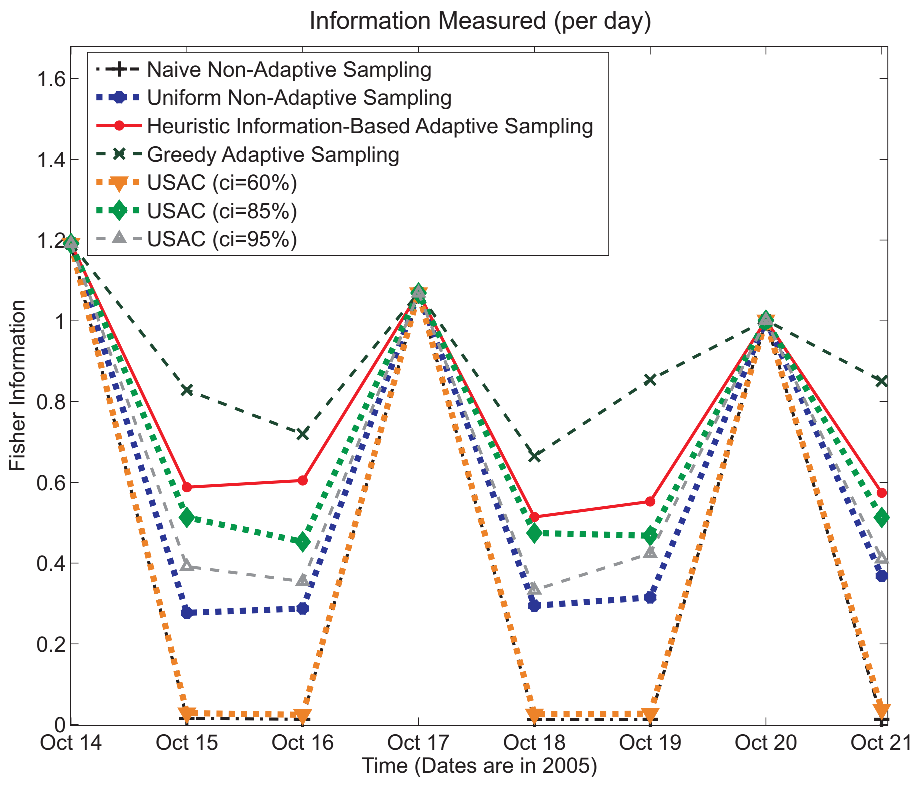


(e) GP Regression with USAC (*ci*=95%).

FIGURE 3.17: GP Regression on sampled data taken between time unit 1380 and 1430 from FLOODNET node 1 operating on Oct 15th 2005. One time unit represents a one minute interval.



(a) Cumulative information gathered over an 8-day period plotted against time.



(b) Information gathered daily over an 8-day period plotted against time.

Figure 3.18: Information measurement graph performances.

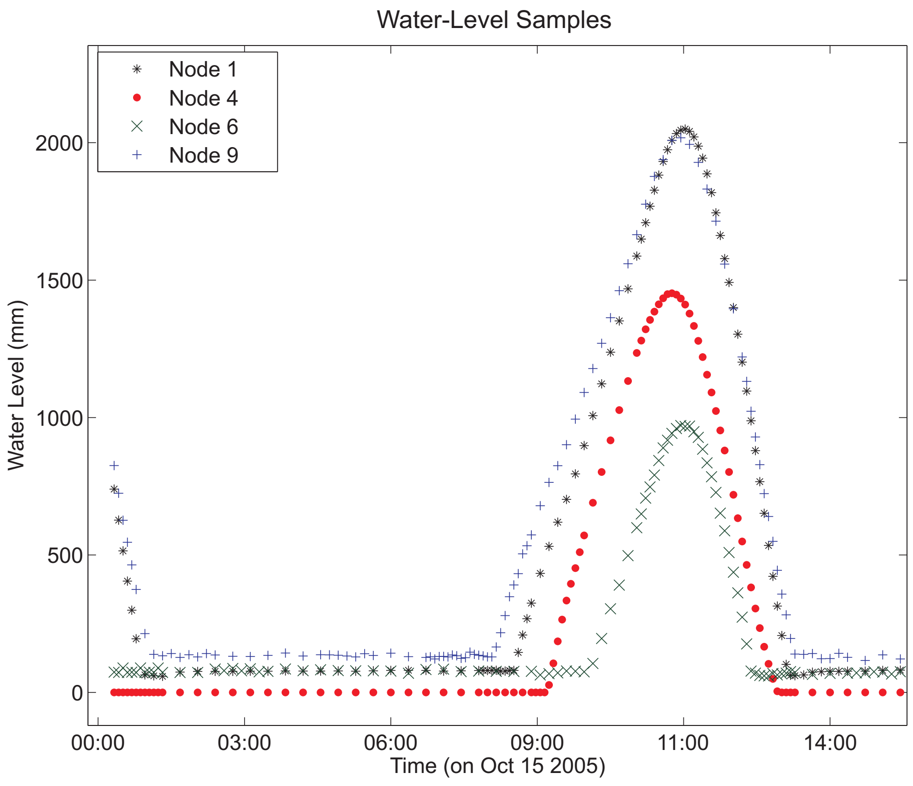


FIGURE 3.19: Water samples gathered on the second day of the simulation using the heuristic information-based adaptive sampling algorithm. Graph only displays some selected nodes for better visibility.

The heuristic information-based adaptive sampling algorithm also outperforms USAC for all values of *ci*. The main reason is due to the absence of a forward planner in USAC. In more detail, figure 3.15(a) shows how USAC ($ci = 60\%$) behaves poorly with performance similar to that of the naïve one. With this non-carefully chosen *ci* value, a small change in environmental readings will trigger each node to change its sampling rate to its maximum (of which energy is never reserved for possible future usage). As it does not have the power to continuously sample at this maximum rate, it often runs out of energy during a day and so collects no information for a long period of time. Figures 3.15(c), 3.16(e), and 3.17(e), on the other hand, show how USAC ($ci = 95\%$) performs in a similar fashion to the uniform non-adaptive one. In this setting, the next predicted data is highly likely to fall within the bounded error, *ci*, and hence, the algorithm dictates that each node should decrease its sampling rate due to the presence of low value predictable readings. Here, the harvested and renewable energy is not allocated effectively because the algorithm does not maintain the nodes in energy neutral operation mode (i.e. balance the amount of energy harvesting against that of energy consumption). USAC's optimal *ci* value for FLOODNET data is found to be 85% in which the algorithm collects information with value 8% lower than that collected by our heuristic information-based adaptive sampling algorithm over the trial period.

Additionally, figure 3.18(b) shows more clearly how the adaptive sampling algorithms

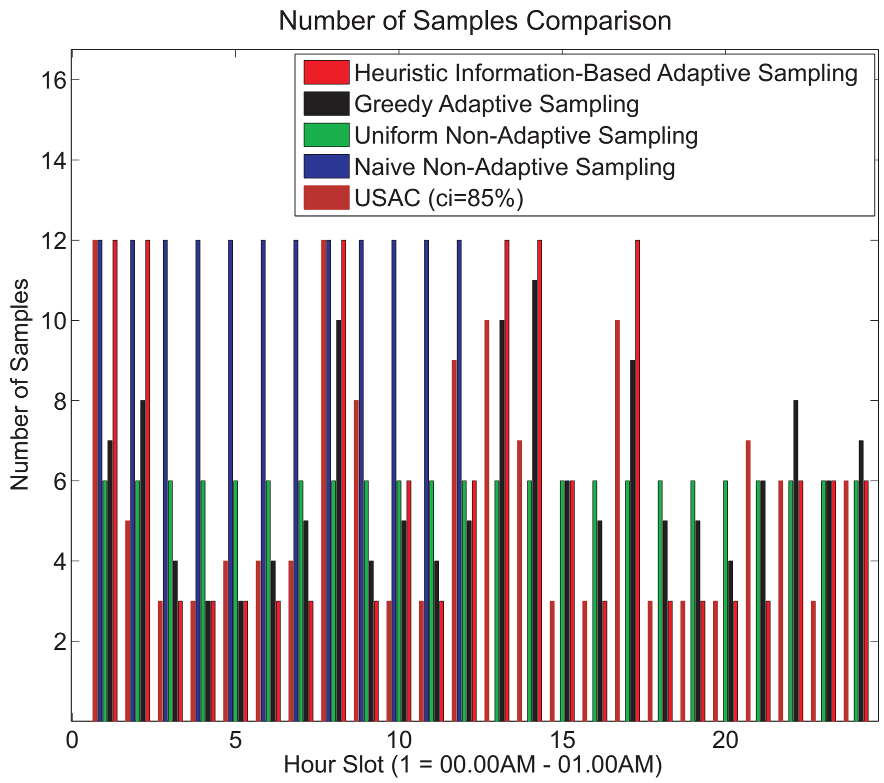


FIGURE 3.20: Number of samples taken in each hour slot from FLOODNET node 1 operating on Oct 15th 2005. The heuristic information-based adaptive sampling algorithm behaves in the same pattern as the greedy one. Both generally samples more often during dynamic events while sacrificing some energy to sample less during static ones.

achieve this performance. After leaving the schedule updating mode (i.e. the second day of a simulation, as can be seen in figures 3.19 and 3.20), a node is able to perform adaptive sampling by conserving its battery energy in order to take more samples during the most dynamic events, while taking fewer samples during the static ones. In our case, the dynamic events of a tide occur at the time it comes in (specifically when the node rises off mud, between 07.00 and 09.00 in figure 3.19), reaches the peak (between 10.00 and 11.00), and goes out (between 12.00 and 14.00). During these events, nodes normally set their sampling rates to a maximum value (i.e. in our case, at five minute intervals). As a result, from the second day onward, figure 3.18(b) shows a gain in information value collected (particularly during the dynamic events), except when the nodes are in updating mode (on Oct 17th and 20th). In this mode, all nodes sample at their maximum rates (as discussed in section 3.4.3.2), therefore on those dates, the information valuations of the five approaches are the same.

3.5.4.2 Run Time Performance

The optimal adaptive sampling algorithm works only for very small problems as it very rapidly becomes infeasible for even small- to medium-sized ones. For instance, consider

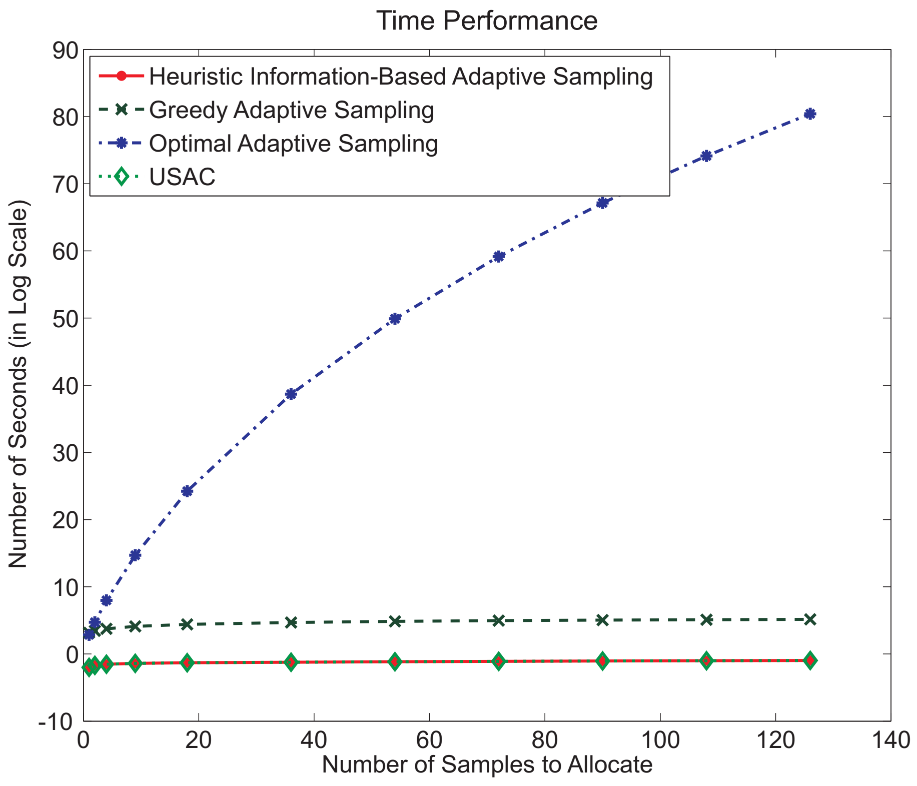


FIGURE 3.21: Computational time performance.

an adaptive sampling problem in which a node has sufficient battery capacity to sample only 3 times from the possible 288 sampling points in a day. In this scenario, there are $10^5(C_3^{288})$ solutions to enumerate, which is just about possible to do with a modern computer in a finite amount of time. However, for a slightly larger problem, where a node has the flexibility to sample 36 times from the possible 288 sampling points in a day, there are now $10^{37}(C_{36}^{288})$ solutions to enumerate, which is intractable in a reasonable amount of time, even for a very fast computer. Assuming a 3GHz desktop PC on which the GP regression technique takes approximately 5 seconds, we estimate this would take 10^{30} years to compute. In more detail, figure 3.21 shows a comparison of the computation time of all the algorithms that we consider.

The greedy adaptive sampling algorithm significantly reduces the computation time, since it reduces the number of possible combinations of sampling points that must be compared. However, it is still too slow to be run on the nodes that have similar computational power to FLOODNET's. This is because there are still $\frac{1}{2} \cdot addSamp \cdot ((remSP - addSamp) + remSP))$ possible solutions to iterate, where as defined in section 3.4.2, $addSamp$ and $remSP$ are the number of additional samples to be allocated and the possible remaining sampling points respectively. For instance, for a problem in which a node is capable of allocating 36 samples from the possible 288 sampling points, there are still 9720 solutions to evaluate, and experiments indicate that this takes approximately 14 hours to compute on a standard 3Ghz desktop PC.

The heuristic information-based adaptive sampling algorithm, on the other hand, runs in real-time on the current configuration. This is due to the performance of the linear programming optimization technique and the simplified information valuation metric. Again, consider the 36 sample problem for example, this algorithm calculates the preferred solution in hundreds of milliseconds. Finally, due to the similar nature of linear regression method used for valuing information, USAC also executes in real-time.

3.6 Summary

In this chapter, we have focused on issues associated with energy management in general and information-based adaptive sampling in particular. We have developed a principled information metric based upon Fisher information and Gaussian process regression that allows the information content of a node's observations to be expressed (thus addressing Contribution 1), and given a set of node readings, we have shown how an optimal and a greedy adaptive sampling algorithm can be devised. They are, however, only tractable for very small problems and, thus, we have developed a more practical, heuristic information-based adaptive sampling algorithm with the ultimate aim of maximizing the information value of the data collected (Contribution 2). This approach is a better choice for larger sampling problems (beating the benchmarks and obtaining performance close to the optimal one, but with much lower time complexity). The empirical results show that all three decentralised control algorithms for information-based adaptive sampling are effective in balancing the trade-offs associated with wanting to gain as much information as possible by sampling as often as possible, with the constraints imposed on these activities by the limited power available (Contribution 3).

Although the effectiveness of these algorithms is evaluated within the FLOODNET domain, the challenges that are involved here are very similar to those that occur in the design of many other WSNs. Specifically, many WSNs are being deployed in the domain of environmental phenomena monitoring (see section 2.1), and data in these settings typically exhibits periodic features (as we have with the tides) due to the natural cycle of day and night. The GP regression algorithm learns this periodicity from the temporally data and, thus, can be applied directly. It is also expected to work well in domains in which readings are more noisy since the GP handles sensor noise well. The linear programming technique, together with the utility functions and constraints, can also be adapted to meet the design objectives of other WSNs in general. However, some WSN applications also require spatially dense node deployments to achieve satisfactory coverage (such as object tracking applications). This is because with high spatial densities of sensors in the deployed area, multiple nodes often record information about a single event and collect spatially correlated data. These spatially proximal node observations are highly correlated with the degree of correlation increasing typically with the diminishing inter-node separation. As our algorithms do not explicitly take spatial correlation into account, they will not work well in reducing the redundancy in each event information sensed by multiple nodes. Moreover, the manual setting of the calibration cycle (as discussed in section 3.4) might prevent our algorithms from performing well when applied in WSN applications whose data exhibits extremely irregular patterns. We will, however, work towards automating this parameter setting as part of our future work.

We have so far only looked at controlling WSNs by adapting the nodes' sampling actions. Moreover, we have argued that there are many realistic application areas where such an approach is sufficient. However, there are also clearly other environments where the problem model needs to be extended to incorporate the nodes' inter-related sampling, transmitting, forwarding, and routing actions in multi-hop WSNs. Thus, we address this case in the following chapter in which we develop adaptive sampling, transmitting, forwarding, and routing algorithms that will allow nodes to make principled trade-offs between using their energy reserves to take more samples and transmit them, or to relay data from another node to the base station.

Chapter 4

Decentralised Control of Adaptive Sensing, Forwarding, and Routing

So far, we have looked at information-based decentralised control of adaptive sampling in WSNs in which the nodes transmit their samples either directly to the base station or using a non-adaptive multi-hop routing method, with the system-wide goal of maximising the information value of the data collected, constrained by their limited energy resources. As discussed in chapter 2, however, FLOODNET nodes need to transmit their readings in an adaptive multi-hop fashion as most of them do not have a direct link with the base station and at any particular point of time, they will have different remaining energy budgets and total values of information collected so far.

We thus turn our attention to the inter-related adaptive sampling, transmitting, forwarding, and routing problem. Specifically, we here develop two novel optimal decentralised algorithms. The first one varies each sensor's sampling, transmission, and forwarding rates to ensure all nodes in a network focus their limited resources on maximising the information content of the data delivered to the base station while assuming that the route by which the data is forwarded is *fixed*, either because the underlying communication network is a tree or because an arbitrary choice of route has been made (Contribution 4). The other deals with *flexible* routing and dictates each node to make optimal decisions regarding both the integration of actions that each node should perform, and also the route by which this data should be forwarded to the base station (Contribution 5). We then empirically evaluate them and show that they represent a trade-off in optimality, communication cost, and processing time (Contribution 6).

To this end, in section 4.1, we consider an exemplar scenario where this problem typically occurs. In section 4.2, we then detail a specific type of WSN for which our proposed decentralised control mechanism is extremely well suited. In section 4.3, we state the formal model of our distributed constraint optimization problem (i.e. the inter-related adaptive sampling, transmitting, forwarding, and routing problem) by extending our system model from the previous chapter (in section 3.2). In section 4.4 we detail two novel algorithms for solving the problem. In both cases, we show how we find the optimal local allocation decisions. Our experimental evaluation of these algorithms are presented in section 4.5. The chapter is then concluded in section 4.6.

4.1 Inter-Related Information-Based Adaptive Sampling, Transmitting, Forwarding, and Routing

In a multi-hop cooperative WSN, the nodes need to coordinate in order to maximise the network's global objective. In general, they are typically not able to simply sample and transmit their own samples at their maximum frequency because doing so would deplete their energy and, hence, they would not be able to relay (or forward) their neighbours' data messages. Such uncoordinated behaviour would mean that very little data would end up at the base station because it is never forwarded on. Therefore, there exists a trade-off associated with wanting to gain as much information as possible by sampling and transmitting as often as possible, with the constraints of the limited resources available on these nodes.

To address this efficiently and effectively, our approach (more details of which are given in section 4.4) seeks to provide a distributed coordination mechanism for the cooperative nodes to potentially conserve battery energy by taking and transmitting fewer of their own samples, while allocating some of their energy to forward and route the data of others. By doing this, the system should maximise the total information that reaches the base station, subject to the constrained energy resources of each individual node within the network.

To illustrate this point, consider the scenario case expected in figure 4.1(a). Here, nodes i and 3 are out of the base station's wireless transmission range. Hence, they require the help of node 1 to forward their data. Nodes have their own *energy budgets* (as one of their private values which we will discuss in more detail in section 4.3) to perform sampling, transmitting, and forwarding activities of which each requires a different amount of energy. Now, if node i's or 3's sampled data is judged to be more important than that of node 1, then sacrificing a portion of node 1's energy to forward some or all nodes i's or 3's data to the base station will yield a higher information value at the base station.

Now, consider the slightly different case of figure 4.1(b). Here, node i has two options to route node 6's and its own data through; namely (i) node 1 and (ii) node 2. If node 2 has important data to be transferred to the base station, then node i is better off either by: (i) routing both node 6's and its own data through node 1 or (ii) routing the smallest in size data between the two through node 2 while the other is forwarded via node 1.

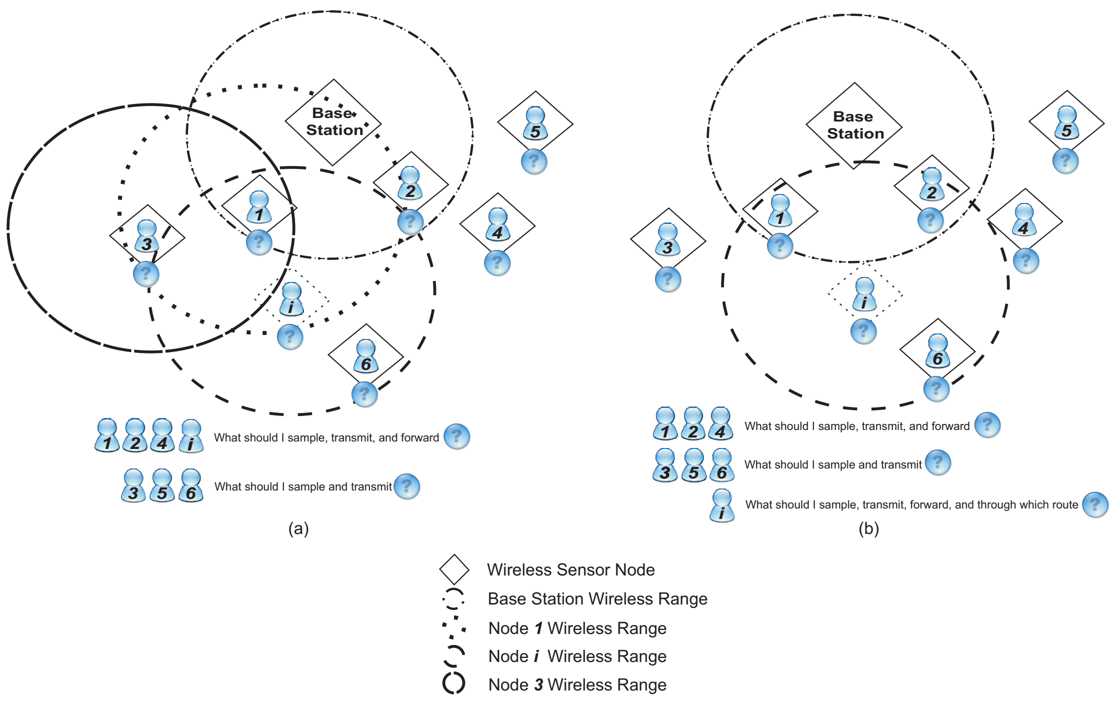

Base Station
1
2
3
4
5
6
i
What should I sample, transmit, and forward
What should I sample and transmit
(a)
What should I sample, transmit, forward, and through which route
(b)
Wireless Sensor Node
Base Station Wireless Range
Node 1 Wireless Range
Node i Wireless Range
Node 3 Wireless Range

Figure 4.1: Inter-related information-based adaptive sampling, transmitting, forwarding, and routing scenario in (a) a tree-structured network and (b) an arbitrary network that exhibits loops. Dotted node i could represent any subnetwork.

For both these distributed constraint optimization settings, a scalable and robust distributed coordination mechanism needs to be devised to provide the nodes with the flexibility of making their own efficient local decisions. Control must be distributed among the nodes, each of which cooperates to calculate the optimal decisions by exchanging coordination messages with neighbourhood nodes. Given that the nodes may have different energy constraints and communication costs, these local allocation decisions must include those that: (i) dictate each node's *sampling rate* over a particular time interval, (ii) adapt the node's *transmitting, receiving, and forwarding* behaviour, and (iii) select the best, feasible energy efficient *routing path* with the fewest number of hops towards the base station. The size of the exchanged coordination messages must be kept to minimum as there is also a resource cost associated with this activity.

In more detail, we note that these local decisions are inter-dependent as no node in the network is able to make such a decision individually. This is because the sampling, transmitting, forwarding, and routing choices of an individual node will affect those available to all of the other nodes in the network. This is may be directly, since they are on the path from this node to the base station and, thus, must devote energy to the forwarding of this node's data, rather than acquiring and transmitting their own. Or it may be indirectly through the propagation of these effects. Now, as explained in section 2.2, this message-exchange coordination mechanism inevitably involves an additional small communication cost in order to derive an optimal distributed resource allocation algorithm for networks in which the nodes possess private values. We thus motivate our new algorithms by considering WSN applications that deal with large data sets. Specifically, we chose *wireless visual sensor networks* (WVSN) as the nodes capture image (or visual) data of a large size and so the additional overhead of coordination is much less significant compared to the size and volume of the data sent (Zahariadis and Voliotis, 2007).

