Using Local and Global Knowledge in Wireless Sensor Networks

A thesis submitted in partial fulfilment

of the requirement for the degree of Doctor of Philosophy

Christopher Gwilliams

31st March 2015

Cardiff University
School of Computer Science & Informatics

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD.

Signed ..... (candidate)

Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate)

Date .....

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

A copy of this document in various transparent and opaque machine-readable formats and related software is available at `http://github.com/encima`.

Abstract

Wireless sensor networks (WSNs) have advanced rapidly in recent years and the volume of raw data received at an endpoint can be huge. We believe that the use of local knowledge, acquired from sources such as the surrounding environment, users and previously sensed data, can improve the efficiency of a WSN and automate the classification of sensed data. We define local knowledge as knowledge about an area that has been gained through experience or experimentation. With this in mind, we have developed a three-tiered architecture for WSNs that uses differing knowledge-processing capabilities at each tier, called the Knowledge-based Hierarchical Architecture for Sensing (K-HAS). A novel aligning ontology has been created to support K-HAS, joining widely used, domain-specific ontologies from the sensing and observation domains. We have shown that, as knowledge-processing capabilities are pushed further out into the network, the profit - defined as the value of sensed data - is increased; where the profit is defined as the value of the sensed data received by the end user.

Collaborating with Cardiff University School of Biosciences, we have deployed a variation of K-HAS in the Malaysian rainforest to capture images of endangered wildlife, as well as to automate the collection and classification of these images. Technological limitations prevented a complete implementation of K-HAS and an amalgamation of tiers was made to create the Local knowledge Ontology-based Remote-sensing Informatics System (LORIS). A two week deployment in Malaysia suggested that the architecture was viable and that, even using local knowledge at the endpoint of a WSN, improved the efficiency of the network. A simulation was implemented to model K-

HAS and this indicated that the network became more efficient as knowledge was pushed further out towards the edge, by allowing nodes to prioritise sensed data based on inferences about its content.

Acknowledgements

My eternal gratitude goes to Alun Preece and Alex Hardisty for their guidance, assistance, support, enthusiasm and faith in me throughout the research and writing of this thesis. Their encouragement of testing new ideas, questioning everything and ensuring I step out of the box that it seems all too easy to stay inside. I am also grateful for the skills they have pushed me to gain, from teaching to presenting to professional writing. None of this work would have been possible without you both.

Benoit Goosens, Mike Bruford, Danica Stark and all of those involved with Danau Girang Field Centre have been fountains of knowledge, support and inspiration. At the very least, I thank you for your hospitality each year and to the staff members that put up with a computer scientist that is clearly not well adapted to life in the rainforest with limited connectivity to the outside world.

I am grateful to the School of Computer Science & Informatics, staff and students have provided methods of growth, new skills and support throughout my time at Cardiff University. My thanks go to Matt Williams, for challenging my work from the early stages right through to completion and providing relief through films, and Will Webberley, a friend gained through the PhD that provided support, both professionally and personally, and I do not believe I could have completed this work without him. Special mentions must go to Martin Chorley, Jonathan Quinn, Gualtiero Colombo, Konrad Borowiecki, Diego Pizzocaro, Przemyslaw Woznowski, Rich Coombs, Liam Turner, Mark Greenwood and Matthew John for being there in the pub, as well as in academia.

My deepest thanks also go to family and close friends, with special mentions to Natasha Kinsley-York for supporting me throughout the start of the PhD and helping me deal with the stress incurred and Chris Fellows, for his assistance and also being a sounding board for my many rants, ideas and everything in between. Finally, to Katie Lewis, for her ideas, support, patience and all-round awesomeness that has not only helped my work, but ensured I enjoy every second of my down time.

I would like to dedicate this work to my father, who passed away in 2009, there is no way I would be anywhere close to this point without his input, teaching and encouragement pushing me to go further than I ever imagined I could go. Words cannot express the love and gratitude I have for Robert and Gina Gwilliams, two role models that truly set the bar high.

Contents

List of Publications

The work introduced in this thesis is based on the following publications.