4.2 Wireless Visual Sensor Networks

A WVSN consists of a number of spatially distributed smart camera devices, each of which is capable of performing basic capturing, processing, and compression of visual data of a scene from a variety of viewpoints (i.e. only within a specific area) before relaying it to a base station in a multi-hop fashion to be jointly processed, analysed, or fused into some form more useful than the individual data (e.g. to reconstruct the entire surveillance area covered by the network). Each node has its own image sensor, image processing, storage, communication, and limited power units (see figure 4.2 for an example of a wireless smart camera mote).

Such networks are intended for distributed image acquisition (in a variety of applications such as object tracking (Simon et al., 2004), unattended area surveillance (Bramberger et al., 2006), and other security related applications) over large and possibly hostile environments and, as such, are required to operate for extended periods of time with minimal human intervention. The increased computational power of the nodes within a WVSN compared to those typically deployed within a conventional WSN, the large amounts of visual information that they collect, and the high energy cost of wirelessly communicating this information through the network, mean that efficient energy management is a key challenge in these networks.

To this end, recent work has explored decentralised coordination algorithms that enable the nodes to autonomously adapt and adjust their sampling and routing behaviours (recall the discussion in section 2.6). However, much of this work has specifically addressed WSNs in which the nodes are assumed to be low power, computationally constrained devices, and as such, has considered simple heuristic algorithms that allow the nodes to make local decisions to improve the overall performance of the network.

While such approaches have proved valuable in the context in which they were developed, when applied to WVSN they do not fully exploit the computational power available to the nodes. Furthermore, the large amounts of visual data that the nodes within the WVSN collect and communicate means that the communication resources available to the decentralised coordination mechanism are much greater than those of many conventional WSNs. When taken together, the move to WVSN means that there is now the

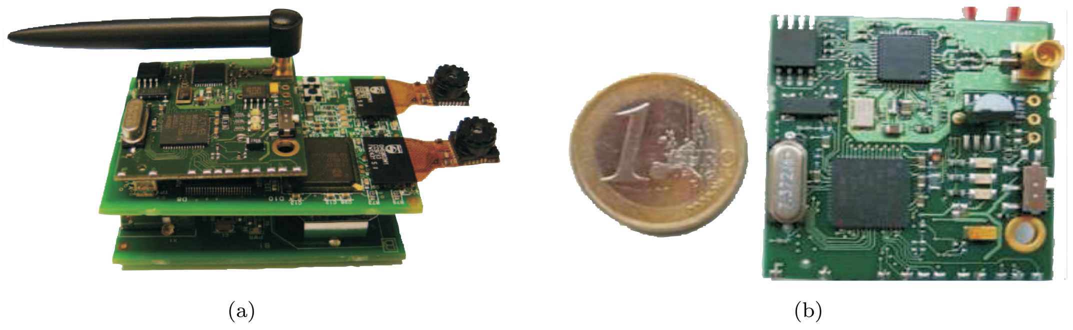


FIGURE 4.2: (a) Wireless smart camera mote (taken from (Kleihorst et al., 2006)) which is battery-powered and equipped with two VGA colour image sensors, SIMD (Single Instruction Multiple Data) image-analysis processor and 8051 micro-controller, a dual port RAM with 128KB memory, and a ZigBee IEEE 802.15.4 communication module, as shown in (b).

possibility of deploying algorithms that can optimally maximise the overall effectiveness of the network through distributed computation, rather than use local heuristics.

Thus, it is exactly this challenge that we address in this chapter and to this end, we present two optimal and decentralised algorithms for adaptive sampling, transmitting, forwarding, and routing to solve the *distributed constraint optimization* (DCOP or DisCOP) problem.

4.3 Problem Description

Here, we formalise the generic inter-related adaptive sampling, transmitting, forwarding, and routing problem that we face. To this end, let n be the number of visual sensor nodes within a WVSN system and the set of all nodes (or agents) be $I = \{1, \ldots, n\}$. Each node $i \in I$ can sample at s_i different rates over a period of time. Its set of possible sampling (or frame) rates is denoted by $C_i = \{c_i^1, \ldots, c_i^{s_i}\}$; the elements of which appear in increasing order such that $c_i^j < c_i^{j+1}$. Specifically, each element of this set, c_i^j, is a positive integer that describes the number of samples that the node takes during any specific time interval[1].

Each node has private information regarding the information content of the samples that it acquires, and this is represented by an array of 2-tuples $\mathfrak{F}_i = \left[(0,0), \left(c_i^1, v_i^{c_i^1}\right), \ldots, \left(c_i^{s_i}, v_i^{c_i^{s_i}}\right)\right]$, where the first value of each tuple is the number of samples that the node may take and $v_i^{c_i^j}$ is the corresponding information content for those c_i^j samples. The information value of the visual data generally increases as the node takes more samples[2].

[1]We remark that the model described so far is the same as that for the information-based adaptive sampling in chapter 3 (in section 3.2). For this reason, we use the same notations as those used before.

[2]Our algorithms are, however, not restricted to any particular types of information valuation function.

We assume that should the node choose to acquire no samples, it will yield zero information value (see later in this section for a more detailed discussion of this information value). Furthermore, we assume that the visual data captured by a node needs to be processed at the base station with that of other nodes and, therefore, the information content of the data will only be accounted for if it arrives successfully at the base station.

We further assume that each node has an energy budget, B_i (also a private value initially known only to the node), such that its total energy consumption over a given period of time cannot exceed this budget, and here, we consider three specific kinds of energy consumption for each node in the network; namely the energy required to (i) acquire, e_i^{s}, (ii) transmit, e_i^{Tx}, and (iii) receive, e_i^{Rx}, a single sample. We disregard the energy required for other types of processing since it is negligible in comparison. Now, since the node has to transmit its own data towards the base station, the total energy required for this activity is thus $E_i^{\mathrm{S}} = e_i^{\mathrm{s}} + e_i^{\mathrm{Tx}}$ per sample (we will later on refer to the combination of these processes as *sensing*). Similarly, the node could potentially spend a portion of its energy to help its neighbourhood nodes to *forward* their own samples (and/or data that these nodes are forwarding for another node). This incoming data forwarding process requires a total energy of $E_i^{\mathrm{F}} = e_i^{\mathrm{Rx}} + e_i^{\mathrm{Tx}}$ per sample.

Each node initially stores its collected samples into its local memory buffer in order to be transmitted at a later stage. The transmission period and interval are predetermined. During each transmission phase, the transmitter module of each node is turned on for the purpose of transmitting data or message packets to the base station in a multi-hop fashion since some nodes might be out of the base station's wireless range (recall figure 4.1), and hence, require the help of others to relay their packets. Battery-powered visual sensor nodes typically offer reasonably small on-board memory and, hence, at the end of the transmission phase, each node's memory buffer is flushed, reinitialized, and ready to store new sampled data (Mathur et al., 2006).

We describe the route through which the samples, c_i^j, will be transmitted to the base station by the vector $R(c_i^j) = (r_i^1, \ldots, r_i^b)$, where $r_i^l \in I$. The first element of this vector is the origin node i that actually takes the samples. Each subsequent element of this vector is unique and r_i^l will forward the data to r_i^{l+1}. Thus, for the data value of c_i^j samples to be taken into account, its routing set must contain the base station node as its last node.

Given the formal description of the problem above, we now wish to maximise the value of the collected data that arrives at the base station. That is, we wish to solve:

$$\mathcal{D}_i^* = \arg \max_{\{\mathcal{D}_i, \mathfrak{F}_i\}} \sum_{i=1}^{n} \sum_{c_i^j \in C_i} d_i^{R(c_i^j)} v_i^{c_i^j} \tag{4.1}$$

In this expression, $d_i^{R(c_i^j)} \in \mathcal{D}_i \in \{0, 1\}$, is a decision variable where "1" represents a state where the node carries out the corresponding c_i^j sampling action and the samples follow the $R(c_i^j)$ route to arrive at the base station, and "0" represents the state where the node does not carry out the corresponding c_i^j sampling action. This objective function is maximised subject to the energy budget constraint on each node $i \in I$, such that:

$$B_i \geq c_i^j E_i^{\mathrm{S}} + f_i E_i^{\mathrm{F}} \tag{4.2}$$

where f_i represents the total incoming data (or forwarded number of samples from its set of neighbourhood nodes D_i), and is given by:

$$f_i = \sum_{d \in D_i} c_d^j + f_d \tag{4.3}$$

where $i \in R(c_d^j)$. Note that the total outgoing number of samples from node i is thus $c_i^j + f_i$. We must also constrain the node to choose one and only one sampling rate, such that:

$$\sum_{c_i^j \in C_i} d_i^{R(c_i^j)} = 1 \tag{4.4}$$

for all different possible routes in the network.

Now, consider a simple scenario expected in figure 4.3(a). Here, nodes i, 1, and 3 have their own sets of sampling actions with s_i, s_1, and s_3 being different numbers of sampling actions respectively. The nodes also have their corresponding energy budgets, B_i, B_1, and B_3 to perform either sampling and transmitting their own data (requiring E_i^{S} amount of energy per sample) and/or receiving and forwarding the other nodes' data (requiring E_i^{F} amount of energy per sample). For the sake of simplicity, we here assume E_i^{S} requires 8 units of energy and E_i^{F} requires 12 units of energy[3], $\forall i \in I$. Each sampling action has a corresponding information value if the samples (collected by the corresponding sampling action) successfully arrive at the base station. In this case, it turns out that the optimal resource allocation for maximising the gathered information value is where node 1 spends its energy to acquire only two samples, while reserving a portion to relay one and two of nodes i's and 3's samples respectively. The information value collected at the base station is optimized with these sensing and forwarding actions (acquiring the total value of 52.60).

In the case of a network with loops, as depicted in figure 4.3(b), node i now has two options to route its data through; namely (i) node 1 which results in route $R(c_i^j) =$ (i, 1, base station) and (ii) node 2 which results in route $R(c_i^j) =$ (i, 2, base station). In this case, we simply disregard both nodes 4's and 5's sets of possible sampling actions for the sake of simplicity. With our effective and efficient coordination mechanism (which we will describe shortly in the next section), node 2 will decide to use up all its energy to sample five times as both nodes 4 and 5 do not sample. Node 1, however, will sample

[3]This is just for illustration purposes and our algorithms will naturally work for any values.

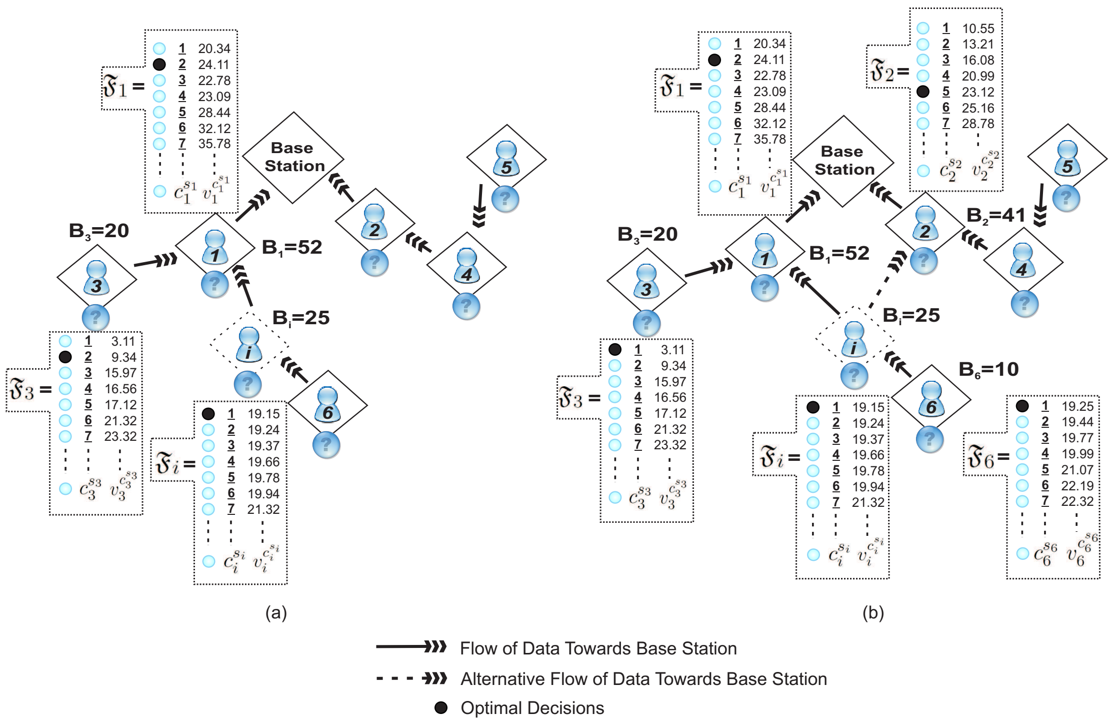

Base Station
B3=20
B1=52
Bi=25
B2=41
B6=10
(a)
(b)
Flow of Data Towards Base Station
Alternative Flow of Data Towards Base Station
Optimal Decisions

FIGURE 4.3: A WVSN configured using the wireless transmission range model as in figure 4.1 and, thus, forms (a) a connected and undirected tree structured network (of which any two nodes are connected by exactly one path) and (b) an arbitrary network with loops. Each node only has a single transmission level. We further assume that communication is bi-directional and multiple nodes within range can thus establish a connection.

only twice and reserve some of its energy to relay one of each node *i*'s, 3's and 6's samples. The optimal information value collected with these nodes' behaviour is thus 88.74.

For both these small sized networks with complete information, a naïve approach to finding the optimal subset of actions is to simply enumerate all possible combinations. This approach, however, is too computationally intensive and works only for very small problems as it very rapidly becomes intractable. For instance, consider the case in figure 4.4(a) in which each node has 30 different sampling actions, the naïve algorithm would now need to evaluate approximately $3 \cdot 10^{29}$ ($\prod_{i=1}^{n} c_i^{s_i}$) solutions (where n is the number of nodes within the the network), and considerably more possibilities ($\prod_{i=1}^{n} (q_i \cdot c_i^{s_i})$, where q_i is the total number of unique routes from node i to the base station) for the case in figure 4.4(b). These are clearly impossible to compute in a reasonable amount of time regardless of processor speed.

Against this background, the problem as formulated above, can be viewed as being similar to *multiple-choice knapsack*[4] problems (i.e. NP-complete resource allocation or

[4]There are m items and the set of all items is $T = \{1, \ldots, m\}$. Each item $t \in T$ has a value v_t and a weight w_t. The items are subdivided into o categories and exactly one item must be taken from each category. The maximum weight that can be carried in a bag is G. Given these, we need to determine

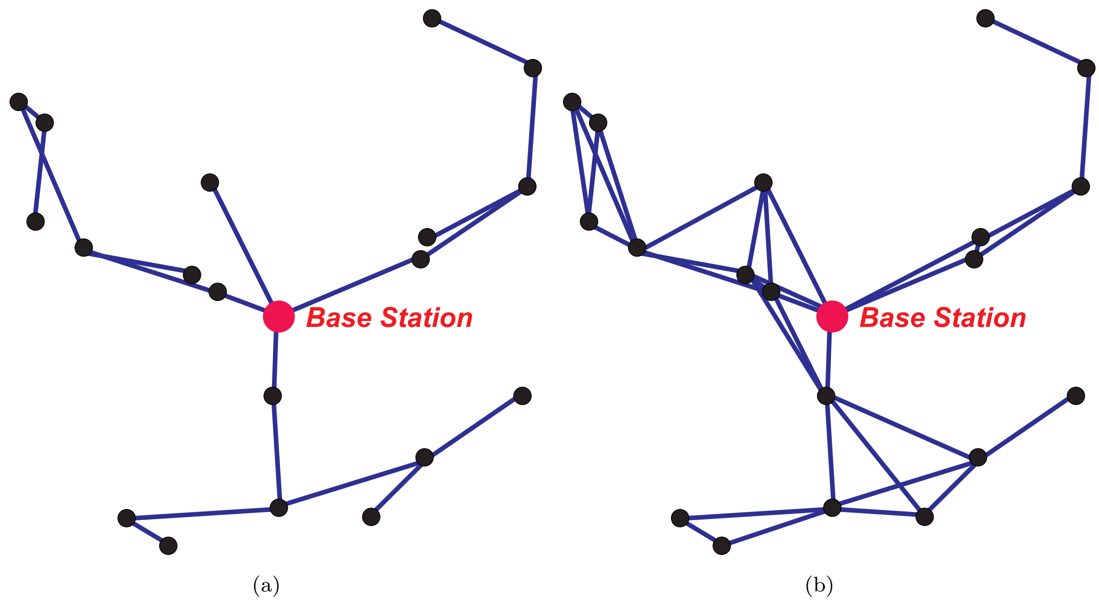


FIGURE 4.4: (a) A tree-structured network formed when each node makes an arbitrary choice of route from (b) a randomly created and connected WVSN (of 20 nodes, each of which is represented by a black filled dot) whose underlying communication network exhibits loops.

distributed constraint optimization problems) (Sinha and Zoltners, 1979), that exhibit the *optimal substructure* property. This property means that the optimal solution can be constructed efficiently from optimal solutions to its subproblems. Given this insight, we propose algorithms to solve this problem based on the sort of computationally efficient dynamic programming technique that are often used on such knapsack problems (see section 2.2).

Having formulated our problem, we need to describe an information metric to value the various visual data that the nodes may acquire. In our particular case, we seek a generic metric in order to simply express the nodes' satisfaction of any particular sensing, forwarding, and routing action when faced with a choice between different alternatives. Now, assuming that we concern only about portions of data rather than anything specific about image data, and given the fact that more samples will generally generate data with a higher information content, we choose a simple linear information valuation function given by:

$$v_i^{c_i^j} = \alpha_i c_i^j \tag{4.5}$$

where α_i is a weighting factor (with support $[0, 1]$) that models the typical situation that the sensors within the network are heterogeneous, have different capabilities (i.e. the resolution of their cameras, the quality of their optics, or the sophistication of their image processing algorithms) and varying fields of view. Hence, samples from some sensors will

which items to include in the bag such that the total weight does not exceed its given limit, while the total value is maximised.

contribute more to the total amount of information collected at the base station than others (Rinner et al., 2008). However, we remark that our algorithms are not restricted to this linear information valuation function, and in some domains, it may be more appropriate to model the information as a strictly concave function where continuing to increase the sampling rate generates decreasing gains in information content[5].

4.4 The Algorithms

Having described our problem representation and information metric, we now focus on the algorithms that seek to use the limited resources on each node in order to maximise the total value of information that arrives at the base station. To do so, nodes are required to make optimal use of their energy resources to cooperatively sense and forward data to the base station. In doing so, they should place higher priority on those samples that have a higher information content, and this is achieved by exchanging coordination messages between connected nodes. In particular, samples that are less important are sensed and relayed only if there is sufficient remaining energy to do so. Thus, under this control regime, some nodes might need to sacrifice their own limited energy to forward other node's data, rather than taking any samples themselves.

To this end, we distinguish three types of messages being exchanged by the nodes; namely (i) *actual data messages* containing visual data sampled by the nodes, and two types of coordination messages composed of (ii) *meta-data messages* that describe the information content of the visual data together with the number of samples taken to produce that data (i.e. $v_i^{c_i^j}$ and c_i^j respectively), and (iii) *control messages* that contain the allocation decisions. In WVSNs, the size of the actual data messages overwhelms that of the coordinations messages and, hence, exchanging these before sending the actual data, can significantly increase the information collected at the base station by making more efficient use of each node's constrained energy.

The goal of the algorithms that we derive is to calculate the optimal sensing, forwarding, and next-hop decisions of each node. This is given by:

$$Cmax_I = \{(i, c_i^j, R(c_i^j)) | d_i^{R(c_i^j)} = 1, \forall i \in I, \forall c_i^j \in C_i, \forall d_i^{R(c_i^j)} \in \mathcal{D}_i^*\} \tag{4.6}$$

and represents a set of 3-tuples indicating for each node in the network, the sensing and forwarding rates that it should adopt and the route that it should use to transmit

[5]We did not chose to use the mean Fisher information metric (as described in the previous chapter) based on a GP regression to infer the temporal variation characteristics of the visual data being sensed. This is because this metric is computationally intensive and would significantly affect the performance of our algorithms but solely during the times when each node needs to evaluate the information content of its potential sets of readings. More comprehensive information valuation functions in the domain of image processing could have also been used. However, they would also typically be too computationally intensive to be run on nodes with the physical constraints of computational and power resources.

its own and its forwarded data to the base station, in order to maximise the objective function in equation 4.1, subject to the constraints in equations 4.2 and 4.4.

We now present our two novel algorithms. Both of them are efficient as they satisfy the data flow conservation of the network where no energy is wasted by transmitting data that later will not be forwarded to the final destination. We start with the algorithm that assumes that the route by which data is forwarded to the base station is fixed, and then progress to the other that assumes flexible routing and makes optimal decisions regarding both the sensing and forwarding actions that each node should perform, and also the route by which this data should be forwarded to the base station.

4.4.1 The Algorithm With Fixed Routing

In this case, each node $i \in I$ can only forward its data to exactly one other node (which will later be referred to as its *parent*). This may be because the underlying communication network of the WVSN is tree structured or because it actually exhibits loops but an arbitrary choice of route has been made (effectively turning the loopy communication network into a tree). An example of a WVSN whose underlying network structure is a tree is shown in figure 4.5. We remark that in such tree-structured networks, there is only one unique route between each node and the base station (e.g. $R(c_4^j) = (4,\ 2,\text{ base station})$, $R(c_3^j) = (3,\ 1,\text{ base station})$, and $R(c_6^j) = (6,\ i,\ 1,\text{ base station})$).

In general, the nodes within a network will deplete their energy resources at different rates since they will have different sampling rates, and will be transmitting and forwarding different quantities of visual data. Each node $i \in I$ thus needs to compute the highest information value it can transmit by using at most $b_i^k \leq B_i$ of its energy. As we have described in section 4.3, the energy consumption of node i only includes E_i^{S} and E_i^{F} (i.e. the energy to acquire and transmit a sample, and the energy required to receive and transmit a forwarded sample from another node). It is therefore sufficient that b_i^k satisfies the following criteria:

$$\begin{aligned} & b_i^k = c_i^j E_i^{\mathrm{S}} + f_i E_i^{\mathrm{F}} \qquad \text{where } c_i^j, f_i \geq 0 \\ & b_i^k \leq B_i \end{aligned} \tag{4.7}$$

where c_i^j is its own number of samples and f_i is the number of forwarded incoming samples.

Now, let $\mathfrak{O}_i = \left[\left(b_i^1, Vmax_i^1, Cmax_i^1\right), \ldots, \left(b_i^{K_i}, Vmax_i^{K_i}, Cmax_i^{K_i}\right)\right]$ denote an array of 3-tuples sorted incrementally by b_i^k where $k = 1, \ldots, K_i$, and b_i^k is the energy limit that satisfies equation 4.7 (i.e. K_i is the number of feasible b_i^k-s for node i) which will later on be referred to as the labels of $\mathfrak{O}_i$, $Vmax_i^k$ is the maximum information value that node i can transmit to its parent by using at most b_i^k, and $Cmax_i^k$ is the set of sensing

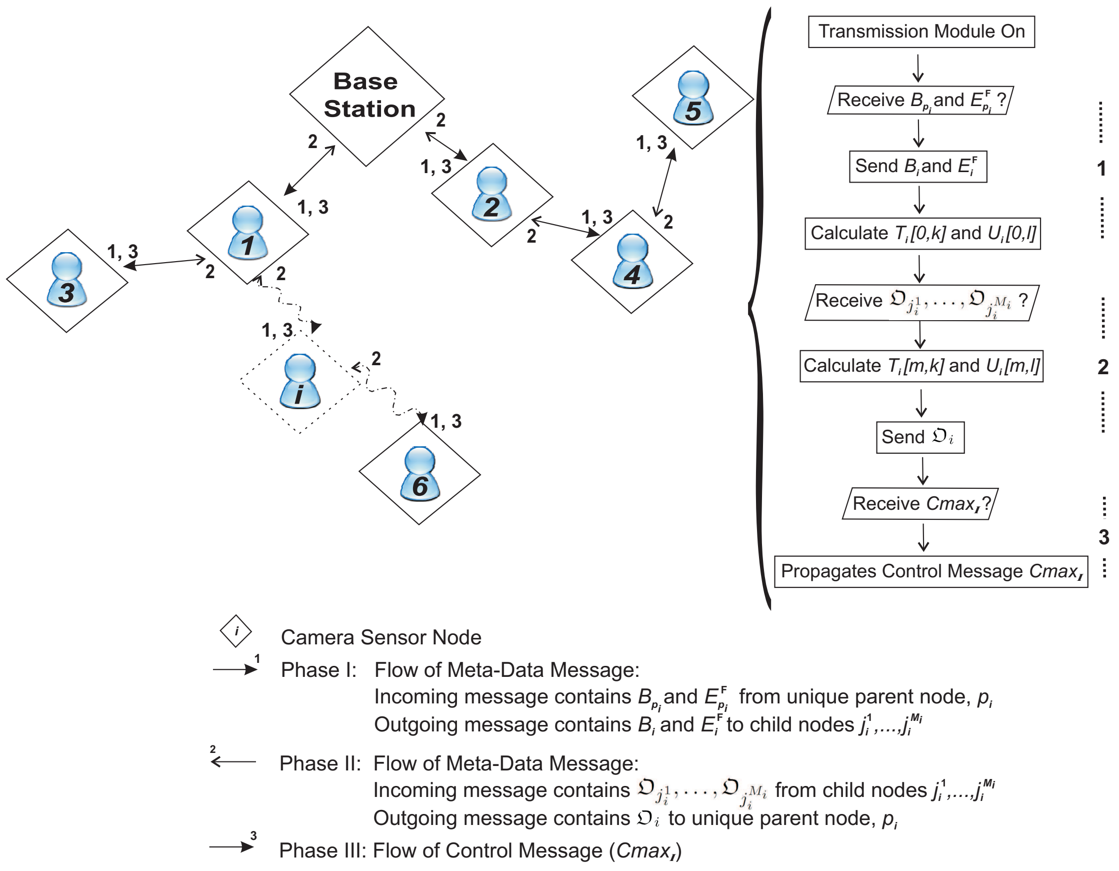

Base Station
1
2
3
4
5
6
i
1, 3
2
Transmission Module On
Receive B_{p_i} and $E^F_{p_i}$?
Send B_i and E^F_i
Calculate $T_i[0,k]$ and $U_i[0,l]$
Receive $\mathfrak{D}_{j_i^1},\dots,\mathfrak{D}_{j_i^{M_i}}$?
Calculate $T_i[m,k]$ and $U_i[m,l]$
Send $\mathfrak{D}_i$
Receive $Cmax_l$?
Propagates Control Message $Cmax_l$
Camera Sensor Node
Phase I: Flow of Meta-Data Message:
Incoming message contains B_{p_i} and $E^F_{p_i}$ from unique parent node, p_i
Outgoing message contains B_i and E^F_i to child nodes $j_i^1,\dots,j_i^{M_i}$
Phase II: Flow of Meta-Data Message:
Incoming message contains $\mathfrak{D}_{j_i^1},\dots,\mathfrak{D}_{j_i^{M_i}}$ from child nodes $j_i^1,\dots,j_i^{M_i}$
Outgoing message contains $\mathfrak{D}_i$ to unique parent node, p_i
Phase III: Flow of Control Message ($Cmax_l$)

FIGURE 4.5: The flow of the algorithm in a connected and undirected tree-structured WVSN.

and forwarding actions that will produce data with the value of $Vmax_i^k$. This set of actions is calculated by taking into account its own and its descendants' data.

The algorithm installed on each node runs in three phases (see the state diagram in figure 4.5 and algorithm 3). In the first, meta-data message propagation is initiated by the base station. To this end, messages containing the value of each node's energy budget, B_i, and energy consumption for forwarding, E_i^{F}, are propagated down the tree (i.e. as soon as any node receives this information from its unique parent node, p_i (see state 1 or line 7), it sends its own information to its set of children, $J_i = \left\{j_i^1, \dots, j_i^{M_i}\right\}$ (line 9)). Having sent this information each node i then enters an idle mode in which it waits for the $\mathfrak{D}$ meta-data arrays from its child nodes. With regard to figure 4.5, the leaf nodes (3, 5, and 6) eventually receive the meta-data message of nodes 1, 4, and i respectively.

In the second phase, node i waits until it has received all the $\mathfrak{D}$ meta-data arrays from its children (denoted by $\mathfrak{D}_{j_i^1}, \dots, \mathfrak{D}_{j_i^{M_i}}$, see state 2 or lines 14-15), and then calculates its own $\mathfrak{D}_i$ (line 18). If $M_i = 0$, then node i is a leaf node. With regard to figure 4.5, nodes 3, 5, and 6 are the leaf nodes and, consequently, they correspondingly calculate $\mathfrak{D}_3$, $\mathfrak{D}_5$, and $\mathfrak{D}_6$ which is then sent to their respective parent nodes 1, 4, and i. We will

Algorithm 3 Optimal adaptive sensing and forwarding with fixed routing.

```
 1: loop
 2:   if $sTime = NOW$ then                                          ▷ Time to sample
 3:     $readings \leftarrow$ PerformSampling($sTime$)                  ▷ Sampling action, $c_i^j$
 4:     SetSTime($sTime + sRate$)
 5:   end if
 6:   if $tTime = NOW$ then          ▷ Time to transmit, transmission module is turned on
 7:     $[B_{p_i}, E^{\mathrm{F}}_{p_i}] \leftarrow$ WaitMetaData($p_i$)   ▷ Receives $B_{p_i}$ and $E^{\mathrm{F}}_{p_i}$ from its unique parent node, $p_i$
 8:     for each $j_i^m \in J_i$ do              ▷ Iterates each child node in $J_i = \left\{j_i^1, \ldots, j_i^{M_i}\right\}$
 9:       SendMetaData($j_i^m$, $[B_i, E_i^{\mathrm{F}}]$)          ▷ Sends $B_i$ and $E_i^{\mathrm{F}}$ to child node $j_i^m$
10:     end for
11:     CalcFirstRowTables($readings$)   ▷ Calculates the 1st rows of $T_i$ and $U_i$ using (4.8) and (4.11) respectively
12:     if $\neg leafNode$ then
13:       for each $j_i^m \in J$ do
14:         $\mathfrak{D}_{j_i^m} \leftarrow$ WaitMetaData($j_i^m$)     ▷ Receives $\mathfrak{D}_{j_i^m}$ from child node $j_i^m$
15:         CalcTheRestTables($\mathfrak{D}_{j_i^m}$)   ▷ Calculates the other rows of $T_i$ and $U_i$ using (4.9) and (4.12) respectively
16:       end for
17:     end if
18:     $\mathfrak{D}_i \leftarrow$ CalcMetaDataArray()     ▷ Determines $\mathfrak{D}_i$ which is basically the last row of $U_i$
19:     SendMetaData($p_i, \mathfrak{D}_i$)                  ▷ Sends $\mathfrak{D}_i$ to unique parent node, $p_i$
20:     $Cmax_I \leftarrow$ WaitControlMessage($p_i$)   ▷ Receives control message from unique parent node, $p_i$
21:     PropagateControlMessage($j_i^m, Cmax_I$)   ▷ Sends control message to each child node, $j_i^m \in J_i$
22:     PerformTransmit($readings, Cmax_I$)
23:     SetNodeOptimalBehaviour($Cmax_I$)   ▷ Sets node's optimal sensing and forwarding actions
24:     SetTTime($tTime + tRate$)                  ▷ Node sets its next transmitting time
25:     $readings \leftarrow \{\}$
26:   end if
27: end loop
```

discuss this illustrative example in more detail in the next section.

Now, let T_i be a table with $M_i + 1$ rows numbered from 0 to M_i, and K_i columns, where K_i is the number of all the b_i^k values that satisfy equation 4.7. Thus, each column k has label b_i^k. Let $T_i\,[x, y]$ denote the element of the table that is in the x^{th} row and the column with label b_i^y. $\mathfrak{D}_{j_i^m}\,[x]$ is the x^{th} element of $\mathfrak{D}_{j_i^m}$, which is a 3-tuple. As every node could choose not to sample (yielding zero value), then $\mathfrak{D}_{j_i^m}\,[0] = T_i\,[m, 0] = 0$ for all $0 \leq m \leq M_i$. Moreover, we can assume that the set of labels in each $\mathfrak{D}_{j_i^m}$ that node i has received is the same as the set of labels in its table T_i. We will show how we can guarantee this later on. T_i's other elements are filled as follows:

$$T_i\,[0, k] = \max\{v_i^{c_i^j}\} \tag{4.8}$$

$$T_i\,[m, k] = \max_{0 \leq r \leq k} \left\{T_i\,[m-1, r] + Vmax_{j_i^m}^{k-r}\right\} \tag{4.9}$$

for all $1 \leq k \leq K_i$ and $1 \leq m \leq M_i$, where $(c_i^j, v_i^{c_i^j}) \in \mathfrak{F}_i$, and $\mathfrak{F}_i$ is the array of 2-tuples defined in the previous section.

According to equation 4.8, we can see that $T_i\,[0, k]$ stores the maximum information value of data that can be delivered to node i's parent by sensing only (with the energy consumption not exceeding the energy limit b_i^k). Due to the fact that each of the sets of labels in $\mathfrak{D}_{j_i^m}$ is equivalent to the set of labels of table T_i, equation 4.9 gives the maximum value of data that node i can deliver to its parent (noting that this data now does not only include its own sensed data but also its children's data that will potentially be forwarded through it). Hence, $T_i[1,k]$ is the maximum value that can be sent by taking into account the sensed data and the data from j_i^1, with respect to the b_i^k energy limit. $T_i[2,k]$ stores the maximum value when the data from child node j_i^2 is also included. In general, $T_i[m,k]$ is the maximum information value that node i can transmit to its parent, given the b_i^k energy limit. The data considered is the potential forwarded data from child nodes $j_i^1,\ldots,j_i^m$ and node i's own sensed data. Note that while it is necessary to construct the entire table, as in conventional dynamic programming solutions to the multiple-choice knapsack problem, it is only the last row that provides useful meta-data regarding the maximum information values of data that can be transmitted given different feasible values of b_i^k. Indeed, it is only the last element of this row that represents the maximal information value that node i can transmit to the parent node.

Now, the next step of the algorithm is to calculate $\mathfrak{O}_i$. To do so, let U_i denote a table similar to T_i. However, its labels b_i^l, now, satisfy the following:

$$\begin{aligned} & b_i^l = (c_i^j + f_i)E_{p_i}^{\mathrm{F}} \qquad \text{where } c_i^j, f_i \geq 0 \\ & b_i^l \leq B_{p_i} \end{aligned} \tag{4.10}$$

where B_{p_i} is the energy budget of i's unique parent node, p_i, and $E_{p_i}^{\mathrm{F}}$ is the value of energy consumption of the parent for forwarding a sample. Recall that these values were delivered to node i in the first stage. Let L_i denote the number of all b_i^l that satisfy equation 4.10. Similarly, we can calculate table U_i's elements in a similar fashion to those of T_i as described earlier, but with the new labels:

$$U_i[0,l] = \min\left(\max\{v_i^{c_i^j}\}, T_i[0,K_i]\right) \tag{4.11}$$

$$U_i[m,l] = \min\left(\max_{0\leq r\leq l}\left\{U_i[m-1,r] + Vmax_{j_i^m}^{l-r}\right\}, T_i[m,K_i]\right) \tag{4.12}$$

for all $1 \leq l \leq L_i$ and $1 \leq m \leq M_i$, where $(c_i^j, v_i^{c_i^j}) \in \mathfrak{F}_i$.

We can now construct the meta-data array of node i such that $\mathfrak{O}_i = \left[\left(b_i^1, U_i[M_i,1], Cmax_i^1\right),\ldots,\left(b_i^{L_i}, U_i[M_i,L_i], Cmax_i^{L_i}\right)\right]$, where $U_i[M_i,l]$ is the maximum information value that node i can transmit to its parent node (by using at most b_i^k energy) which can subsequently forward the received i's data by using at most b_i^l energy. $Cmax_i^l$ is the set of sensing and forwarding actions that will produce data with the value of $U_i[M_i,l]$. Hence, once $\mathfrak{O}_i$ is sent to the parent node, its labels will be the same as those in table T_{p_i} of the parent node. This second phase meta-data message containing $\mathfrak{O}_i$ propagates up the network arriving back at the base station (line 19).

In the third phase of the algorithm, each parent node will have received meta-data arrays from all of its children. The base station will be able to calculate the highest information value it can potentially receive from all the nodes beneath it in the network.

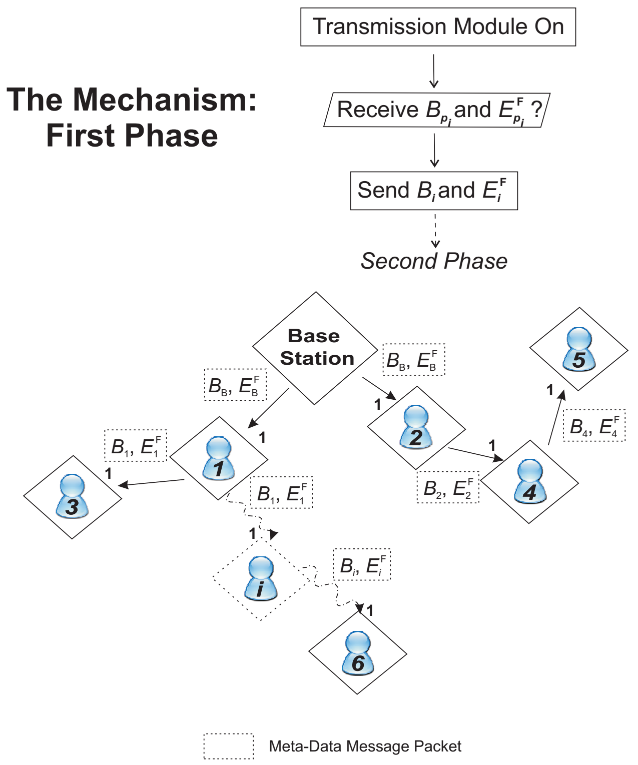
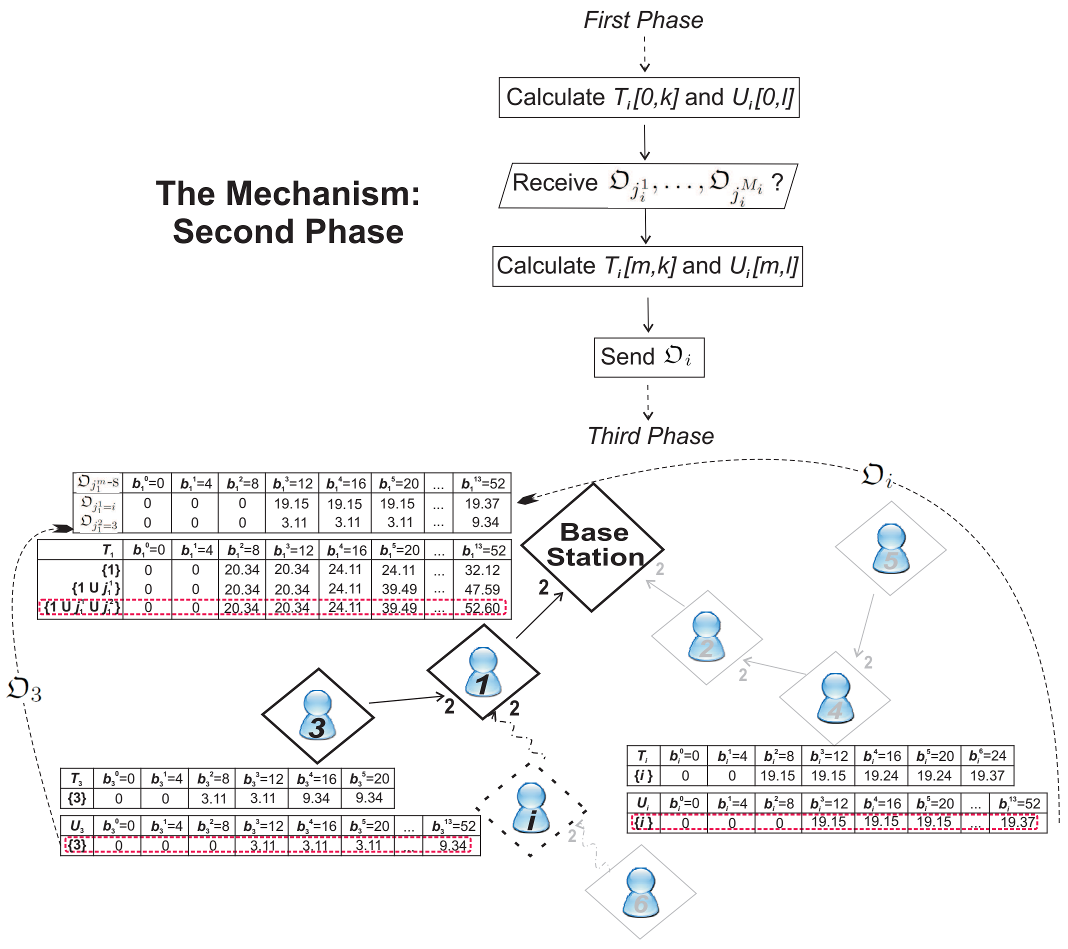


FIGURE 4.6: The first phase of the algorithm with fixed routing.

Based on the structure of $\mathfrak{D}_i$, each node i can easily determine what amount of data it should receive from each child node and, hence, how many samples it should acquire and transmit itself. A control message containing this set is then propagated down the network (see state 3 or lines 20-21), and this control message informs each node of its optimal sensing and forwarding decisions (lines 22-23). In this way, there is a guarantee that all of the data transmitted by each node will reach the base station. The control message eventually reaches the leaf nodes which then start to acquire and transmit visual data as planned.

4.4.2 Illustrative Example

To illustrate how the algorithm works in practice, reconsider the scenario expected in figure 4.3(a). For the sake of simplicity, we can assume here, that nodes with no $\mathfrak{F}_i$ (nodes 2, 4, 5, and 6) do not have anything interesting worth sampling in their fields of view. Each of nodes i, 1, and 3 has as one of their private values, s_i, s_1, and s_3 different numbers of sampling actions and thus, C_i, C_1, and C_3 sets of sampling actions (or rates) respectively. The nodes have as well their corresponding $\mathfrak{F}_i$, $\mathfrak{F}_1$, and $\mathfrak{F}_3$ regarding the information content of the samples that they acquire (which are chosen arbitrarily for illustrative purposes), and energy budgets, B_i, B_1, and B_3 to perform either sampling and transmitting their own data (requiring $E_i^{\mathrm{S}} = 8$ units of energy per sample, $\forall i \in I$), and/or forwarding the other nodes' data (requiring $E_i^{\mathrm{F}} = 12$ units of energy per sample,


First Phase
Calculate $T_i[0,k]$ and $U_i[0,l]$
Receive $\mathfrak{D}_{j_i^1}, \ldots, \mathfrak{D}_{j_i^{M_i}}$?
Calculate $T_i[m,k]$ and $U_i[m,l]$
Send $\mathfrak{D}_i$
Third Phase
The Mechanism: Second Phase
Base Station
$\mathfrak{D}_i$
$\mathfrak{D}_3$
1
2
3
4
5
6
i

$\mathfrak{D}_{j_1^m}$-S	$b_1^0=0$	$b_1^1=4$	$b_1^2=8$	$b_1^3=12$	$b_1^4=16$	$b_1^5=20$	...	$b_1^{13}=52$
$\mathfrak{D}_{j_1^1=i}$	0	0	0	19.15	19.15	19.15	...	19.37
$\mathfrak{D}_{j_1^2=3}$	0	0	0	3.11	3.11	3.11	...	9.34

T_1	$b_1^0=0$	$b_1^1=4$	$b_1^2=8$	$b_1^3=12$	$b_1^4=16$	$b_1^5=20$	...	$b_1^{13}=52$
{1}	0	0	20.34	20.34	24.11	24.11	...	32.12
$\{1 \cup j_1^1\}$	0	0	20.34	20.34	24.11	39.49	...	47.59
$\{1 \cup j_1^1 \cup j_1^2\}$	0	0	20.34	20.34	24.11	39.49	...	52.60

T_3	$b_3^0=0$	$b_3^1=4$	$b_3^2=8$	$b_3^3=12$	$b_3^4=16$	$b_3^5=20$
{3}	0	0	3.11	3.11	9.34	9.34

U_3	$b_3^0=0$	$b_3^1=4$	$b_3^2=8$	$b_3^3=12$	$b_3^4=16$	$b_3^5=20$	...	$b_3^{13}=52$
{3}	0	0	0	3.11	3.11	3.11	...	9.34

T_i	$b_i^0=0$	$b_i^1=4$	$b_i^2=8$	$b_i^3=12$	$b_i^4=16$	$b_i^5=20$	$b_i^6=24$
{i}	0	0	19.15	19.15	19.24	19.24	19.37

U_i	$b_i^0=0$	$b_i^1=4$	$b_i^2=8$	$b_i^3=12$	$b_i^4=16$	$b_i^5=20$	...	$b_i^{13}=52$
{i}	0	0	0	19.15	19.15	19.15	...	19.37

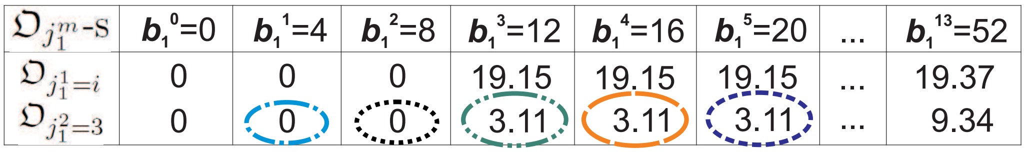
FIGURE 4.7: The second phase of the algorithm with fixed routing. $\mathfrak{D}$ meta-data arrays are represented by dotted rectangles.

$\forall i \in I$).

In the first phase (as shown in figure 4.6), the base station initiates the meta-data message propagation. After receiving the base station's meta data message, node 1 sends B_1 and E_1^{F} to both nodes 3 and i. Node 2, however, sends its own meta data to node 4. The leaf nodes (including nodes 3, 5, and 6) eventually receive the meta-data message from their respective parents.

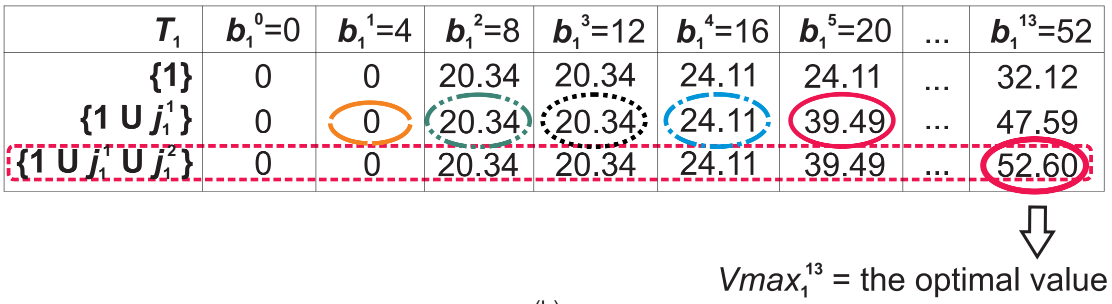
As can be seen in figure 4.7, in the second phase, nodes i and 3 calculate their corresponding tables. More specifically, node i evaluates its T_i and U_i tables, while node 3 finds its T_3 and U_3. The columns of tables T and U represent the energy limits that satisfy equations 4.7 and 4.10 respectively. The rows of the tables represent the set of nodes whose data has been taken into account. For instance in table T_i, where $row = \{i\}$, if node i has b_i^0, b_i^1, b_i^2, b_i^3, b_i^4, b_i^5, and b_i^6 as energy limits, in return it will be able to sense its own data with the maximum values of 0, 0, 19.15, 19.15, 19.24, 19.24, or 19.37. These values are calculated using equation 4.8. Node i then evaluates U_i using equation 4.11. Each cell of this table stores the maximum information value that its parent node 1 could potentially forward given the upper bound value found in table T_i (19.37 in this

TABLE 4.1: (a) $\mathfrak{D}_{j_1^m}$ meta-data arrays that have arrived at node 1 from its child nodes $j_1^1 = i$ and $j_1^2 = 3$, and (b) the T_1 table of node 1.