- [**?**] - Gwilliams, C., Preece, A., Hardisty, A., Goossens, B., & Ambu, L. N. (2012). Local and Global Knowledge to Improve the Quality of Sensed Data. International Journal of Digital Information and Wireless Communications (IJDIWC), 2(2), 164–180.

- [**?**] - Gwilliams, C., Preece, A. D., & Hardisty, A. R. (2012). Poster: using local and global knowledge in wireless sensor networks. In Proceedings of the 10th international conference on Mobile systems, applications, and services (pp. 515–516).

- [**?**] - Gwilliams, C., Preece, A., Hardisty, A., Goossens, B., & Ambu, L. N. (2012). K-HAS: An Architecture for Using Local and Global Knowledge in Wireless Sensor Networks. In The International Conference on Informatics and Applications (ICIA2012) (pp. 331–341).

List of Figures

List of Tables

Listings

List of Acronyms

API Application Programming Interface

BLE Bluetooth Low Energy

DA Data Aggregation

Darwin-SW Darwin Core Semantic Web

DC Data Collection

DGFC Danau Girang Field Centre

DP Data Processing

DwC Darwin Core

DwC-A Darwin Core Archive

EML Ecological Metadata Language

EPO Extension Plug-in Ontologies

EXIF EXchangeable Image Format

FN False Negative

FP False Positive

GBIF Global Biodiversity Information Facility

GEAR Geographical Energy Aware Routing

GK Global Knowledge

GSN Global Sensor Networks

HK High Knowledge

IFTTT If This Then That

INSIGHT INternet-Sensor InteGration for HabitaT monitoring

IoT Internet of Things

IR Infrared

JVM Java Virtual Machine

K-HAS Knowledge-based Hierarchical Architecture for Sensing

LK Local Knowledge

LKWS Lower Kinabatangan Wildlife Sanctuary

LOS Line of Sight

LORIS Local knowledge Ontology-based Remote-sensing Informatics System

MCFA Minimum Cost Forwarding Algorithm

MK Minimal Knowledge

NK No Knowledge

O&M Observations & Measurements

OBOE Extensible Observation Ontology

OGC Open Geospatial Consortium

OpenCV Open Computer Vision

OWL Web Ontology Language

QoS Quality of Service

ROI Region of Interest

RDL Ruleset Definition Language

SBC Single Board Compuetr

SDO Sensor Data Ontology

SensorML Sensor Model Language

SHO Sensor Hierarchy Ontology

SOS Sensor Observation Service

SPIN Sensor Protocol for Information via Negotiation

SPS Sensor Planning Ontology

SSN Semantic Sensor Network

SUMO Suggested Upper Merged Ontology

SWE Sensor Web Enablement

TEEN Threshold sensitive Energy Efficient sensor Network

TN True Negative

TOVE Toronto Virtual Enterprise

TP True Positive

Wi-Fi Wireless Fidelity

WSN Wireless Sensor Network

Chapter 1

Introduction

A wireless sensor network (WSN) consists of a collection of nodes with sensing and, typically, wireless communication capabilities. These sensing nodes can be complex and powerful devices with the ability to sense multiple phenomena simultaneously [**?** **?** **?**], or they can be simple motes with limited processing power [**?** **?** **?**]. A node has one or more sensors attached to it, with a wireless radio that is used to transmit the sensed data to an endpoint, where an endpoint is the base station of the network. Most WSNs use a single base station, providing a single endpoint, but there are networks with multiple endpoints; either used for different types of sensed data or to ensure that the load of the network is spread out.

Upon deployment, these nodes use their wireless capabilities to form communication links with their neighbours, where a neighbour is any node that is within transmission range. The way that nodes discover, and communicate with their neighbours is defined by their routing protocol. Routing protocols vary based on the purpose of the WSN, the requirements of data transmission as well as the characteristics of the nodes. Communication between nodes is expensive and drains the available power faster than any other action that a node performs [**?**]. For example, if a WSN is deployed in a building with constant power availability, then the routing protocol would not need to be modified to ensure nodes sleep to conserve battery or disable their radios for a period of time. However, not every WSN has unlimited resources at their disposal and these protocols, as well as the underlying structure of the network, are used to ensure the network is able to perform well for as long as possible.