$\mathfrak{D}_{j_1^m}$-s	b_1^0=0	b_1^1=4	b_1^2=8	b_1^3=12	b_1^4=16	b_1^5=20	...	b_1^{13}=52
$\mathfrak{D}_{j_1^1=i}$	0	0	0	19.15	19.15	19.15	...	19.37
$\mathfrak{D}_{j_1^2=3}$	0	0	0	3.11	3.11	3.11	...	9.34

(a)



T_1	b_1^0=0	b_1^1=4	b_1^2=8	b_1^3=12	b_1^4=16	b_1^5=20	...	b_1^{13}=52
{1}	0	0	20.34	20.34	24.11	24.11	...	32.12
{1 U j_1^1}	0	0	20.34	20.34	24.11	39.49	...	47.59
{1 U j_1^1 U j_1^2}	0	0	20.34	20.34	24.11	39.49	...	52.60

$Vmax_1^{13}$ = the optimal value

(b)

case). Node i can now construct its meta-data array from only the last row of table U_i such that $\mathfrak{D}_i = \left[\left(b_i^1, U_i\left[M_i, 1\right], Cmax_i^1\right), \ldots, \left(b_i^{L_i}, U_i\left[M_i, L_i\right], Cmax_i^{L_i}\right)\right]$.

Node i sends $\mathfrak{D}_i$ to its parent node 1. $\mathfrak{D}_{j_1^1=i}$ then arrives (as shown in table 4.1(a) where $row = \mathfrak{D}_{j_1^1=i}$) from child nodes $j_1^1 = i$ containing the maximum values of node i's sensed data that node 1 could potentially forward to node 1's parent node given node 1's energy limits (that is $Vmax^0_{j_1^1=i}$, $Vmax^1_{j_1^1=i}$, $Vmax^2_{j_1^1=i}$, $Vmax^3_{j_1^1=i}$, $Vmax^4_{j_1^1=i}$, $Vmax^5_{j_1^1=i}$, till $Vmax^{13}_{j_1^1=i}$ respectively). Node 3 calculates the values of its own tables and eventually $\mathfrak{D}_3$ in a similar way. Node 3 as well sends $\mathfrak{D}_3$ to its parent node 1.

After receiving the $Vmax^k_{j_1^1=i}$ and $Vmax^k_{j_1^2=3}$ values of $\mathfrak{D}_{j_1^1=i}$ and $\mathfrak{D}_{j_1^2=3}$ arriving from its child nodes $j_1^1 = i$ and $j_1^2 = 3$ respectively, node 1 can now calculate its T_1 and U_1 tables. The elements of T_1's first row are evaluated using equation 4.8 (by taking into account only node 1's sensed data) as above. The elements of T_1's other rows (both $row = \{i \cup j_1^1\}$ and $row = \{i \cup j_1^1 \cup j_1^2\}$) are, however, found using equation 4.9. These elements represent the maximum information that node 1 could send by taking into account not only its own sensed data, but also the data that could be potentially forwarded from its child nodes i and 3.

For instance where $row = \{i \cup j_1^1\}$ and $column = b_1^2$, node 1 could allocate all its $b_1^2 = 8$ energy resources to either acquire and transmit its own data (resulting in maximum information value of 20.34) or to forward data from its child node j_1^1 (resulting in maximum information value of 0, as node 1 requires at least 12 units of energy to receive and forward a sample from its connected nodes). Alternatively, it could as well divide its $b_1^2 = 8$ energy resources by allocating a half ($b_1^1 = 4$) to its own and another ($b_1^1 = 4$)

to j_1^1 (resulting in maximum information value of 0+0=0, as with 4 units of energy node 1 is not capable of sensing nor forwarding any sample). In this case, it turns out that node 1 is better off spending all its energy for its own sensing activities.

Specifically, where $row = \{i \cup j_1^1 \cup j_1^2\}$ and $column = b_1^5$ (see table 4.1(b)), node 1 could either:

- Allocate all its $b_1^5 = 20$ energy resources to sense its own data and forward j_1^1's data (resulting in a maximum information value of 39.49 by sensing once and forwarding one of node i's samples),
- Allocate all its $b_1^5 = 20$ energy resources to forward data from its other child node j_1^2 (resulting in a maximum information value of 3.11 by forwarding one of node 3's samples),
- Divide its $b_1^5 = 20$ energy resources by allocating a portion of $b_1^1 = 4$ energy resources to both its own and j_1^1, and $b_1^4 = 16$ to j_1^2 (resulting in a maximum information value of 0+3.11=3.11 by only forwarding one of node 3's samples),
- Divide its $b_1^5 = 20$ energy resources by allocating a portion of $b_1^2 = 8$ energy resources to both its own and j_1^1, and $b_1^3 = 12$ to j_1^2 (resulting in a maximum information value of 20.34+3.11=23.45 by sensing once and forwarding one of node 3's samples),
- Divide its $b_1^5 = 20$ energy resources by allocating a portion of $b_1^3 = 12$ energy resources to both its own and j_1^1, and $b_1^2 = 8$ to j_1^2 (resulting in a maximum information value of 20.34+0=20.34 by only sensing once), or
- Divide its $b_1^5 = 20$ energy resources by allocating a portion of $b_1^4 = 16$ energy resources to both its own and j_1^1, and $b_1^1 = 4$ to j_1^2 (resulting in a maximum information value of 24.11+0=24.11 by sensing twice).

In this case, node 1 chooses the first option since by doing so it yields the highest information value of 39.49. T_1's other elements are calculated in a similar way[6]. Note that node 1 is not required to evaluate its U table because its parent node, the base station, has more energy than the ordinary nodes and, hence, is capable of receiving any transmitted data.

As can be seen in figure 4.8, the base station starts the third phase by propagating control message $Cmax_I$ down the network. We can now see that the optimal sensing and forwarding decisions are stored inside $Cmax_1^{K_1=13}$ with the maximum information value $Vmax_1^{K_1=13}$ of 52.60. $Cmax_1^{K_1=13}$ determines what amount of data node 1 will forward for each child node and, hence, how many samples it should acquire and transmit

[6]Note that this is somewhat similar to a knapsack problem where we consider allocating spaces in a bag for different items.

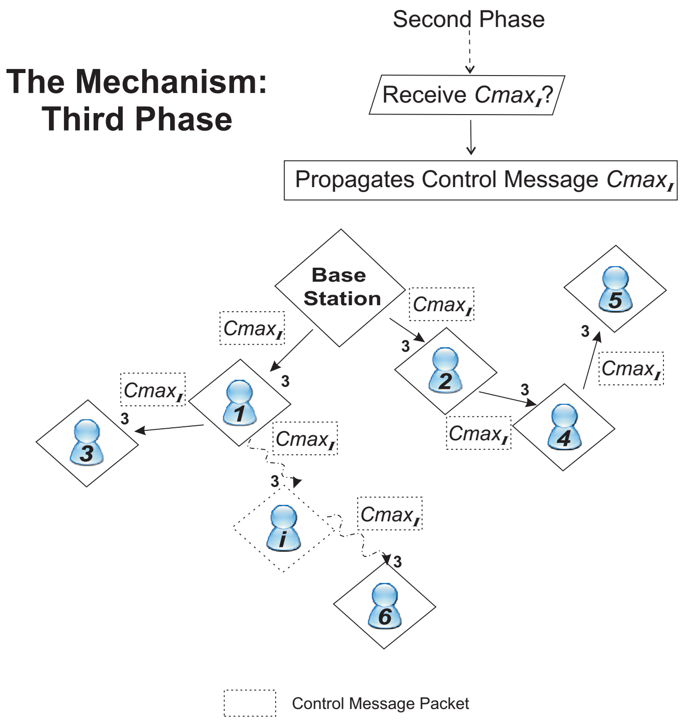


FIGURE 4.8: The third phase of the algorithm with fixed routing.

its own. In this case, nodes i and 3 need to sample once and twice respectively. Node 1, however, spends its energy to take only two samples, while reserving a portion to forward one and two of node i's and 3's samples respectively. This third phase of the mechanism satisfies the data flow conservation of the network by guaranteeing that all taken samples will eventually reach the base station.

4.4.3 The Algorithm With Flexible Routing

Next, we consider the algorithm which assumes flexible routing, and makes optimal decisions regarding both the sensing and forwarding actions that each node should perform, and also the route by which data should be forwarded to the base station (see figure 4.9 for an illustration of this case). In order to make the routing decision tractable, we place one minor restriction on the routes that our algorithm can consider. Specifically, we assume that the nodes always forward their data toward the base station; that is, they will not forward data to a node that is further from the base station (in terms of hop count) than themselves. We believe this is a reasonable assumption. There might be cases where several nodes are better off taking longer paths. However, in general, such paths will deplete the energy resources of a greater number of nodes, and are thus unlikely to be optimal solutions. Furthermore, we assume that the data samples of a node can only be sent as a bundle (i.e. they are indivisible). The data readings of different nodes could be, however, sent through different routes if there is more than one option to choose from.

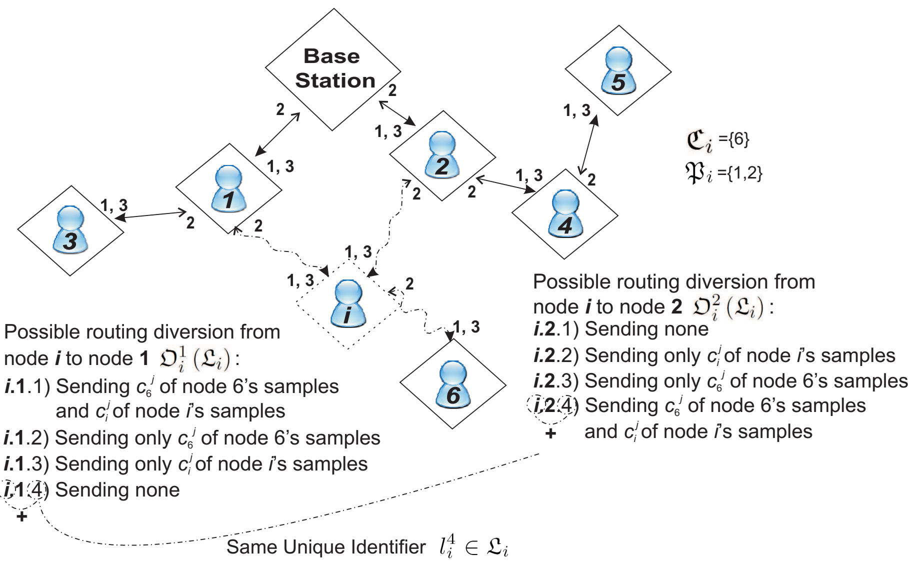

Base Station
1
2
3
4
5
6
i
1, 3
2
$\mathfrak{C}_i$ ={6}
$\mathfrak{P}_i$ ={1,2}
Possible routing diversion from node *i* to node **1** $\mathfrak{O}_i^1(\mathfrak{L}_i)$:
i.**1**.1) Sending c_6^j of node 6's samples and c_i^j of node *i*'s samples
i.**1**.2) Sending only c_6^j of node 6's samples
i.**1**.3) Sending only c_i^j of node *i*'s samples
i.**1**.4) Sending none
Possible routing diversion from node *i* to node **2** $\mathfrak{O}_i^2(\mathfrak{L}_i)$:
i.**2**.1) Sending none
i.**2**.2) Sending only c_i^j of node *i*'s samples
i.**2**.3) Sending only c_6^j of node 6's samples
i.**2**.4) Sending c_6^j of node 6's samples and c_i^j of node *i*'s samples
+
+
Same Unique Identifier $l_i^4 \in \mathfrak{L}_i$

FIGURE 4.9: The flow of the algorithm that assumes flexible routing and makes optimal decisions regarding both sensing, forwarding, and next-hop decisions (or routing actions). The phases involved in this algorithm are similar to those in the algorithm for fixed routing.

With these assumptions, we now need to organize the nodes into different *levels*. To this end, a network simulator using a *BGP*-based[7], robust, and scalable routing discovery TCP/IP protocol is developed (more details of which are given in section 4.5). Our coordination mechanism will therefore sit on top of this network protocol. The first level consists of all the nodes that have a 1-hop shortest path to the base station (nodes 1 and 2 in figure 4.9). Nodes that belong to the second level have a 2-hop shortest path to the base station (nodes i, 3, and 4 in figure 4.9). Nodes 5 and 6 have a 3-hop shortest path. Now, according to this hierarchy, each node can only forward its data to higher level nodes within its transmission range. In figure 4.9, for example, node i has two potential shortest routes to choose from; namely (i) node 1 which results in route $R(c_i^j) = (i,\ 1,$ base station) and (ii) node 2 which results in route $R(c_i^j) = (i,\ 2,$ base station). Node i does not consider routing through node 6 since 6 is a greater hop count away from the base station than it is.

Furthermore, as we can see from the figure, node i has potentially two bundles of data to consider (its own and data that it is forwarding for node 6). In addition, it has two possible shortest paths to choose between (either through node 1 or 2 for each of the bundled data). Thus, a number of routing options exist for this node. It could send both bundles of data through node 1, such that $R(c_i^j)$ contains $(i,\ 1,\ \dots)$ and $R(c_6^j)$ contains $(\dots,\ i,\ 1,\ \dots)$, or it could send them through node 2, such that $R(c_i^j)$ contains $(i,\ 2,\ \dots)$ and $R(c_6^j)$ contain $(\dots,\ i,\ 2,\ \dots)$. Other alternatives are to send each of them separately through each possible route, such that $R(c_i^j)$ contains $(i,\ 1,\ \dots)$ and $R(c_6^j)$ contains $(\dots,\ i,\ 2,\ \dots)$, or the other way around.

[7]BGP stands for Border Gateway Protocol and it is a common network routing protocol on the Internet (Feigenbaum et al., 2005).

Algorithm 4 Optimal adaptive sensing and forwarding with flexible routing.

```
1: loop
2:     if $sTime = NOW$ then                                   ▷ Time to sample
3:         $readings \leftarrow$ PerformSampling($sTime$)        ▷ Sampling action, $c_i^j$
4:         SetSTime($sTime + sRate$)
5:     end if
6:     if $tTime = NOW$ then            ▷ Time to transmit, transmission module is turned on
7:         for each $p_i^{\mathfrak{n}} \in \mathfrak{P}_i$ do       ▷ Iterates each parent node, $p_i^{\mathfrak{n}} \in \mathfrak{P}_i$
8:             $[B_{p_i^{\mathfrak{n}}}, E^{\mathrm{F}}_{p_i^{\mathfrak{n}}}] \leftarrow$ WaitMetaData($p_i^{\mathfrak{n}}$)   ▷ Receives $B_{p_i^{\mathfrak{n}}}$ and $E^{\mathrm{F}}_{p_i^{\mathfrak{n}}}$ from parent node $p_i^{\mathfrak{n}}$
9:         end for
10:        for each $j_i^m \in J_i$ do        ▷ Iterates each child node in $J_i = \{j_i^1, \ldots, j_i^{M_i}\}$
11:            SendMetaData($j_i^m$, $[B_i, E_i^{\mathrm{F}}]$)       ▷ Sends $B_i$ and $E_i^{\mathrm{F}}$ to child node $j_i^m$
12:        end for
13:        CalcFirstRowTables($readings$)  ▷ Calculates the 1st rows of $T_i$ and $U_i^{p_i^{\mathfrak{n}}}$ (for each parent node, $p_i^{\mathfrak{n}}$ in $\mathfrak{P}_i$) using (4.8) and (4.11) respectively
14:        $\mathfrak{C}_i \leftarrow \{i\}$                    ▷ Adds this node to the set of descendants $\mathfrak{C}_i$
15:        if $\neg leafNode$ then
16:            for each $j_i^m \in J_i$ do
17:                $\mathfrak{D}^i_{j_i^m} \leftarrow$ WaitMetaData($j_i^m$)       ▷ Receives $\mathfrak{D}^i_{j_i^m}$ from child node $j_i^m$
18:                CalcTablesWithIdentifier($\mathfrak{D}^i_{j_i^m}$)  ▷ Calculates the other rows of $T_i$ using (4.9) by identifying the same forwarding partition with the same unique identifier
19:                $\mathfrak{C}_i \leftarrow \mathfrak{C}_i \cup j_i^m$        ▷ Adds child node $j_i^m$ to the set of descendants $\mathfrak{C}_i$
20:            end for
21:        end if
22:        for each $p_i^{\mathfrak{n}} \in \mathfrak{P}_i$ do
23:            $\mathfrak{L}_i \leftarrow$ PartitionPossibleForwarding($\mathfrak{C}_i$)   ▷ Partitions the possible forwardings using a mapping function that decides the direction of each bundle, $u_i^j$, from one of its descendants in $\mathfrak{C}_i$
24:            $\mathfrak{D}_i^{p_i^{\mathfrak{n}}} \leftarrow$ CalcMetaDataArray($\mathfrak{L}_i$) ▷ Calculates the other rows of $U_i^{p_i^{\mathfrak{n}}}$ using (4.12) to forms its own $\mathfrak{D}_i^{p_i^{\mathfrak{n}}}$ meta-data for parent node $p_i^{\mathfrak{n}}$
25:            SendMetaData($p_i^{\mathfrak{n}}$, $\mathfrak{D}_i^{p_i^{\mathfrak{n}}}$)       ▷ Sends $\mathfrak{D}_i^{p_i^{\mathfrak{n}}}$ to parent node $p_i^{\mathfrak{n}}$
26:        end for
27:        $Cmax_I \leftarrow$ WaitControlMessage($p_i^{\mathfrak{n}}$)   ▷ Receives control message from parent node $p_i^{\mathfrak{n}}$ in $\mathfrak{P}_i$
28:        PropagateControlMessage($j_i^m$, $Cmax_I$)   ▷ Sends control message to each child node, $j_i^m \in J_i$
29:        PerformTransmitIncRouting($readings, Cmax_I$)
30:        SetNodeOptimalBehaviourIncRouting($Cmax_I$)   ▷ Sets node's optimal sensing, forwarding, and next-hop decisions
31:        SetTTime($tTime + tRate$)                ▷ Node sets its next transmitting time
32:        $readings \leftarrow \{\}$
33:    end if
34: end loop
```

Now, let $\mathfrak{P}_i$ denote the set of parent nodes (which are nodes with a one hop shorter shortest path to the base station) of node i and $\mathfrak{C}_i$ denote the set of its descendants. Thus, at each node $i \in I$, there are at most $|\mathfrak{P}_i|^{|\mathfrak{C}_i|+1}$ possibilities to forward the bundled data, where $|\mathfrak{P}_i|$ and $|\mathfrak{C}_i|$ are the sizes of $\mathfrak{P}_i$ and $\mathfrak{C}_i$ respectively. This is because each node has to forward $|\mathfrak{C}_i| + 1$ bundles through $|\mathfrak{P}_i|$ different shortest paths. Next, let $\mathfrak{L}_i$ denote the set of these possibilities (with $|\mathfrak{L}_i| = |\mathfrak{P}_i|^{|\mathfrak{C}_i|+1}$) and each $l_i^t \in \mathfrak{L}_i$, a possible partition of forwarding at node i. That is, $l_i^t = \left[F\left(u_i^1\right), \ldots, F\left(u_i^{|\mathfrak{C}_i|+1}\right)\right]$ where u_i^j is the j^{th} bundle that might arrive at node i from one of its descendants, $F\left(u_i^j\right)$ is a mapping function that decides the forwarding direction (or path) for this particular bundle, and $u_i^{|\mathfrak{C}_i|+1}$ is node i's own bundle of samples.

Given this, our algorithm with flexible routing is similar to that with fixed routing, and as before, it runs in three phases (see algorithm 4). The first, in which the parent nodes send their information regarding $B_{p_i^{\mathfrak{n}}}$ and $E^{\mathrm{F}}_{p_i^{\mathfrak{n}}}$ to their child nodes, is identical (see lines 7-13). There are, however, slight modifications in the next phase. These modifications are needed to keep track of all the possible partitions of forwarding for nodes which have

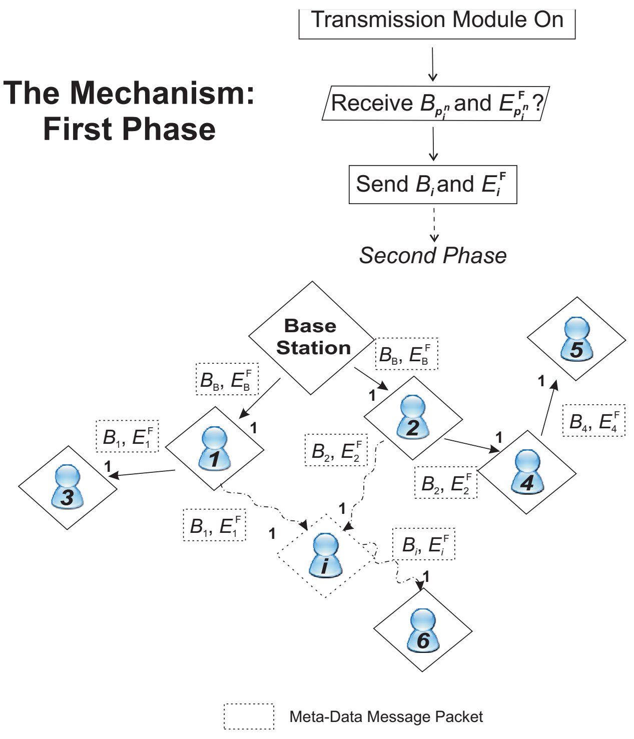


FIGURE 4.10: The first phase of the algorithm with flexible routing.

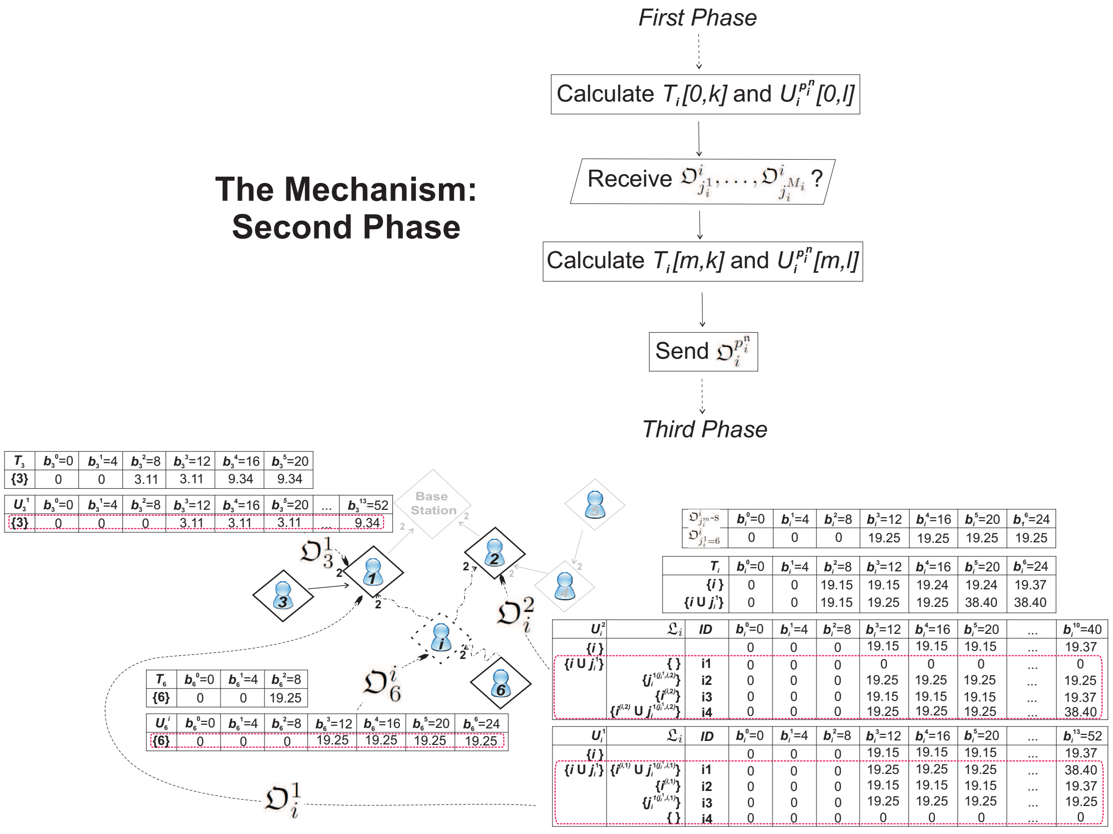
more than one shortest path routes to the base station.

In more detail, in the second phase, instead of sending one $\mathfrak{O}_i$ to a unique parent (as in the case of tree-structured networks), here, each node i has to calculate all the $\mathfrak{O}_i^{p_i^{\mathfrak{n}}}\left(l_i^t\right)$ meta-data arrays for each $l_i^t \in \mathfrak{L}_i$ partition of forwarding for each $p_i^{\mathfrak{n}} \in \mathfrak{P}_i$ (see lines 23-25). Specifically, this is achieved by first calculating the T_i table as we did for the first algorithm (line 17). In this case, however, we join each of the arriving $\mathfrak{O}^i_{j_i^m}\left(l^t_{j_i^m}\right)$ from its children $j_i^1, \ldots, j_i^{M_i}$ with those that belong to the same forwarding partition with the same unique identifier (line 18). The unique identifier is formed and attached to a particular partition of forwarding when there are more than one possible routes to forward to (line 23). As in figure 4.9, a feasible unique identifier could be the index of $l^t_{j_i^m}$. Next, we calculate $U_i^{p_i^{\mathfrak{n}}}$ tables for each $p_i^{\mathfrak{n}} \in \mathfrak{P}_i$ as in the first algorithm (line 24).

The rest of the second phase and third phase remain the same as that of the algorithm with fixed routing described previously (see lines 27-30).

4.4.4 Illustrative Example

To illustrate how the second algorithm works in practice, reconsider the scenario described in figure 4.3(b). Each of nodes i, 1, 2, 3, and 6 is set with s_i, s_1, s_2, s_3, and s_6 different numbers of sampling actions and thus, C_i, C_1, C_2, C_3, and C_6 sets of sampling actions (or rates) respectively, as one of their private values. The nodes also have their corresponding $\mathfrak{F}$-s regarding the information content of the samples that they acquire


First Phase
Calculate $T_i[0,k]$ and $U_i^{p_i^n}[0,l]$
Receive $\mathfrak{D}^i_{j_i^1}, \ldots, \mathfrak{D}^i_{j_i^{M_i}}$?
Calculate $T_i[m,k]$ and $U_i^{p_i^n}[m,l]$
Send $\mathfrak{D}_i^{p_i^n}$
Third Phase
The Mechanism: Second Phase
Base Station
1
2
3
4
5
6
i
$\mathfrak{D}_3^1$
$\mathfrak{D}_6^i$
$\mathfrak{D}_i^1$
$\mathfrak{D}_i^2$

T_3	$b_3^0=0$	$b_3^1=4$	$b_3^2=8$	$b_3^3=12$	$b_3^4=16$	$b_3^5=20$
{3}	0	0	3.11	3.11	9.34	9.34

U_3^1	$b_3^0=0$	$b_3^1=4$	$b_3^2=8$	$b_3^3=12$	$b_3^4=16$	$b_3^5=20$	...	$b_3^{13}=52$
{3}	0	0	0	3.11	3.11	3.11	...	9.34

T_6	$b_6^0=0$	$b_6^1=4$	$b_6^2=8$
{6}	0	0	19.25

U_6^i	$b_6^0=0$	$b_6^1=4$	$b_6^2=8$	$b_6^3=12$	$b_6^4=16$	$b_6^5=20$	$b_6^6=24$
{6}	0	0	0	19.25	19.25	19.25	19.25

$\mathfrak{D}^i_{j_i^m}$-s	$b_i^0=0$	$b_i^1=4$	$b_i^2=8$	$b_i^3=12$	$b_i^4=16$	$b_i^5=20$	$b_1^6=24$
$\mathfrak{D}^i_{j_i^1=6}$	0	0	0	19.25	19.25	19.25	19.25

T_i	$b_i^0=0$	$b_i^1=4$	$b_i^2=8$	$b_i^3=12$	$b_i^4=16$	$b_i^5=20$	$b_i^6=24$
{i}	0	0	19.15	19.15	19.24	19.24	19.37
{i U j_i^1}	0	0	19.15	19.25	19.25	38.40	38.40

U_i^2	$\mathfrak{L}_i$	ID	$b_i^0=0$	$b_i^1=4$	$b_i^2=8$	$b_i^3=12$	$b_i^4=16$	$b_i^5=20$	...	$b_i^{10}=40$
{i}			0	0	0	19.15	19.15	19.15	...	19.37
{i U j_i^1}	{ }	i1	0	0	0	0	0	0	...	0
	{$j_i^{1(j_i^1,i,2)}$}	i2	0	0	0	19.25	19.25	19.25	...	19.25
	{$i^{(i,2)}$}	i3	0	0	0	19.15	19.15	19.15	...	19.37
	{$i^{(i,2)}$ U $j_i^{1(j_i^1,i,2)}$}	i4	0	0	0	19.25	19.25	19.25	...	38.40

U_i^1	$\mathfrak{L}_i$	ID	$b_i^0=0$	$b_i^1=4$	$b_i^2=8$	$b_i^3=12$	$b_i^4=16$	$b_i^5=20$	...	$b_i^{13}=52$
{i}			0	0	0	19.15	19.15	19.15	...	19.37
{i U j_i^1}	{$i^{(i,1)}$ U $j_i^{1(j_i^1,i,1)}$}	i1	0	0	0	19.25	19.25	19.25	...	38.40
	{$i^{(i,1)}$}	i2	0	0	0	19.15	19.15	19.15	...	19.37
	{$j_i^{1(j_i^1,i,1)}$}	i3	0	0	0	19.25	19.25	19.25	...	19.25
	{ }	i4	0	0	0	0	0	0	...	0

FIGURE 4.11: The second phase of the algorithm with fixed routing where nodes 3 and 6 correspondingly send $\mathfrak{D}_3^1$ and $\mathfrak{D}_6^i$ to their respective parent nodes 1 and i. Node i subsequently sends $\mathfrak{D}_i^1$ and $\mathfrak{D}_i^2$ to its parent nodes 1 and 2 correspondingly. $\mathfrak{D}$ meta-data arrays are represented by dotted rectangles. Tables are enlarged in the next figure.

(which are chosen arbitrarily for illustrative purposes) and energy budgets B-s to perform either sensing and/or forwarding. Other nodes (4 and 5) do not sample in this case.

In the first phase (as shown in figure 4.10), the base station initiates the meta-data message propagation. We remark that node i receives two meta-data messages (one from node 1 and another from node 2). The leaf nodes (including nodes 3, 5, and 6) eventually receive the meta-data message from their respective parents.

In the second phase, the leaf nodes 3 and 6 calculate their corresponding tables in a similar way as they do with the algorithm for fixed routing (as both nodes only have one parent). As can be seen in figure 4.11, node 3 sends $\mathfrak{D}_3^1$ to its parent node 1 while node 6 sends its own to node i. After receiving $\mathfrak{D}^i_{j_i^1=6}$ (which contains $Vmax^k_{j_i^1=6}$ values) from child nodes $j_i^1 = 6$ (see table 4.2(a)), node i can now calculate its T_i, U_i^1, and U_i^2 tables. The U_i tables are for its parent nodes 1 and 2 respectively. The elements of T_i's first and second rows (as shown in table 4.2(b)) are evaluated using equations 4.8 (by taking into account only node i's sensed data) and 4.9 (by also incorporating node 6's forwarded data) respectively.

TABLE 4.2: After receiving (a) $\mathfrak{O}^i_{j_i^1=6}$ meta-data array from its child nodes $j_i^1 = 6$, node i evaluates its (b) T_i and (c) U_i tables.

$\mathfrak{O}^i_{j_i^m}$-s	b_i^0=0	b_i^1=4	b_i^2=8	b_i^3=12	b_i^4=16	b_i^5=20	b_i^6=24
$\mathfrak{O}^i_{j_i^1=6}$	0	0	0	19.25	19.25	19.25	19.25

(a)

T_i	b_i^0=0	b_i^1=4	b_i^2=8	b_i^3=12	b_i^4=16	b_i^5=20	b_i^6=24
$\{i\}$	0	0	19.15	19.15	19.24	19.24	19.37
$\{i \cup j_i^1\}$	0	0	19.15	19.25	19.25	38.40	38.40

(b)

U_i^2	$\mathfrak{L}_i$	*ID*	b_i^0=0	b_i^1=4	b_i^2=8	b_i^3=12	b_i^4=16	b_i^5=20	...	b_i^{10}=40
$\{i\}$			0	0	0	19.15	19.15	19.15	...	19.37
$\{i \cup j_i^1\}$	$\{\}$	i1	0	0	0	0	0	0	...	0
	$\{j_i^{1(j_i^1,i,2)}\}$	i2	0	0	0	19.25	19.25	19.25	...	19.25
	$\{i^{(i,2)}\}$	i3	0	0	0	19.15	19.15	19.15	...	19.37
	$\{i^{(i,2)} \cup j_i^{1(j_i^1,i,2)}\}$	i4	0	0	0	19.25	19.25	19.25	...	38.40

U_i^1	$\mathfrak{L}_i$	*ID*	b_i^0=0	b_i^1=4	b_i^2=8	b_i^3=12	b_i^4=16	b_i^5=20	...	b_i^{13}=52
$\{i\}$			0	0	0	19.15	19.15	19.15	...	19.37
$\{i \cup j_i^1\}$	$\{i^{(i,1)} \cup j_i^{1(j_i^1,i,1)}\}$	i1	0	0	0	19.25	19.25	19.25	...	38.40
	$\{i^{(i,1)}\}$	i2	0	0	0	19.15	19.15	19.15	...	19.37
	$\{j_i^{1(j_i^1,i,1)}\}$	i3	0	0	0	19.25	19.25	19.25	...	19.25
	$\{\}$	i4	0	0	0	0	0	0	...	0

(c)

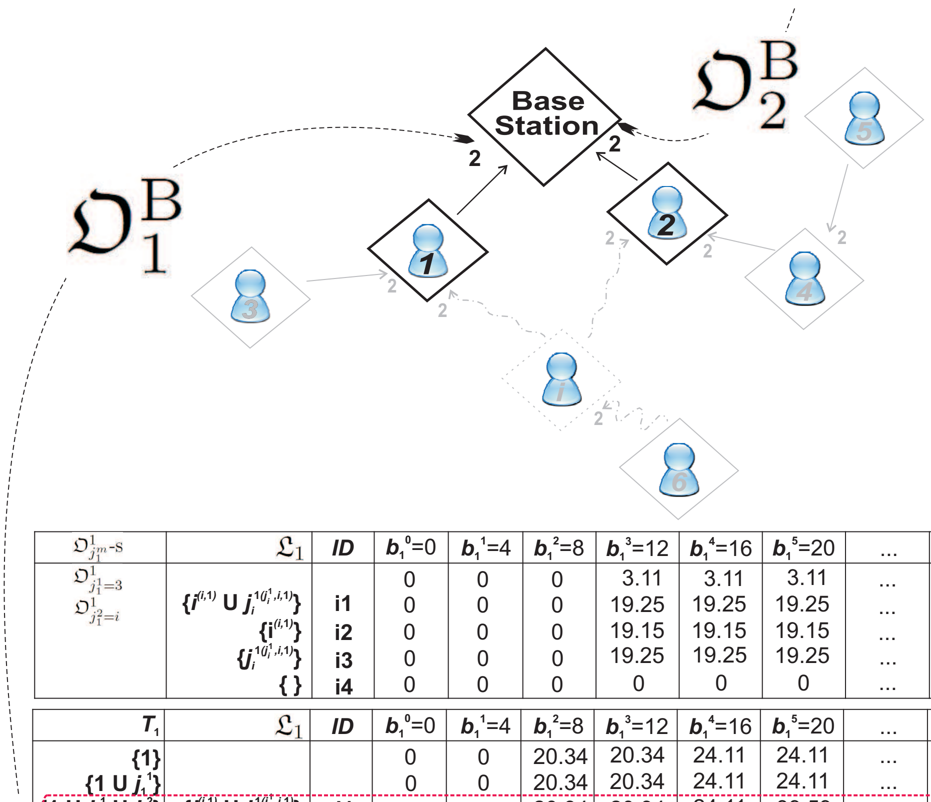
Node i has two potential routes for forwarding node 6's and its own data and, hence, there are four ($|\mathfrak{L}_i| = 2^2$) possible partitions of forwarding for each U_i table; namely:

- Node i routes both node 6's and its own data through node 1 (where $\mathfrak{L}_i = \{i^{(i,1)} \cup j_i^{1(j_i^1,i,1)}\}$ in U_i^1 and $\mathfrak{L}_i = \{\}$ in U_i^2),
- Node i routes only its data through node 1 while node 6's is forwarded towards node 2 (where $\mathfrak{L}_i = \{i^{(i,1)}\}$ in U_i^1 and $\mathfrak{L}_i = \{j_i^{1(j_i^1,i,2)}\}$ in U_i^2),
- Node i routes its data through node 2 but sends node 6's through node 1 (where $\mathfrak{L}_i = \{j_i^{1(j_i^1,i,1)}\}$ in U_i^1 and $\mathfrak{L}_i = \{i^{(i,2)}\}$ in U_i^2), and
- Node i routes both node 6's and its own data through node 2 (where $\mathfrak{L}_i = \{\}$ in U_i^1 and $\mathfrak{L}_i = \{i^{(i,2)} \cup j_i^{1(j_i^1,i,2)}\}$ in U_i^2).

As shown in table 4.2(c) where $row = \{i \cup j_i^1\}$, for each U_i table, there are four l_i-s, each of which represents a possible partition of forwarding at node i, together with the same unique identifier (in *column ID*) attached to it. We will later see how this identifier is useful.

$\mathfrak{D}^2_{j_2^m}$-s	$\mathfrak{L}_2$	ID	$b_2^0=0$	$b_2^1=4$	$b_2^2=8$	$b_2^3=12$	$b_2^4=16$	$b_2^5=20$	...	$b_2^{10}=40$
$\mathfrak{D}^2_{j_2^1=i}$	{ }	i1	0	0	0	0	0	0	...	0
	$\{j_i^{1(j_i^1,i,2)}\}$	i2	0	0	0	19.25	19.25	19.25	...	19.25
	$\{i^{(i,2)}\}$	i3	0	0	0	19.15	19.15	19.15	...	19.37
	$\{i^{(i,2)} \cup j_i^{1(j_i^1,i,2)}\}$	i4	0	0	0	19.25	19.25	19.25	...	38.40

T_2	$\mathfrak{L}_2$	ID	$b_2^0=0$	$b_2^1=4$	$b_2^2=8$	$b_2^3=12$	$b_2^4=16$	$b_2^5=20$	...	$b_2^{10}=40$
{2}			0	0	10.55	10.55	13.21	13.21	...	23.12
$\{2 \cup j_2^1\}$	{ }	i1	0	0	10.55	10.55	13.21	13.21	...	23.12
	$\{j_i^{1(j_i^1,i,2)}\}$	i2	0	0	10.55	19.25	19.25	29.80	...	35.33
	$\{i^{(i,2)}\}$	i3	0	0	10.55	19.15	19.15	29.70	...	35.23
	$\{i^{(i,2)} \cup j_i^{1(j_i^1,i,2)}\}$	i4	0	0	10.55	19.25	19.25	29.80	...	51.61



$\mathfrak{D}^1_{j_1^m}$-s	$\mathfrak{L}_1$	ID	$b_1^0=0$	$b_1^1=4$	$b_1^2=8$	$b_1^3=12$	$b_1^4=16$	$b_1^5=20$	...	$b_1^{13}=52$
$\mathfrak{D}^1_{j_1^1=3}$			0	0	0	3.11	3.11	3.11	...	9.34
$\mathfrak{D}^1_{j_1^2=i}$	$\{i^{(i,1)} \cup j_i^{1(j_i^1,i,1)}\}$	i1	0	0	0	19.25	19.25	19.25	...	38.40
	$\{i^{(i,1)}\}$	i2	0	0	0	19.15	19.15	19.15	...	19.37
	$\{j_i^{1(j_i^1,i,1)}\}$	i3	0	0	0	19.25	19.25	19.25	...	19.25
	{ }	i4	0	0	0	0	0	0	...	0

T_1	$\mathfrak{L}_1$	ID	$b_1^0=0$	$b_1^1=4$	$b_1^2=8$	$b_1^3=12$	$b_1^4=16$	$b_1^5=20$	...	$b_1^{13}=52$
{1}			0	0	20.34	20.34	24.11	24.11	...	32.12
$\{1 \cup j_1^1\}$			0	0	20.34	20.34	24.11	24.11	...	33.45
$\{1 \cup j_1^1 \cup j_1^2\}$	$\{i^{(i,1)} \cup j_i^{1(j_i^1,i,1)}\}$	i1	0	0	20.34	20.34	24.11	39.59	...	65.62
	$\{i^{(i,1)}\}$	i2	0	0	20.34	20.34	24.11	39.49	...	52.60
	$\{j_i^{1(j_i^1,i,1)}\}$	i3	0	0	20.34	20.34	24.11	39.59	...	52.70
	{ }	i4	0	0	20.34	20.34	24.11	24.11	...	33.45

FIGURE 4.12: Nodes 1 and 2 correspondingly send $\mathfrak{D}_1^{\mathrm{B}}$ and $\mathfrak{D}_2^{\mathrm{B}}$ to the base station.

After receiving $\mathfrak{D}^1_{j_1^1=3}$ and $\mathfrak{D}^1_{j_1^2=i}$ (which contain $Vmax^k_{j_1^1=3}$ and $Vmax^k_{j_1^2=i}$ values) from child nodes $j_1^1 = 3$ and $j_1^2 = i$ respectively, node 1 can now evaluate its T_1 table, as shown in figure 4.12. Node 1's $\mathfrak{D}_1^{\mathrm{B}}$ is then sent to the base station (B stands for Base station). Node 2 also sends its own $\mathfrak{D}_2^{\mathrm{B}}$ to the base station after receiving $\mathfrak{D}^2_{j_2^1=i}$ from node i and evaluating its T_2. In this case, both nodes 1 and 2 are not required to evaluate their U tables because their parent node, the base station, has sufficient energy to receive any transmitted data.

The base station eventually receives the meta-data arrays $\mathfrak{D}^{\mathrm{B}}_{j_{\mathrm{B}}^1=1}$ and $\mathfrak{D}^{\mathrm{B}}_{j_{\mathrm{B}}^2=2}$ (see figure 4.13). Now, as each corresponding row of the arriving $\mathfrak{D}^{\mathrm{B}}_{j_{\mathrm{B}}^m}$-s contains the same ID,

$\mathfrak{O}^{B}_{j^m_B}$-s	$\mathfrak{L}_B$	ID	$b_2^0=0$	$b_2^1=4$	$b_2^2=8$	$b_2^3=12$	$b_2^4=16$	$b_2^5=20$	...	$b_2^{10}=40$
$\mathfrak{O}^{B}_{j^2_B=2}$	$\{\}$	i1	0	0	10.55	10.55	13.21	13.21	...	23.12
	$\{j_i^{1(j_i^1,i,2)}\}$	i2	0	0	10.55	19.25	19.25	29.80	...	35.33
	$\{i^{(i,2)}\}$	i3	0	0	10.55	19.15	19.15	29.70	...	35.23
	$\{i^{(i,2)} \cup j_i^{1(j_i^1,i,2)}\}$	i4	0	0	10.55	19.25	19.25	29.80	...	51.61

$\mathfrak{O}^{B}_{j^m_B}$-s	$\mathfrak{L}_B$	ID	$b_1^0=0$	$b_1^1=4$	$b_1^2=8$	$b_1^3=12$	$b_1^4=16$	$b_1^5=20$	...	$b_1^{13}=52$
$\mathfrak{O}^{B}_{j^1_B=1}$	$\{i^{(i,1)} \cup j_i^{1(j_i^1,i,1)}\}$	i1	0	0	20.34	20.34	24.11	39.59	...	65.62
	$\{i^{(i,1)}\}$	i2	0	0	20.34	20.34	24.11	39.49	...	52.60
	$\{j_i^{1(j_i^1,i,1)}\}$	i3	0	0	20.34	20.34	24.11	39.59	...	52.70
	$\{\}$	i4	0	0	20.34	20.34	24.11	24.11	...	33.45

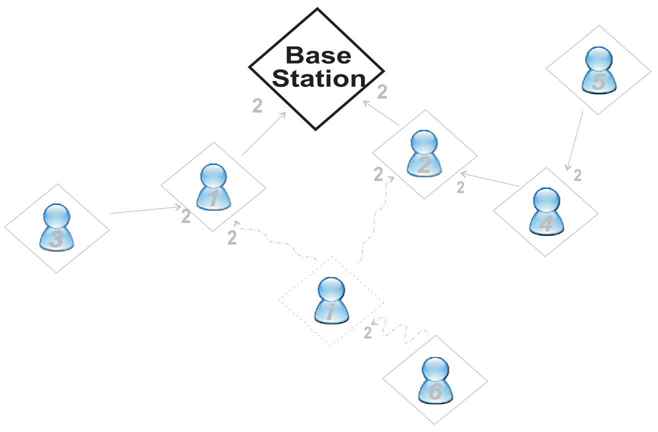


FIGURE 4.13: The base station receives the $\mathfrak{O}^{B}_{1}$ and $\mathfrak{O}^{B}_{2}$ meta-data arrays and evaluates the optimal sensing, forwarding, and routing actions that yield the highest information value collected.

the base station combines each pair of these rows as one partition. For instance, where both $\mathfrak{L}_B = \{i^{(i,1)} \cup j_i^{1(j_i^1,i,1)}\}$ in the $\mathfrak{O}^{B}_{j^1_B=1}$ table and $\mathfrak{L}_B = \{\}$ in the $\mathfrak{O}^{B}_{j^2_B=2}$ table have the same unique identifier, $i1$, the total information value that could potentially be collected with this particular sensing and routing behaviour (i.e. partition) is 88.74 ($Vmax^{13}_{j^1_B=1} + Vmax^{10}_{j^2_B=2} = 65.62 + 23.12$). This is achieved when both nodes i and 6 sample once. Node i then routes both node 6's and its own data through node 1 which samples just twice itself. Nodes 2 and 3 sample five times and twice correspondingly. Node 1 also forwards two of node 3's collected samples. Other possible partitions result in lower information value gathered.

The base station starts the third phase by propagating control message $Cmax_I$ down the network in a similar manner as it does for the algorithm with fixed routing.

4.5 Empirical Evaluation

Having described the algorithms with fixed and flexible routing, we now seek to evaluate their performance and effectiveness when applied to typical WVSNs whose communication networks exhibit loops. Our goal in this empirical evaluation is to quantify the performance of the algorithms in terms of the quantity of information that they deliver to the base station, and the communication and computational costs of the coordination. We expect the algorithm with flexible routing to deliver more information, but make greater demands of computation and communication resources (because of the large number of alternative routes for the data that it must evaluate). However, given that the algorithm with fixed routing can always be applied in this setting, by ignoring the fact that there exist alternative routing options and just making an arbitrary choice, we are interested in the trade-off between the loss in information and the saving in resources that results. We first describe the extended simulator from the previous chapter (in section 3.5.1). We then detail the experimental setup and go on to the actual evaluation.

4.5.1 The Simulator Extension

This section details the extensions that are needed to the previously defined domain models upon which DC-WSNS is built. In the previous chapter, each node within the simulated network is pre-initialized with a static routing table that it uses to route its collected samples in a multi-hop fashion to the base station. However, in most WSN deployments, this node with a fixed and non-adaptive routing scheme is undesirable, particularly if deployed in environments that are highly dynamic. The nodes therefore need to be designed to *self-configure* at the deployment stage (e.g. in finding their own shortest routes to the base station or synchronizing their clocks), and to *self-adapt* whenever the state (or topology) of the network changes (e.g. in finding alternative routes if their shortest paths cease or updating them if a shorter shortest path is established). To this end, we extend the domain models of the simulator to include the enhanced message exchange (or communication) and the newly built node models. We know consider each of these in turn. The other models remain the same.

4.5.1.1 The Message Exchange/Communication Model

As described in section 4.4, we distinguish three types of messages being exchanged by the nodes under our coordination mechanism. However, in our simulator, we introduce an additional type of message termed BGP-typed messages. These are exchanged for distributed network routing discovery purposes at the network's deployment stage or whenever the topology of the network experiences any changes (e.g. due to the addition, removal, failure, or battery depletion of nodes).

This type of message is categorized into four kinds; namely (i) BGP-OPEN, (ii) BGP-UPDATE, (iii) BGP-KEEPALIVE, and (iv) BGP-NOTIFICATION messages (or packets). The OPEN message is used by a node to establish a BGP TCP/IP-based connection (or session) with its neighbour node. Once established, the node can then send or receive any KEEPALIVE (that is a handshake message, sent at regular intervals, which the node uses to inform and keep the session alive), UPDATE (that is a message the node uses to announce new, withdrawal, or any changes of routes), or NOTIFICATION (that is a message used to shut down the established session) messages to or from its session-connected node.

In some WSN scenarios, the limited bandwidth available to nodes in the network might be an interesting issue to explore. In our case, however, we assume that message packets, regardless of their size, require the same amount of time to transmit. Moreover, we do not consider the dropping or corruption of BGP-typed, meta-data, or control message packets and, hence, assume that they are always transferred successfully to their destination nodes. There exists no communication interference within the transmission range of nearby nodes. And we still assume that there is no difference between overhearing and receiving in terms of power consumption. Moreover, the nodes are still pre-programmed to ignore and drop packets that are not destined to them. Therefore, we acknowledge that the message exchange model implemented here is somewhat idealised as it assumes that some portion of a node's energy is consumed only when it either transmits or receives a message[8]. Incorporating a more realistic communication model into our simulator will be part of our future work (see section 5.2 for more details).

4.5.1.2 The Node Model

Each node in the network is installed with a distributed, scalable, and robust BGP-based routing discovery protocol and the two algorithms described in the previous section. The BGP routing discovery works by maintaining a table of IP networks. We therefore assume that the node possesses sufficient memory to store its growing routing table. To discover failing nodes, the protocol implements an *acknowledgement* mechanism, in which recipient nodes acknowledge the originator of any messages they receive.

In more detail, the node uses a simple *finite state machine* that consists of three main states; namely *Idle*, *OpenSent*, and *Established*, for self-configuring purposes (see figure 4.14). In the Idle state, the node is listening to a TCP/IP connection from any of its neighbourhood nodes. In the OpenSent state, it sends a BGP-OPEN message and listens for one in return from its session-connected node. In the Established state, it listens for a BGP-KEEPALIVE message from its peer and if no message is received after a certain period of time (according to the *time-out* or *acknowledgement* mechanism), it transitions back to the IDLE state. After the Established state, the node sends a

[8]We remark that these assumptions and limitations are the same as those found in the domain models of the simulator defined in the previous chapter. However, rather than manually creating the routing table of each node at system start-up time, we here use an adaptive, scalable, and robust distributed BGP-based network routing discovery protocol.

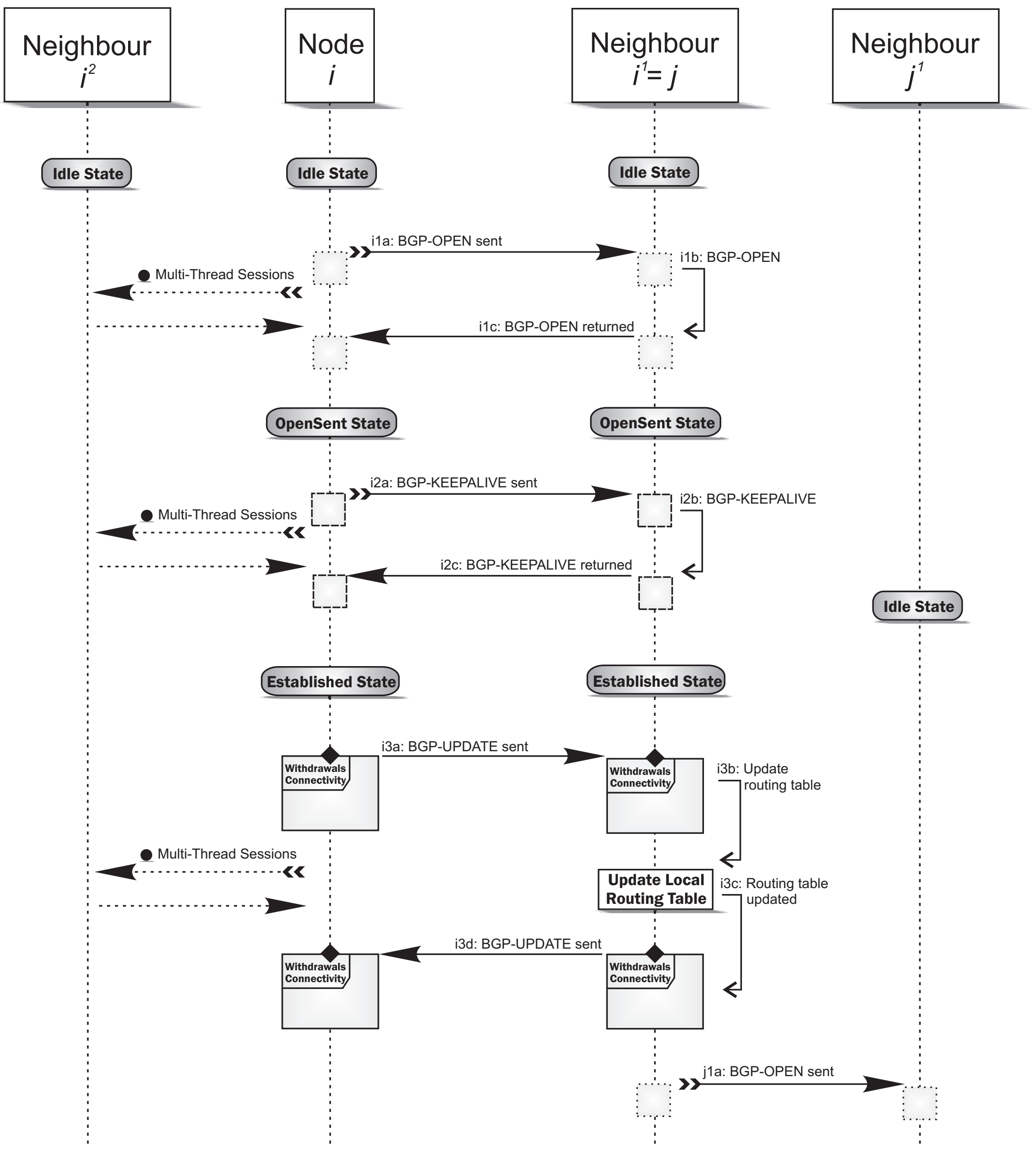
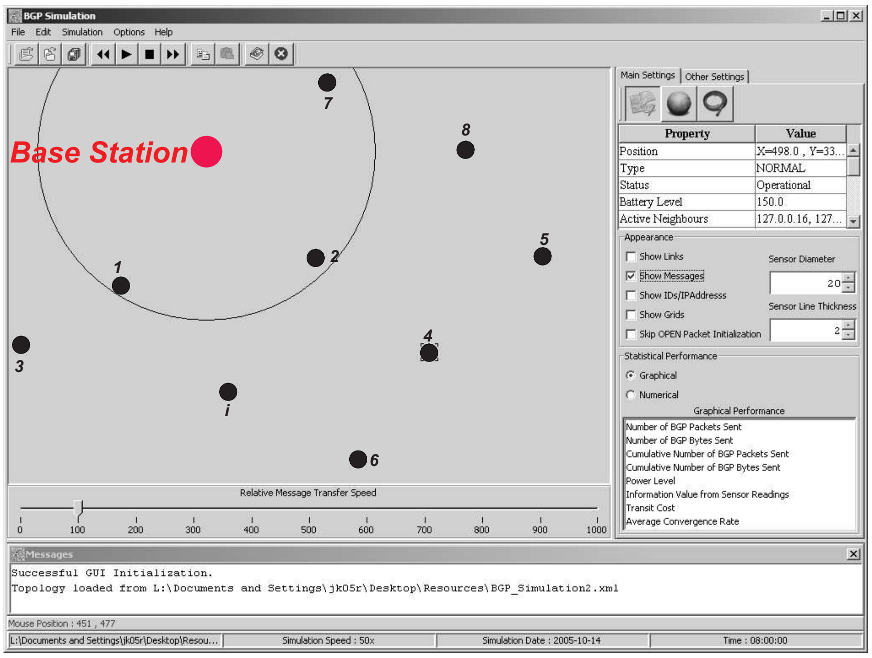


FIGURE 4.14: The sequence diagram of BGP-based routing discovery protocol. Node i initiates a session to notify its neighbour node, i^1, regarding its routing table updates.

BGP-UPDATE message about new destination nodes to which it offers reachability and withdrawals to which the destination no longer offers connectivity.

As shown in figures 4.15 and 4.18(a), at a *network deployment* phase, the nodes are not aware of the network's topology as there is no central coordinator. The network thus enters a *network discovery* phase in which the base station broadcasts a BGP-OPEN message that is heard by the neighbourhood nodes within its transmission range. The recipient nodes are now aware of the base station's presence and, subsequently, update their routing tables by exchanging BGP-UPDATE messages with it. A successful routing table update will trigger each of the recipient nodes to broadcast a new BGP-OPEN message in order to notify its own neighbourhood nodes about the changes (see figure 4.16). The local computation of a single node can thus be viewed as consisting of a



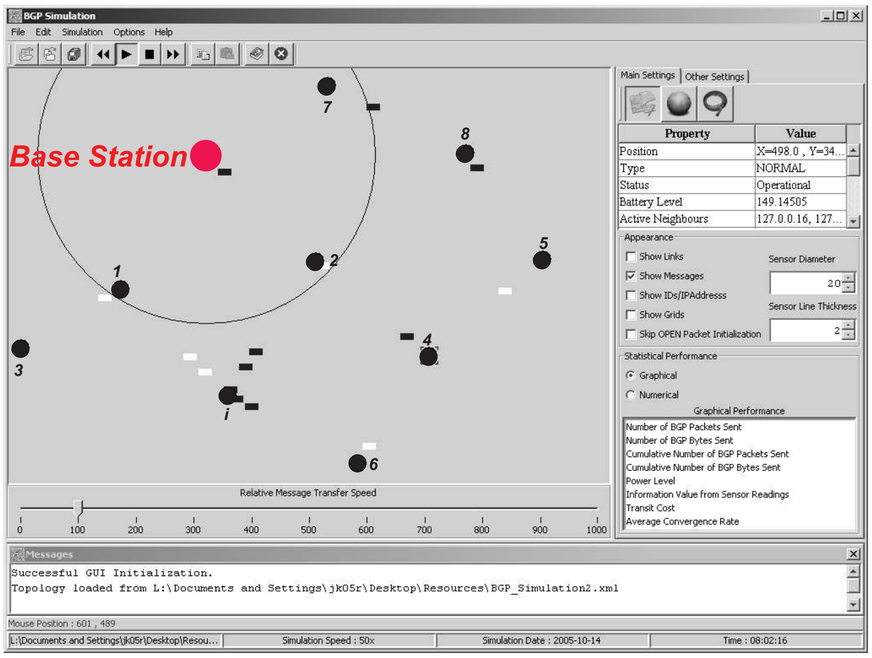
Property	Value
Position	X=498.0 , Y=33...
Type	NORMAL
Status	Operational
Battery Level	150.0
Active Neighbours	127.0.0.16, 127...

☐ Show Links
☑ Show Messages
☐ Show IDs/IPAddresss
☐ Show Grids
☐ Skip OPEN Packet Initialization
Sensor Diameter 20
Sensor Line Thickness 2
(● Graphical) ○ Numerical

```
Successful GUI Initialization.
Topology loaded from L:\Documents and Settings\jk05r\Desktop\Resources\BGP_Simulation2.xml
```

Mouse Position : 451 , 477
L:\Documents and Settings\jk05r\Desktop\Resou... | Simulation Speed : 50x | Simulation Date : 2005-10-14 | Time : 08:00:00

FIGURE 4.15: A randomly created and connected WVSN of 9 sensor nodes in a simulation. The unfilled circle represents the wireless transmission range of the base station. All other ordinary nodes are set with the same transmission range.



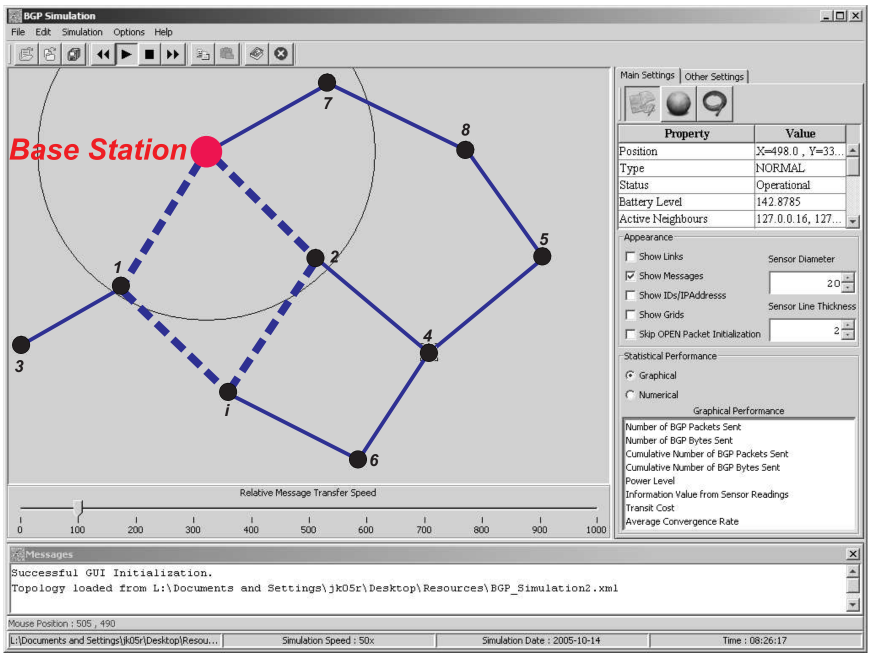
Property	Value
Position	X=498.0 , Y=34...
Type	NORMAL
Status	Operational
Battery Level	149.14505
Active Neighbours	127.0.0.16, 127...

☐ Show Links
☑ Show Messages
☐ Show IDs/IPAddresss
☐ Show Grids
☐ Skip OPEN Packet Initialization
Sensor Diameter 20
Sensor Line Thickness 2
(● Graphical) ○ Numerical

```
Successful GUI Initialization.
Topology loaded from L:\Documents and Settings\jk05r\Desktop\Resources\BGP_Simulation2.xml
```

Mouse Position : 601 , 489
L:\Documents and Settings\jk05r\Desktop\Resou... | Simulation Speed : 50x | Simulation Date : 2005-10-14 | Time : 08:02:16

FIGURE 4.16: The network enters its route discovery phase. The nodes exchange BGP-UPDATE messages to update their own routing tables. BGP messages are represented by filled rectangles (more specifically, the white and black ones are BGP-KEEPALIVE and BGP-UPDATE messages respectively). We note that, in the simulator's display panel, the message packets are visualised and directed towards their destinations.



Property	Value
Position	X=498.0 , Y=33...
Type	NORMAL
Status	Operational
Battery Level	142.8785
Active Neighbours	127.0.0.16, 127...

Appearance

- [ ] Show Links
- [✓] Show Messages
- [ ] Show IDs/IPAddresss
- [ ] Show Grids
- [ ] Skip OPEN Packet Initialization

Sensor Diameter: 20

Sensor Line Thickness: 2

Statistical Performance

[✓ Graphical] Numerical

Graphical Performance

- Number of BGP Packets Sent
- Number of BGP Bytes Sent
- Cumulative Number of BGP Packets Sent
- Cumulative Number of BGP Bytes Sent
- Power Level
- Information Value from Sensor Readings
- Transit Cost
- Average Convergence Rate

```
Successful GUI Initialization.
Topology loaded from L:\Documents and Settings\jk05r\Desktop\Resources\BGP_Simulation2.xml
```

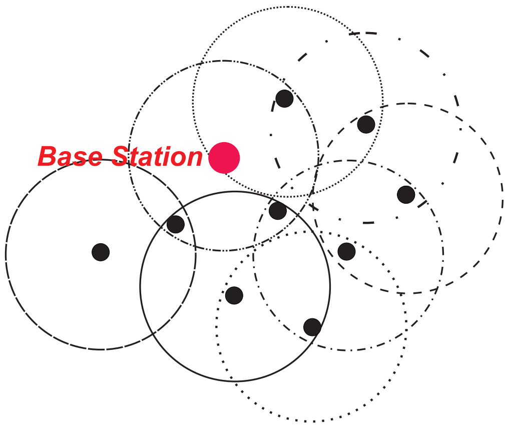
Mouse Position : 505 , 490

L:\Documents and Settings\jk05r\Desktop\Resou... | Simulation Speed : 50x | Simulation Date : 2005-10-14 | Time : 08:26:17

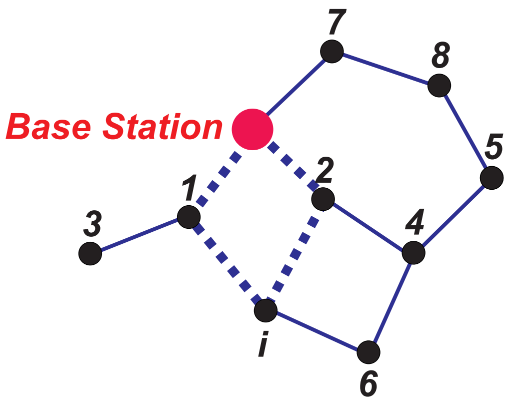
FIGURE 4.17: The network converges into a stable state and enters its coordination phase. In this particular case, node i has two shortest paths to the base station; through node 1 or 2 (both with 2 hops). The shortest paths are represented by dotted lines. Node i's alternative paths to the centre are represented by solid lines.

sequence of stages. Each stage consists of receiving routing tables from its neighbours, followed by local computation, followed (perhaps) by sending its own routing table to its own neighbourhood nodes (if its routing table changed).

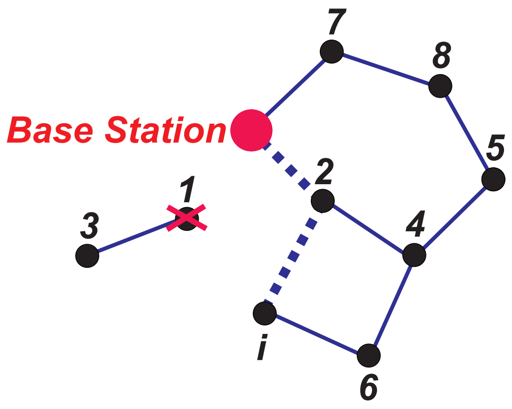
This sequence will stop when the network converges into a stable state and enters a *network coordination* phase (see figure 4.17), in which each node's routing table is synchronized with those of others and, thus, the node is aware of its surrounding nodes, as well as its fewest number of hops (i.e. shortest path routes) to the base station. The node will possibly have a number of shortest path routes (of which each has the same fewest number of hops) to the base station and, therefore, will not send data to a node that is further from the base station in term of hop count than itself (recall the discussion in section 4.4.3). As in figure 4.18(b), this means node i will not forward data through nodes 6, 4, and 2 nor through nodes 6, 4, 5, 8, and 7 to reach the base station. In the case where the shortest paths cease (e.g. due to the battery depletion in figure 4.18(c) or the removal in figure 4.18(d) of any intermediate node), the network reenters the route discovery phase in order to find out the new shortest paths that potentially have a bigger number of hops to the base station. Similarly, in the case where a new shortest path is found (e.g. due to the addition of an intermediate node as in figure 4.18(e)), the newly added node will first broadcast a BGP-OPEN message notifying its surrounding regarding its presence.



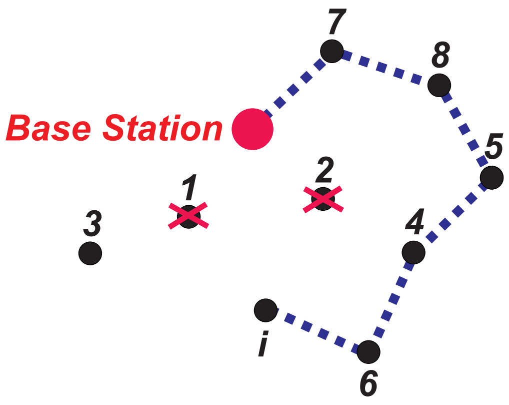
(a) The same WVSN as that created in figure 4.15. A 2-D spherical radio transmission range model is used while assuming there exists no obstruction between nodes in the network.



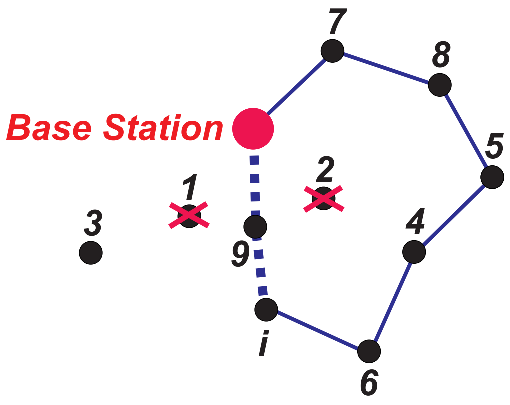
(b) At the end of a network routing discovery phase, nodes are aware of their shortest path routes. Node i has two shortest paths to the base station; through node 1 or 2 (both with 2 hops). The shortest paths are represented by dotted lines.



(c) Node 1 depletes its battery. Node i removes node 1 from its routing table and, hence, broadcasts a BGP-OPEN message notifying its neighbour about the changes. Node 3 is isolated and, therefore, unaware of these.



(d) Node 2 is removed. Node i's shortest paths is now through nodes 6, 4, 5, 8, and 7.



(e) Node 9 is added into the network. It broadcasts a BGP-OPEN message informing its presence. Node i establishes its new shortest route to the base station.

FIGURE 4.18: Network discovery phase using BGP-based TCP/IP protocol.

4.5.2 Network and Parameters Setup

In our experiments, we create instances of a WVSN by randomly deploying the nodes within a unit square (i.e. x and y coordinates within the square are randomly drawn from a uniform distribution with support $[0, 1]$) and connecting them according to a randomly determined radio transmission range (extending this range as necessary to ensure that there are no unconnected nodes). Each resulting WVSN exhibits a loopy communication network such that for each node there are multiple alternative routes to the base station.

We consider twenty different sampling actions for each of the nodes such that the possible sampling rates, C_i, of each node are initialized to $C_i = \{1, \dots, 20\}$. The corresponding information content $v_i^{c_i^j}$ for each of the $c_i^j \in C_i$ samples is generated using the generic information metric (defined in section 4.3), with the factor, α_i, randomly drawn from a uniform distribution with support $[0, 1]$.

The energy budget of each node is randomly generated (drawn from the same distribution) with a predetermined maximum value which ensures that the network as a whole is energy constrained, and no node is able to sample and transmit at its maximum rates. We scale this predetermined maximum value with the number of nodes in the network since larger networks require that nodes forward data for a larger number of other nodes. We assume that each real valued number inside a coordination message (e.g. the value of node i's energy budget, B_i, or its optimal sampling rate, c_i^j) occupies 4 bytes of communication cost, and the energy consumption of each node for sensing and forwarding a sample is fixed throughout the entire experiment.

For each instance of the created WVSN, the network first enters a route discovery phase (as explained in the previous section). At the end of this (i.e. when the network is in a coordination phase), each node has a number of shortest path routes (of which each has the same fewest number of hops) to the base station. The task of each node is thus to find both its optimal sensing and forwarding actions, as well as the most energy efficient route from the collection of these shortest paths in order to maximise the total value of information gathered at the base station, under its energy constraints. This is when we then execute each of our newly developed algorithms as described below:

- **Algorithm with Flexible Routing**
  We apply our algorithm with flexible routing just once, directly on the loopy communication network of the WVSN (see figure 4.19(a) for an exemplar scenario), such that it determines both the optimal sensing and forwarding actions, as well as the routes.

- **Algorithm with Fixed Routing**
  Prior to applying our algorithm with fixed routing, we allow each node to make

an arbitrary choice of the route that its data (and any data that it forwards for other nodes) will take toward the base station. This effectively turns the loopy communication network into a tree-structured one, with each node effectively selecting their parent in the tree (see figure 4.19(b)). We then apply our algorithm with fixed routing to calculate the optimal sensing and forwarding decisions of each node. For each instance of the WVSN, we repeat this process 100 times, averaging over the unique instances of trees that result.

We perform repeated experiments by creating 100 instances of the WVSN with 6, 9, 12, 15, ..., 60 nodes. The algorithm with flexible routing, however, is only run with up to 21 nodes. This is due to the insufficient memory allocated for the java heap space that is used to keep track all possible partitions of forwarding which grows exponentially with the number of nodes and potential routes.

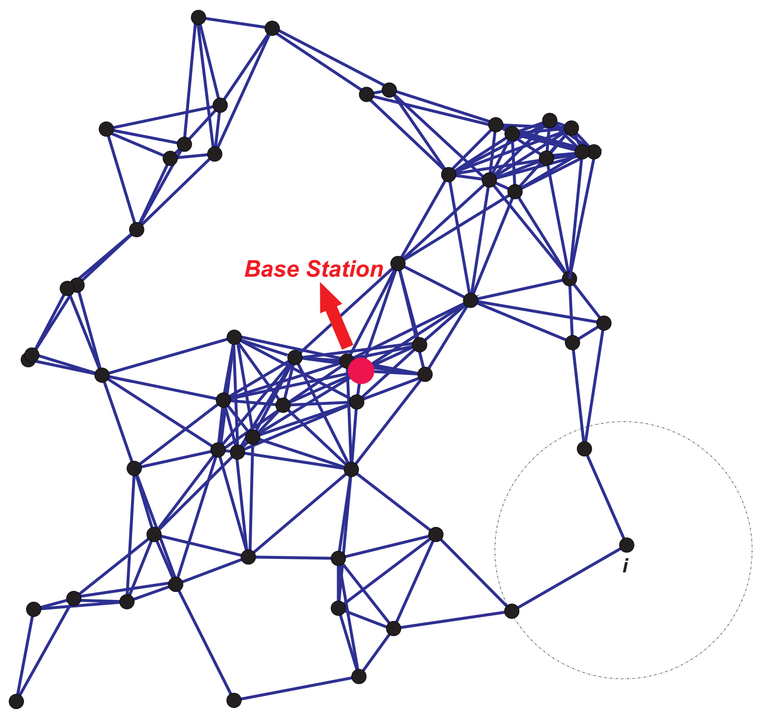
4.5.3 Benchmark Algorithms

In this experiment, we also benchmark our two algorithms against a uniform non-adaptive algorithm with fixed routing. This algorithm dictates that each sensor $i \in I$ in the network should simply choose to allocate its energy budget, B_i, equally to itself and each of its descendants such that it will naïvely sample and transmit the minimum of $\left(\frac{B_i}{|\mathfrak{C}_i| \cdot E_i^{\mathrm{S}}}, \frac{B_{p_i}}{|\mathfrak{C}_{p_i}| \cdot E_{p_i}^{\mathrm{S}}}\right)$ times regardless of whether the samples will eventually be forwarded towards the base station. $|\mathfrak{C}_i|$ and $|\mathfrak{C}_{p_i}|$ are the numbers of descendants of node i and node i's parent, p_i, respectively, and B_{p_i} is the energy budget of node p_i. As outlined in section 4.3, E_i^{S} and $E_{p_i}^{\mathrm{S}}$ are the energy required by node i and p_i correspondingly in order to sense a sample.

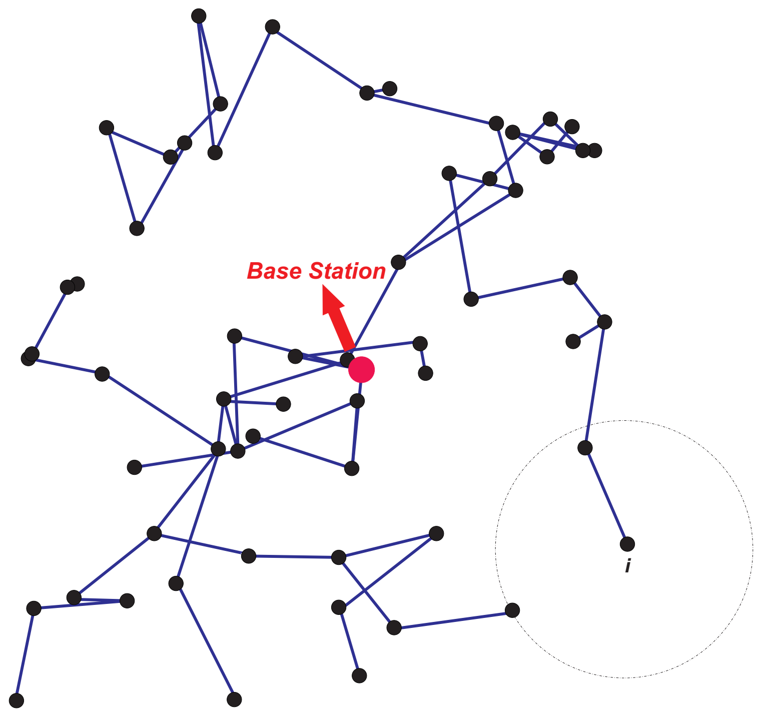
4.5.4 Results

We present the results of the simulation process described above in figures 4.20, 4.21, and 4.22. The error bars shown represent the standard error in the mean, and we remark that in the case of the fixed routing algorithm, the error bars are often smaller than the plotted symbol size.

Considering first figure 4.20. We observe that the algorithm with flexible routing delivers close to twice the quantity of information to the base station compared to the algorithm with an arbitrary fixed routing. This is as expected, since in communication networks that exhibit loops, there are typically many alternative routes available for routing data, and the algorithm with flexible routing is able to exploit these alternatives to deliver the maximum possible information to the base station. Note that the quantity of information delivered does not increase monotonically, but decreases in a number of cases. This effect is due to the way in which we have scaled the maximum energy budget of the nodes as



(a)



(b)

FIGURE 4.19: (a) A randomly created and connected WVSN (of 60 nodes) whose underlying communication network exhibits loops. (b) The resulting tree-structured network formed when each node makes an arbitrary choice of the route that its data will take toward the base station. The dotted circle in each graph represents the wireless range of node i. Neighbourhood nodes within this transmission range can hear the transmitted messages of the node. In both these networks, all nodes are set with the same transmission range.

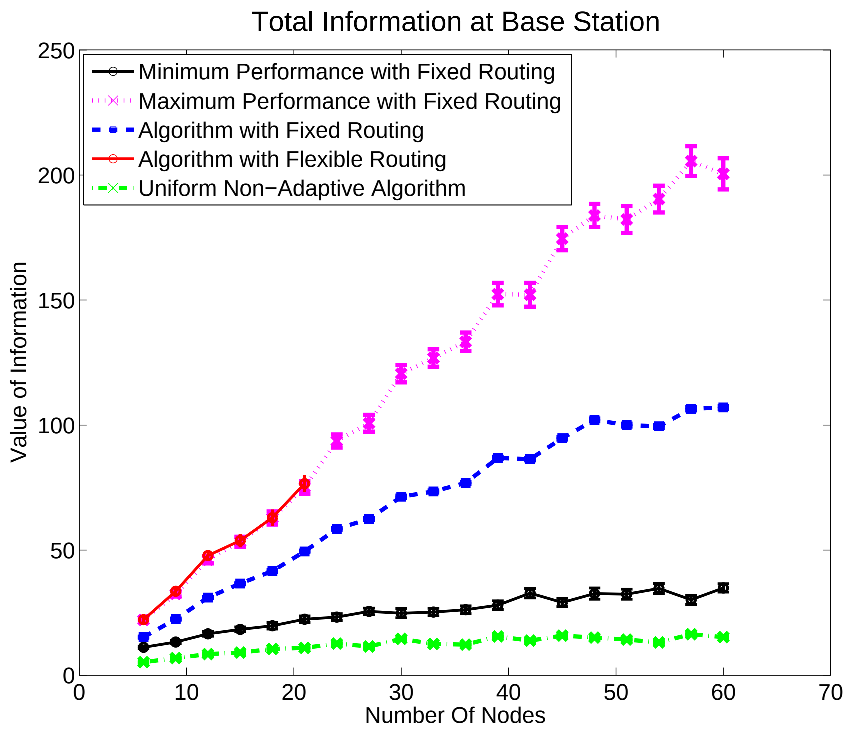


FIGURE 4.20: Simulation results showing the performance of the algorithms with flexible, fixed (with maximum and minimum performance), and uniform non-adaptive routing against total information collected at the base station.

the network increases in size. This scaling fails to fully account for the necessary increase in sample forwarding and, thus, the network becomes increasingly energy constrained as the network grows in size. The uniform non-adaptive algorithm, however, performs poorly compared to each of the two algorithms as it has no intelligence of adapting the actions of each individual nodes within the network.

In the same figure, we also show the mean maximum and minimum performance (that is, for each loopy network, we record the performance of the best and worst tree, and then average over each loopy network that we test) of the algorithm with fixed routing. We remark that by making an appropriate choice of parent, we can derive performance close to that of the algorithm with flexible routing without incurring any additional computation or communication cost as will be explained shortly.

However, the increased information delivered by the algorithm with flexible routing comes at a considerable communication and computational cost. Specifically, figures 4.21 and 4.22 show the total size of coordination messages exchanged by the nodes and the average computation time of each node. Note that these figures are presented on a logarithmic scale for a clearer visualization. In particular, figure 4.21 shows that typically only a few tens of kilobytes of coordination message packets are required by the algorithm with fixed routing, while the algorithm with flexible routing exhibits approximately two orders of magnitude more; with a few megabytes of coordination message packets being

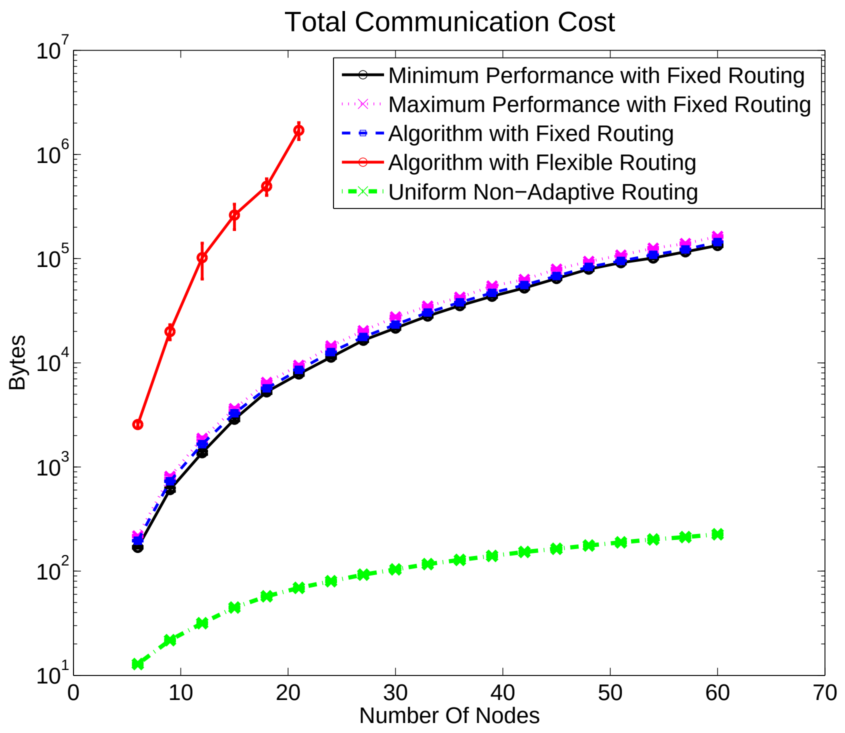


FIGURE 4.21: Simulation results showing the performance of the algorithms with flexible, fixed (with maximum and minimum performance), and uniform non-adaptive routing against total communication cost for coordination.

exchanged. The increase is due to the way in which the flexible routing algorithm keeps track all possible partitions of forwarding that grows exponentially with the number of nodes and potential routes.

Likewise, figure 4.22 shows that the average computation time of a node required by the algorithm with fixed routing is typically less than 1 millisecond, while that of the algorithm with flexible routing approaches 100 milliseconds (a two orders of magnitude increase)[9]. The increase in terms of computation time is due to the additional time which the flexible routing algorithm requires in order to enumerate each of the possible partitions of forwarding.

Speaking more generally, these results indicate that the algorithm with flexible routing is able to deliver significantly more information to the base station, but it incurs a considerable additional computation and communication cost in doing so. The choice of algorithm thus largely depends on the actual application domain. If the network is small, and the size of the actual data messages is large, then the algorithm with flexible routing is most appropriate. However, this algorithm scales poorly as the size or connectivity of the network increases (due to the exponential growth in the number of

[9]These measurements were performed on a 3GHz desktop PC. Typically, the nodes within a WVSN will use much lower powered processors and, thus, while we would expect the ratio between the algorithms to be the same, the overall computation time is likely to be longer.

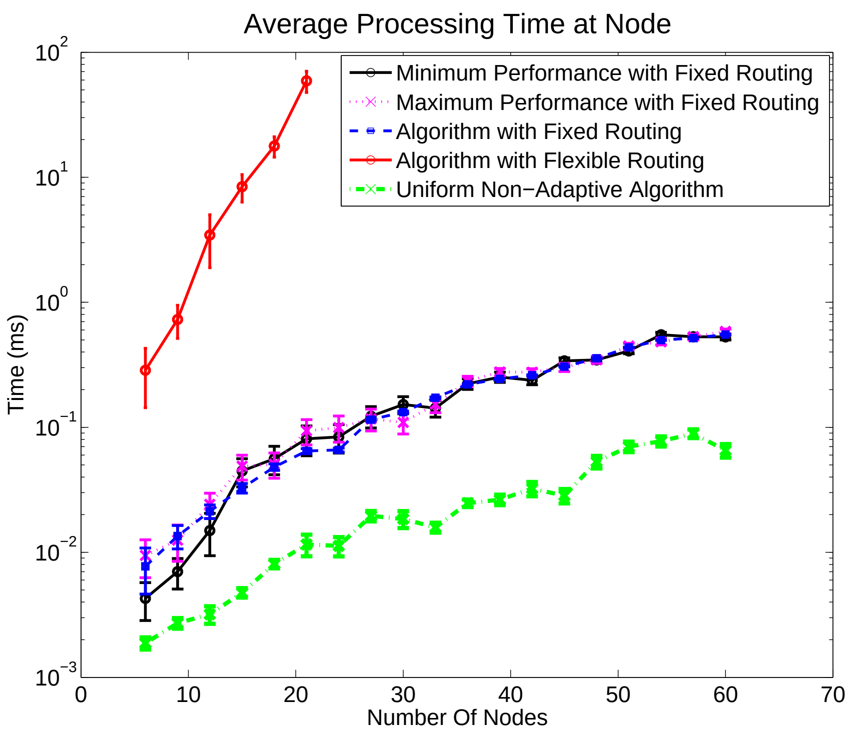


FIGURE 4.22: Simulation results showing the performance of the algorithms with flexible, fixed (with maximum and minimum performance), and uniform non-adaptive routing against average computation time at each node.

possible combinations of routing options that it must evaluate). In such cases, the size of the coordination messages may rapidly approach that of the actual data messages, and hence, coordination may not actually yield any energy saving. To address this, the algorithm with fixed routing may be run on the original loopy network by ignoring the fact that there exist alternative routing options, and having each node make an arbitrary choice of route. While the quantity of information delivered to the base station will be reduced (by up to 50% in our experiments), this solution will scale well and use minimal communication and computational resources.

4.6 Summary

In this chapter, we have considered the problem of inter-related adaptive sampling, transmitting, forwarding, and routing within a wireless visual sensor network in order to manage the limited energy resources of nodes in an effective and efficient way. We have developed two novel and optimal decentralised algorithms: one that assumes fixed routing and calculates the optimal sensing and forwarding actions that each node should perform (Contribution 4), and one that assumes flexible routing and makes optimal decisions regarding both the integration of actions that each node should perform, and also the route by which data should be forwarded to the base station (Contribution

5). In an empirical evaluation, we showed that the algorithm with flexible routing delivered approximately twice the quantity of information to the base station, but at a considerably higher communication and computational cost (Contribution 6). Thus, while the algorithm with flexible routing is suitable for networks with a small numbers of nodes, it scales poorly, and as the size of the network increases, the algorithm with fixed routing should be favoured.

Our ongoing work in this area includes relaxing the restriction that the nodes may only forward data to nodes that are closer to the base station (in terms of hop count) than themselves and, in particular, we would like to characterise the circumstances in which this may yield some benefit. More significantly, we would also like to develop a principled algorithm for making the choice of route when applying the algorithm with fixed routing to WVSN whose underlying communication network exhibits loops. That is, rather than having the nodes make an arbitrary choice of parent in order to convert the loopy network into a tree-structured one, we would like to provide some guiding algorithm such that the tree-structured network that is generated delivers the greatest quantity of information to the centre (without having to run the full algorithm with flexible routing). Our empirical evaluation indicates that there is potential in doing so as the performance of the algorithm with fixed routing is very close to that of the algorithm with flexible routing if the appropriate fixed route is selected (recall figure 4.20)[10]. However, how to select this particular tree-structure in an efficient decentralised manner is an open question.

[10]Note the algorithms are not necessarily identical in this case, since the algorithm with flexible routing allows individual nodes to forward data through multiple routes.

Chapter 5

Conclusions and Future Work

In this final chapter, we conclude the thesis by reviewing its contributions to the field of decentralised control in the domain of WSN, and outline the opportunities for future work. To this end, in section 5.1, we look back at the research challenges and requirements that have motivated this thesis, and then discuss our research contributions. Then, in section 5.2, we propose a number of directions on how our work can be extended.

5.1 Conclusions

WSNs are receiving significant multi-disciplinary research interest from institutions around the globe. The research opportunities in this area are diverse and plentiful. In this thesis, we have focused primarily on issues associated with energy management as the efficient management of the limited energy resources of such networks is central to their successful operation. Within this context, we concentrated in particular on the sampling, transmitting, forwarding, and routing strategies of decentralised control to provide simple local decision rules for each node in the network. Such local decision rules give each node the ability to make its own decisions regarding its own sampling rates adjustment, transmitting, forwarding, and next-hop routing decisions based only on its local information and those of its parents and children, within the context of the system-wide goal of maximising the information value collected.

To this end, in chapter 2, we looked at a number of WSN applications and presented various approaches to controlling such networks. However, currently, most of these approaches adopt a centralised control mechanism which has issues associated with scalability, robustness, and dynamism that often exist in such energy-constrained networks. Given this, decentralised methods are appealing and, therefore, we use an agent-based approach in which each sensor node is represented by an autonomous agent capable of making local autonomous decisions in order to achieve its individual objectives. The individual agents also coordinate their activities cooperatively in a distributed manner towards achieving system-wide goals.

In chapter 3, we first concentrated on adaptive sampling as a means of focusing a node's energy consumption on obtaining the most important data. We tackled the problem by developing a principled information metric based upon Fisher information and Gaussian process regression that allows the information content of a node's observations to be expressed. With this metric, we devised three novel decentralised information-based adaptive sampling algorithms which represent a trade-off in computational cost and optimality. Given a set of node readings, we showed how an optimal and a greedy adaptive sampling algorithms can be devised. These algorithms are, however, only tractable for very small problems and, thus, we develop a more practical, heuristic information-based adaptive sampling algorithm. This approach is a better choice for larger sampling problems (beating the benchmarks and obtaining performance close to the optimal one, but with much lower time complexity). The empirical results show that all the three algorithms are effective in balancing the trade-off associated with wanting to gain as much information as possible by sampling as often as possible, with the energy constraints available to accomplish these activities.

In chapter 4, we then moved on to consider the problem of inter-dependent adaptive sampling, transmitting, forwarding, and routing within a WVSN in order to manage the limited energy resources of nodes in an effective and efficient way. These actions are inter-related in this setting because each node's energy consumption must be optimally allocated between sampling and transmitting its own data, receiving and forwarding the data of other nodes, and routing any data. Specifically, we developed two novel optimal decentralised algorithms. The first assumes that the route by which data is forwarded to the base station is fixed and then calculates the optimal sensing and forwarding rates that each node should perform. The second deals with flexible routing and makes optimal decisions regarding both the integration of actions that each node should perform, and also the route by which this data should be forwarded to the base station. In an empirical evaluation, we showed that the algorithm with flexible routing delivered approximately twice the quantity of information to the base station. However, this gain comes at a considerable communication and computational cost. The algorithm with flexible routing is therefore suitable only for networks with a small numbers of nodes. As the size of the network increases, the algorithm with fixed routing should be favoured.

5.2 Future Work

There are a number of extensions that can be added to our work to make it more applicable to a wider range of scenarios in the domain of WSNs. Specifically these are as follows:

- **Model the Spatial Correlation of Node Samples:** As argued in section 2.3, the incorporation of spatial correlations between nodes into both the information metric and the algorithms would be beneficial. Incorporating this aspect will allow the nodes that are physically close together in a network to autonomously and more efficiently coordinate their limited resources by dividing the sampling tasks between themselves as they may be sensing similar readings and, thus, forwarding redundant data to the base station.

- **Further Exploration on Information Metrics:** The use of Shannon entropy and Kullback-Leibler divergence should then be investigated as alternative information metrics to express the information content of spatio-temporally correlated sensor samples.

- **Extended Mechanisms**. Relaxing the restriction that the nodes may only forward data to nodes that are closer to the base station (in terms of hop count) should be considered. More significantly, we believe it would be valuable to develop a principled decentralised algorithm for making the choice of route when applying the algorithm with fixed routing to networks whose underlying communication networks exhibit loops such that the generated tree-structured network delivers the greatest quantity of information to the base station while using minimum communication and computational resources (as discussed in section 4.5.4).

- **Simulator Improvements:** We think it would be advantageous to extend our current simulator so that a more comprehensive domain model of the wireless communication channels could be addressed in order to model the probability of dropped, undelivered, or corrupted messages more realistically (recall the discussion in section 4.5.1.1). This feature is important to appropriately test our mechanisms' scalability and robustness.

- **Non-Cooperative/Competitive Nodes:** We believe that in some settings it would be necessary to extend the algorithms to be incentive-compatible to include a payment scheme where rational selfish nodes are rewarded (e.g. for forwarding messages to the destination). This will eventually drive them towards achieving global system-wide goals in a non-cooperative setting (e.g. in a mobile ad hoc network where sensors are owned by different providers, each of which has different priorities, as discussed in section 2.5). To this end, the area of *distributed computational mechanism design* (Dash et al., 2003) seems to be a promising point of departure because it applies the tools of economic and game theory to design rules of interaction that yield some desired system-wide goals (Binmore, 1991; Dixit and Nalebuff, 1993; Song et al., 2008). In particular, the nodes are assumed to be economically rational entities that act in a *game-theoretic* way and select a best-response strategy to selfishly maximise their expected utility in accordance with other nodes and, thereby modelling the effect their actions will have on other node' actions. This detailed modelling further facilitates the design of a system in which certain system wide properties emerge, despite the selfish actions and goals of the constituent components' interaction.

- **Real Deployment:** For real deployment purposes, a key next step is to install our algorithms on the actual nodes. This is important in this context in order to see how well they scale and perform in practice in term of maximising the information value collected at the base station.

Appendix A

FLOODNET Log Data

Node Name	Mean Water Level	Variance Water Level	Battery Voltage	Date Stamp	Time Stamp
Floodnet1	55.31	1.21	11.72	05/10/14	00:04:27
Floodnet1	52.84	1.38	11.72	05/10/14	00:09:27
Floodnet1	52.19	0.34	11.72	05/10/14	00:14:27
Floodnet1	64.47	0.25	11.72	05/10/14	00:19:27
nodeA	181.56	1288.18	12.67	05/10/14	00:01:17
nodeA	142.50	668.75	12.67	05/10/14	00:06:17
nodeA	135.94	830.37	12.67	05/10/14	00:11:17
nodeA	135.62	1062.11	12.67	05/10/14	00:16:17
nodeG	587.19	1157.71	12.67	05/10/14	00:00:39
nodeG	588.44	700.68	12.74	05/10/14	00:05:39
nodeG	597.19	1145.21	12.74	05/10/14	00:10:39
nodeG	592.19	1179.59	12.74	05/10/14	00:15:39
nodeC	52.19	129.59	12.54	05/10/14	00:03:16
nodeC	49.69	109.28	12.54	05/10/14	00:08:16
nodeC	47.81	92.09	12.54	05/10/14	00:13:16
nodeC	51.88	108.98	12.54	05/10/14	00:18:16
nodeB	1088.44	456.93	11.78	05/10/14	00:03:00
nodeB	1096.25	579.69	11.78	05/10/14	00:08:00
nodeB	1095.31	574.90	11.78	05/10/14	00:13:00
nodeB	1096.56	610.06	11.71	05/10/14	00:18:00
nodeF	53.12	315.23	12.80	05/10/14	00:02:17
nodeF	60.62	830.86	12.80	05/10/14	00:07:17
nodeF	62.81	920.21	12.80	05/10/14	00:12:17
nodeF	63.12	765.23	12.80	05/10/14	00:17:17
Floodnet1	65.28	0.39	11.73	05/10/14	00:24:27
Floodnet1	67.03	1.84	11.72	05/10/14	00:29:27
Floodnet1	68.25	0.94	11.72	05/10/14	00:34:27
Floodnet1	68.91	0.15	11.72	05/10/14	00:39:27
nodeA	136.88	696.48	12.67	05/10/14	00:21:17
nodeA	143.44	1128.81	12.67	05/10/14	00:26:17
nodeA	138.12	1077.73	12.67	05/10/14	00:31:17
nodeA	144.38	1030.86	12.67	05/10/14	00:36:17
nodeC	53.12	83.98	12.54	05/10/14	00:23:16
nodeC	48.12	127.73	12.54	05/10/14	00:28:16
nodeC	52.50	156.25	12.54	05/10/14	00:33:16
nodeC	51.88	158.98	12.54	05/10/14	00:38:16
nodeB	1097.50	725.00	11.71	05/10/14	00:23:01
nodeB	1100.00	656.25	11.71	05/10/14	00:28:01

Appendix B

Acronyms

ADOPT Asynchronous Distributed Optimization

ADV Advertisement

APO Asynchronous Partial Overlay

ARA Adaptive Routing Algorithm

A/D Analog/Digital

BDDF Bayesian Decentralised Data Fusion

BIP Binary Integer Programming

BGP Border Gateway Protocol

ci Confidence Interval

CPU Central Processing Unit

DBA Distributed Breakout Algorithm

DCE Data Combining Entities

DCOP Distributed Constraint Optimization

DC-WSNS Decentralised Control of Wireless Sensor Network Simulator

DD Directed Diffusion

DF Directional Flooding

DisCOP Distributed Constraint Optimization

DPOP Distributed Pseudotree Optimization Procedure

DSR Dynamic Source Routing

EWSN Environmental Wireless Sensor Network

GBR Gradient Based Routing

GIS Geographical Information System

GLPK GNU Linear Programing Kit

GP Gaussian Process

GPRS General Packet Radio Service

GPS Global Positioning System

IDEALS Information Managed Energy Aware Algorithm for Sensor Networks with Rule Managed Reporting

IDR Information Directed Routing

IDSQ Information-Driven Sensor Querying

LAN Local Area Network

LEACH Low Energy Adaptive Clustering Hierarchy

MAS Multi-Agent System

MBC Market Based Control

METAR Meteorological Aviation Routine Report

MKP Multiple Knapsack Problem

PC Personal Computer

PCMCIA Personal Computer Memory Card International Association

PEGASIS Power Efficient Gathering in Sensor Information Systems

PIC Programmable Interface Controller

REQ Request

RISC Reduced Instruction Set Computer

RR Rumor Routing

SBB Synchronous Branch and Bound

SBC Single Board Computer

SIMD Single Instruction Multiple Data

SOR Self Organised Routing

SORA Self Organising Resource Allocation

SPIN	Sensor Protocols for Information via Negotiation
TCP/IP	Transmission Control Protocol/Internet Protocol
USAC	Utility Based Sensing and Communication Protocol
VGA	Video Graphic Array
WSN	Wireless Sensor Network
WVSN	Wireless Visual Sensor Network

Bibliography

K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–49, May 2005.

I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–14, August 2002.

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

C. Alippi and C. Galperti. An adaptive system for optimal solar energy harvesting in wireless sensor network nodes. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(6):1742–50, July 2008.

A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. W. Zhang, H. Cao, M. Sridharan, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, M. Nesterenko, R. Shah, S. Kulkami, M. Aramugam, L. M. Wang, M. Gouda, Y. R. Choi, D. Culler, P. Dutta, C. Sharp, G. Tolle, M. Grimmer, B. Ferriera, and K. Parker. ExScal: Elements of an extreme scale wireless sensor network. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 102–8, August 2005.

Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation.* ISBN: 047141655X. Wiley Interscience, June 2001.

K. Binmore. *Fun and Games: A Text on Game Theory.* ISBN: 0669246034. D.C. Heath, October 1991.

C. M. Bishop. *Pattern Recognition and Machine Learning.* ISBN: 0387310738. Springer, 2006.

A. Boukerche. *Algorithms and Protocols for Wireless Sensor Networks.* ISBN: 0471798134. WileyBlackwell, November 2008.

D. Braginsky and D. Estri. Rumor routing algorithm for sensor networks. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31, 2002.

R. J. Braithwaite and S. C. B. Raper. Glaciers and their contribution to sea level change. *Physics and Chemistry of the Earth*, 27(32-34):1445–1454, 2002.

M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *Journal of IEEE Computer*, 39(2):68–75, 2006.

R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer. A reactive soil moisture sensor network: Design and field evaluation. *International Journal of Distributed Sensor Networks*, 1(2):149–162, April-June 2005.

B. R. Chalamala. Portable electronics and the widening energy gap. *Proceedings of the IEEE*, 95(11):2106–7, November 2007.

P. Chatterjee and N. Das. A distributed algorithm for load-balanced routing in multihop wireless sensor networks. In *Proceedings of the 9th International Conference on Distributed Computing and Networking*, pages 332–8, 2008.

C. F. Chen and J. Ma. Mobile enabled large scale wireless sensor networks. In *Proceedings of the 8th International Conference on Advanced Communication Technology*, pages 333–8, February 2006.

C. S. Chen, M. S. Kang, J. C. Hwang, and C. W. Huang. Application of binary integer programming for load transfer of distribution systems. In *Proceedings of the International Conference on Power System Technology*, volume 1, pages 305–10, 2000.

L. X. Chen. Ns2 based performance measurement of mobile ad hoc networks routing protocols. *Journal of Computational Information Systems*, 3(1):109–115, February 2007.

Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, S. Phelps, J. A. Rodriguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

K. Chintalapudi, T. Fu, J. Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri. Monitoring civil structures with a wireless sensor network. *IEEE Internet Computing*, 10(2):26–34, March-April 2006.

C. Y. Chong and S. P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–56, August 2003.

M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3):293–313, 2002.

U. M. Colesanti, C. Crociani, and A. Vitaletti. On the accuracy of OMNeT++ in the wireless sensor networks domain: Simulation vs. testbed. In *Proceedings of the 4th ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, pages 25–31, 2007.

D. D. Corkill, D. Holzhauer, and W. Koziarz. Turn off your radios! environmental monitoring using power-constrained sensor agents. In *Proceedings of the 1st International Workshop on Agent Technology for Sensor Networks*, pages 31–38, May 2007.

T. M. Cover and J. A. Thomas. *Elements of Information Theory.* ISBN: 0471241954. John Wiley and Sons, March 2006.

N. A. C. Cressie. *Statistics for Spatial Data.* ISBN: 0471002550. Wiley-Interscience, 1993.

T. Dang, N. Bulusu, W. Feng, S. Frolov, and A. Baptista. Adaptive sampling in the COlumbia RIvEr observation network. In *Proceedings of the 5th ACM Conference on Embedded Networked Sensor Systems*, pages 429–30, November 2007.

R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational mechanism design: A call to arms. *IEEE Intelligent Systems*, 18(6):40–7, November/December 2003.

D. De Roure. FLOODNET: A new flood warning system. *Ingenia*, 23:49–51, 2005.

K. A. Delin, R. P. Harvey, N. A. Chabot, S. P. Jackson, M. Adams, D. W. Johnson, and J. T. Britton. Sensor web in Antarctica: Developing an intelligent, autonomous platform for locating biological flourishes in cryogenic environments. In *Proceedings of the 34th Annual Lunar and Planetary Science Conference*, March 2003.

K. A. Delin, S. P. Jackson, D. W. Johnson, S. C. Burleigh, R. R. Woodrow, M. McAuley, J. T. Britton, J. M. Dohm, T. P. A. Ferre, F. Ip, D. F. Rucker, and V. R. Baker. Sensor web for spatio-temporal monitoring of a hydrological environment. In *Proceedings of the 35th Annual Lunar and Planetary Science Conference*, March 2004.

A. Dixit and B. Nalebuff. *Thinking Strategically: Competitive Edge in Business, Politics and Everyday Life.* ISBN: 0393310353. W. W. Norton and Company, April 1993.

H. Dong, J. G. Lu, and Y. X. Sun. Adaptive distributed compression algorithm for wireless sensor networks. In *Proceedings of the 1st International Conference on Innovative Computing, Information and Control*, pages 283–286, 2006.

E. Egea-Lopez, F. Ponce-Marin, and J. Vales-Alonso. OBIWAN: Wireless sensor networks with OMNeT++. In *Proceedings of the 13th IEEE Mediterranean Electrotechnical Conference*, pages 777–80, 2006.

V. Ekanayake, C. Kelly-IV, and R. Manohar. An ultra low-power processor for sensor networks. *SIGPLAN Notices*, 39(11):27–36, November 2004.

A. Elsaify, P. Padhy, K. Martinez, and G. Zou. GWMAC - A TDMA based MAC protocol for a glacial sensor network. In *Proceedings of the 4th ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, October 2007.

E. Ertin. Gaussian process models for censored sensor readings. In *Proceedings of the 14th IEEE International Workshop on Statistical Signal Processing*, pages 665–9, August 2007.

A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 639–646, 2008.

J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, 18(1):61–72, July 2005.

B. R. Frieden. *Science from Fisher Information: A Unification.* ISBN: 0521009111. Cambridge University Press, June 2004.

J. Gao and L. Zhang. Load-balanced short-path routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(4):377–88, April 2006.

A. Girard. *Approximate Methods for Propagation of Uncertainty with Gaussian Process Models.* PhD thesis, University of Glasgow, Scotland, UK, October 2004.

C. Guestrin, A. Krause, and A. P. Singh. Near optimal sensor placements in Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 265–272, 2005.

T. Gulrez and M. Kavakli. Precision position tracking in virtual reality environments using sensor networks. In *Proceedings of the IEEE International Symposium on Industrial Electronics*, pages 1997–2003, 2007.

S. Habib, M. Safar, and M. Taha. Using an evolutionary search approach to determine the sensors' coverage within wireless sensor networks. In *Proceedings of the 5th International Conference on Advances in Mobile Computing and Multimedia*, pages 165–74, 2007.

S. Haddrill. Summer floods 2007: Learning the lessons. Technical report, Association of British Insurers, UK, `http://www.abi.org.uk/BookShop/ResearchReports/Flooding%20in%20the%20UK%20Full.pdf`, November 2007.

J. Harman. Review of 2007 summer floods. Technical report, Environment Agency, UK, `http://publications.environment-agency.gov.uk/pdf/GEHO1107BNMI-e-e.pdf?lang=_e`, December 2007.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statictical Learning: Data Mining, Inference, and Prediction.* ISBN: 0387848576. Springer, March 2009.

T. He, S. Krishnamurthy, L. Q. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. VigilNet: An integrated sensor network system for energy-efficient surveillance. *ACM Transactions on Sensor Networks*, 2(1):1–38, February 2006.

M. Hempstead. An ultra low power system architecture for sensor network applications. In *Proceedings of the 32nd International Symposium on Computer Architecture*, pages 208–19, 2005.

K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming*, pages 222–36, 1997.

B. Horling, R. Vincent, R. Mailler, J. Y. Shen, R. Becker, K. Rawlins, and V. Lesser. Distributed sensor network for real time tracking. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 417–24, 2001.

J. Huang, R. Kumar, A. E. Kamal, and R. Weber. Development of a wireless soil sensor network. *American Society of Agriculture and Biosystems Engineering*, 2008.

I. S. Hwang, K. Roy, H. Balakrishnan, and C. Tomlin. A distributed multiple-target identity management algorithm in sensor networks. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 728–34, 2004.

C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16, February 2003.

N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, April 2001.

X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pages 463–8, 2005.

J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "Smart Dust". In *Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 271–8, 1999.

A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems*, 6 (4), 2007.

A. Kansal, D. Potter, and M. B. Srivastava. Performance aware tasking for environmentally powered sensor networks. 31(1):223–34, June 2004.

H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. ISBN: 0470519231. WileyBlackwell, August 2007.

H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. ISBN: 3540402861. Springer, 2004.

D. A. Khan and X. H. Peng. Investigation of energy efficiency for singlehop and multihop routing schemes in a wireless sensor network. In *Proceedings of the 6th IEEE International Conference on 3G and Beyond*, pages 31–34, 2005.

M. M. H. Khan, T. Abdelzaher, and K. K. Gupta. Towards diagnostic simulation in sensor networks. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems*, pages 252–265, 2008.

N. Kimura and S. Latifi. A survey on data compression in wireless sensor networks. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, volume 2, pages 8–13, 2005.

R. Kleihorst, B. Schueler, A. Danilin, and M. Heijligers. Smart camera mote with high performance vision system. In *Proceedings of the ACM SenSys Workshop on Distributed Smart Cameras*, October 2006.

Y. B. Ko, J. M. Choi, and J. H. Kim. A new directional flooding protocol for wireless sensor networks. In *Proceedings of the International Conference on Information Networking Technologies for Broadband and Mobile Networks*, pages 93–102, 2004.

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 2–10, 2006.

A. Krause, A. P. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–84, February 2008.

S. Kroc and V. Delic. Personal wireless sensor network for mobile health care monitoring. In *Proceedings of the 6th International Conference on Telecommunications in Modern Satellite, Cable, and Broadcasting Services*, volume 2, pages 471–4, 2003.

J. Kulik, W. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, 8(2-3):169–185, March/May 2002.

A. Ledeczi, A. Nadas, P. Volgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dora, K. Molnar, M. Maroti, and G. Simon. Countersniper system for urban warfare. *ACM Transactions on Sensor Networks*, 1(2):153–77, 2005.

V. Lesser. *Distributed Sensor Networks: A Multiagent Perspective.* ISBN: 1402074999. Springer, July 2003.

Y. Q. Li, H. Y. Yu, B. Su, and Y. H. Shang. Hybrid micropower source for wireless sensor network. *IEEE Sensors Journal*, 8(9):678–681, June 2008.

S. Lindsey and C. S. Raghavendra. PEGASIS: Power efficient gathering in sensor information systems. In *Proceedings of the IEEE Aerospace Conference*, volume 3, pages 3–1125–3–1130, 2002.

J. Liu, F. Zhao, and D. Petrovic. Information-directed routing in adhoc sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):851–61, April 2005a.

L. F. Liu, S. H. Zou, L. Zhang, and S. D. Cheng. An energy-efficient routing and self-organization algorithm in wireless sensor networks. *Journal of China Universities of Posts and Telecommunications*, 12(2):87–93, June 2005b.

B. P. L. Lo and G. Z. Yang. Key technical challenges and current implementations of body sensor networks. In *Proceedings of the 2nd International Workshop on Wearable and Implantable Body Sensor Networks*, pages 1–5, 2005.

X. M. Lu, M. Spear, K. Levitt, and S. F. Wu. iBubble: Multi-keyword routing protocol for heterogeneous wireless sensor networks. In *Proceedings of the 27th IEEE Communications Society Conference on Computer Communications*, pages 1642–50, 2008.

D. J. C. Mackay. Introduction to Gaussian process. In *Proceedings of Neural Networks and Machine Learning*, pages 133–65, 1998.

R. Mailler and V. Lesser. Asynchronous partial overlay: A new algorithm for solving distributed constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 25:529–576, January/April 2006.

G. Mainland, D. C. Parkes, and M. Welsh. Decentralised, adaptive resource allocation for sensor networks. In *Proceedings of the 2nd USENIX/ACM Symposium on Networked Systems Design and Implementation*, pages 315–28, 2005.

A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor network for habitat monitoring. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97, 2002.

A. Makarenko and H. Durrant-Whyte. Decentralized data fusion and controls in active sensor network. In *Proceedings of the 7th International Conference on Information Fusion*, pages 479–86, 2004.

M. Marin-Perianu, N. Meratnia, P. Havinga, L. M. S. de Souza, J. Muller, P. Spiess, S. Haller, T. Riedel, and C. Decker. Decentralized enterprise systems: A multiplatform wireless sensor network approach. *IEEE Wireless Communications*, 14(6):57–66, December 2007.

K. Martinez, P. Padhy, A. Elsaify, G. Zou, A. Riddoch, J. K. Hart, and H. L. R. Ong. Deploying a sensor network in an extreme environment. In *Proceedings of* *the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pages 186–93, 2006.

G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 374–381, 2006.

M. Medidi, J. Ding, and S. Medidi. Data dissemination using gossiping in wireless sensor networks. In *Proceedings of SPIE - the International Society for Optical Engineering*, pages 316–327, 2005.

S. Meninger, J. O. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang. Vibration-to-electric energy conversion. In *Proceedings of the International Symposium on Low Power Electronics and Design, Digest of Technical Papers*, pages 48–53, 1999.

G. Merrett, B. M. Al-Hashimi, N. White, and N. Harris. Resource aware sensor nodes in wireless sensor networks. *Journal of Physics: Conference Series*, 15(1):137–42, 2005.

V. Mhatre and C. Rosenberg. Design guidelines for wireless sensor networks: Communication, clustering and aggregation. *Ad Hoc Networks*, 2(1):45–63, January 2004.

P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–80, January 2005.

J. C. Neal, P. M. Atkinson, and C. W. Hutton. Flood inundation model updating using an ensemble Kalman filter and spatially distributed measurements. *Journal of Hydrology*, 336(3-4):401–415, April 2007.

S. Oh, P. Chen, M. Manzo, and S. Sastry. Instrumenting wireless sensor networks for real-time surveillance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3128–33, May 2006.

M. A. Osborne, A. Rogers, S. Ramchurn, S. J. Roberts, and N. R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 109–120, 2008.

P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings. A utility-based sensing and communication model for a glacial sensor network. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, pages 1353–1360, 2006.

P. Padhy, K. Martinez, A. Riddoch, H. L. R. Ong, and J. K. Hart. Glacial environment monitoring using sensor networks. In *Proceedings of the Workshop on Real-World Wireless Sensor Networks*, June 2005.

C. S. Park and P. H. Chou. AmbiMax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes. In *Proceedings of the 3rd IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 168–77, September 2006.

S. Park, A. Savvides, and M. B. Srivastava. SensorSim: A simulation framework for sensor networks. In *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 104–111, 2000.

S. Park, A. Savvides, and M. B. Srivastava. Battery capacity measurement and analysis using lithium coin cell battery. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 382–387, 2001.

A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 266–271, August 2005.

M. Pitt. Learning lessons from the 2007 floods. Technical report, The Pitt Review, UK, `http://archive.cabinetoffice.gov.uk/pittreview/thepittreview.html`, June 2008.

D. Pizzocaro, M. P. Johnson, H. Rowaihy, S. Chalmers, A. Preece, A. Bar-Noy, and T. La Porta. A knapsack approach to sensor-mission assignment with uncertain demands. In *Proceedings of the International Conference on Unmanned/Unattended Sensors and Sensor Networks V*, volume 7112, October 2008.

V. Pop, H. J. Bergveld, P. H. L. Notten, and P. P. L. Regtien. State-of-the-art of battery state-of-charge determination. *Measurement Science and Technology*, 16(12): R93–R110, December 2005.

Y. Qin and C. Y. Lee. An improved dynamic source algorithm for MANETs. In *Proceedings of the SPIE - the International Society for Optical Engineering*, volume 5284, pages 381–7, 2004.

J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy. PicoRadio supports ad hoc ultra-low power wireless networking. *Computer*, 33(7):42–8, July 2000.

P. Rabinowitz and P. J. Davis. *Methods of Numerical Integration.* ISBN: 0486453391. Dover Publications, 2nd edition, February 2006.

V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pages 457–62, 2005.

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

F. Rahman and N. Shabana. Wireless sensor network based personal health monitoring system. *WSEAS Transactions on Systems*, 5(5):966–72, May 2006.

C. E. Rasmussen. Gaussian processes in machine learning. *Artificial Intelligence*, 3176: 63–71, 2004.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* ISBN: 026218253X. The MIT Press, January 2006.

C. Reinisch, W. Kastner, and G. Neugschwandtner. Multicast communication in wireless home and building automation: ZigBee and DCMP. In *Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 1380–3, 2007.

B. Rinner, T. Winkler, W. Schriebl, M. Quaritscha, and W. Wolf. The evolution from single to pervasive smart cameras. In *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–10, September 2008.

A. Rogers, D. D. Corkill, and N. R. Jennings. Agent technologies for sensor networks. *IEEE Intelligent Systems*, 24(2):13–17, 2009.

A. Rogers, E. David, and N. R. Jennings. Self-organised routing for wireless microsensor networks. *IEEE Transactions on Systems, Man, and Cybernetics (Part A)*, 35 (3):349–359, 2005.

C. Schurgers and M. B. Srivastava. Energy efficient routing in wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications for Network-Centric Operations: Creating the Information Force*, volume 1, pages 357–61, 2001.

M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2):69–106, April 2004.

Z. Shelby, C. Pomalaza-Raez, H. Karvonen, and J. Haapola. Energy optimization in multihop wireless embedded and sensor networks. *International Journal of Wireless Information Networks*, 12(1):11–21, January 2005.

M. Shinozuka, M. Q. Feng, P. Chou, Y. Chen, and C. Park. MEMS-based wireless real-time health monitoring for bridges. In *Proceedings of the 3rd International Conference on Eathquake Engineering*, October 2004.

G. Simon, G. Balogh, G. Pap, M. Maroti, B. Kusy, J. Sallai, A. Ledeczi, A. Nadas, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 1–12, 2004.

P. Sinha and A. A. Zoltners. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–15, May-June 1979.

A. Sobeih, W. P. Chen, J. C. Hou, L. C. Kung, N. Li, H. Lim, H. Y. Tyan, and H. Zhang. J-Sim: A simulation environment for wireless sensor networks. In *Proceedings of the 38th Annual Simulation Symposium*, pages 175–87, 2005.

B. Song, C. Soto, A. K. Roy-Chowdhury, and J. A. Farrell. Decentralized camera network control using game theory. In *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–8, 2008.

J. A. Stankovic. Research challenges for wireless sensor networks. *ACM SIGBED Review*, 1(2):9–12, July 2004.

B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: A survey. *Ad Hoc Networks*, 3(3):281–323, May 2005.

J. Tang, B. Hao, and A. Sen. Relay node placement in large scale wireless sensor networks. *Computer Communications*, 29(4):490–501, February 2006.

R. Torah, P. Glynne-Jones, J. Tudor, T. O'Donnell, S. Roy, and S. Beeby. Self-powered autonomous wireless sensor node using vibration energy harvesting. *Measurement Science And Technology*, 19(12), December 2008.

M. Ulema, J. M. Nogueira, and B. Kozbe. Management of wireless ad hoc networks and wireless sensor networks. *Journal of Network and Systems Management*, 14(3): 327–333, September 2006.

M. Veyseh, B. Wei, and N. F. Mir. An information management protocol to control routing and clustering in sensor networks. *Journal of Computing and Information Technology*, 13(1):53–68, March 2005.

M. C. Vuran, O. B. Akan, and I. F. Akyildiz. Spatio-temporal correlation: Theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–59, June 2004.

D. Wagner and R. Wattenhofer. *Algorithms for Sensor and Ad Hoc Networks: Advanced Lectures*. ISBN: 354074990X. Springer, September 2007.

H. B. Wang, G. Pottie, K. Yao, and D. Estrin. Entropy-based sensor selection heuristic for target localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 36–45, 2004.

B. A. Warneke and K. S. J. Pister. An ultra-low energy microcontroller for Smart Dust wireless sensor networks. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 316–17, February 2004.

M. A. Weimer, T. S. Paing, and R. A. Zane. Remote area wind energy harvesting for low-power autonomous sensors. In *Proceedings of the 37th IEEE Power Electronics Specialists Conference*, pages 1–5, June 2006.

G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* ISBN: 0262731312. MIT Press, September 2000.

G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, March-April 2006.

R. Willett, A. Martin, and R. Nowak. Backcasting: Adaptive sampling for sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 124–33, 2004.

M. Wooldridge. *An Introduction to Multi-agent Systems.* ISBN: 047149691X. John Wiley and Sons, March 2002.

L. Xia, X. Chen, and X. H. Guan. A new gradient-based routing protocol in wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Software and Systems*, pages 318–25, 2005.

M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the 2nd International Conference on Multi-Agent Systems*, pages 401–8, 1996.

B. C. Yong, T. Kurokawa, Y. Takefuji, and S. K. Hwa. An O(1) approximate parallel algorithm for the n-task-n-person assignment problem. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1503–6, 1993.

H. W. Yoon, B. H. Lee, T. J. Lee, and M. Y. Chung. Energy efficient with power management routing to increase network lifetime in sensor networks. In *Proceedings of the International Conference on Computational Science and It's Applications*, volume 4, pages 46–55, 2004.

M. A. Youssef, M. F. Younis, and K. A. Arisha. A constrained shortest-path energy-aware routing algorithm for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference Record*, pages 794–9, 2002.

J. Zachary. A decentralized approach to secure management of nodes in distributed sensor networks. In *Proceedings of the IEEE Military Communications Conference*, pages 579–84, 2003.

T. Zahariadis and S. Voliotis. Open issues in wireless visual sensor networking. In *Proceedings of the 14th International Workshop on Systems, Signals and Image Processing and the 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services*, pages 335–338, June 2007.

Y. Zhang and W. He. Multi-mode piezoelectric energy harvesters for wireless sensor network based structural health monitoring. In *Proceedings of the International Society for Optical Engineering*, March 2008.

Y. C. Zhang and L. Cheng. Flossiping: A new routing protocol for wireless sensor networks. In *Proceedings of the IEEE International Conference on Networking, Sensing, and Control*, volume 2, pages 1218–23, 2004.

F. Zhao and L. J. Guibas. *Wireless Sensor Networks: An Information Processing Approach.* ISBN: 1558609148. Morgan Kaufmann, July 2004.

J. Zhou and D. De Roure. FloodNet: Coupling adaptive sampling with energy aware routing in a flood warning system. *Journal of Computer Science and Technology*, 22 (1):121–30, 2007.