Each WSN is different and each will have different constraints, a WSN that monitors traffic along a busy road may experience memory limitations, whereas a WSN that is deployed in the middle of a desert may experience power issues. Typically, however, all WSNs do the same thing: sense one or more characteristics of their environment and forward that data on to a specified endpoint; sometimes called the base station.

1.1 ~~The Local Knowledge Problem~~

~~While large in size, it is typical that a WSN would use low cost nodes with limited power, memory and computational capabilities; causing them to lack the ability to be aware of their surroundings or the data that they are sensing [15]. This means that, unless fixed by a routing protocol or a technician, data is delivered on a chronological basis and is then filtered at the base station, usually manually. Some WSNs store all of the data on the node and users of the network use a 'pull' model to query for data from nodes [**?**], but this requires some technical knowledge and, although it does increase the battery life of the nodes, it is a manual process again.~

~~The environment that a WSN is deployed in is often a rich source of data to be sensed (such as inside a bird's nest to sense temperature, humidity and movement), which often contains patterns that can be used to improve the performance of the network. For example, if a node knows that it is has only been triggering between the hours of 6pm and 5am for the past few weeks, it can then learn to enter a deep sleep outside of those hours or use that time to transmit data it has been storing while it assumes it will be inactive, based on previous days. Alternatively, this knowledge can be used to prioritise data throughout the network so that the most important data is received first, instead of the most recent. An example of this could be two camera nodes deployed facing the entry and exit of a building, tasked with looking for intruders between 5pm and 8am. If the camera facing the exit is triggered at 5:01pm and the camera on the entrance is triggered at 5:05pm, then the knowledge that the security guard leaves through the exit~

between 5:00pm and 5:08 pm will allow the entrance camera to prioritise its capture as more important, as it is an irregular occurrence.

This knowledge can be categorised as either *local* or *global*. Local knowledge (LK) is the knowledge of an area that has been gained through experience or experimentation [**?**] and global knowledge (GK) is knowledge that is generally available to all persons. For example, a researcher who has been tasked with deploying a WSN in the Amazon rainforest would use readily available sources, such as the Internet or prior research, to determine the humidity and weather patterns in order to use a node that could withstand such conditions. This would be classed as GK. However, a native to the Amazon may know that three of the locations in which the nodes are to be deployed are flooded for two weeks of the year, rendering their readings useless for that time period and increasing their risk of failure. This is LK, as it cannot be gained without experiencing the flooding in that area, or measuring local water levels.

We believe that the use of this knowledge can increase the effectiveness of the network, as well as prioritise data based on its value as opposed to when it was recorded. To show this, we have developed a network architecture for WSNs that utilises knowledge from the data it senses, as well as its deployed environment. It is called the Knowledge-based Hierarchical Architecture for Sensing (K-HAS) and this thesis will show how K-HAS addresses the problem of delivering the most important data first and improving the overall efficiency of the network.

1.1 Motivation

Throughout this thesis, we focus on a scenario motivated by our collaboration with Cardiff University School of Biosciences, who run a research centre in the Malaysian rainforest, in Sabah, known as Danau Girang Field Centre (DGFC) [2]. Located on the banks of the Kinabatangan river, DGFC has been running for more than six years and holds Masters, PhD and Undergraduate students from around the world, studying the

ecology and biodiversity of the unique region.

The rainforest surrounding the Kinabatangan river is unique because the area was heavily logged until the late 1970s and the river now serves as a corridor, between large palm oil plantations, connecting two separate rainforest lots. The area is now secondary rainforest (rainforest that has grown since being destroyed) and is experiencing a large variety of wildlife using the area as a habitat, or as a path. Some of this wildlife is unique to this area of the world and DGFC has had sightings of animals that have not been seen in many years [17].

There is a variety of research projects currently underway in the field centre, looking into fish population, crocodile attacks, hornbill habitats or the movement patterns of small mammals. One project that has been running almost since DGFC opened, is the *corridor monitoring programme*, a programme that consists of dozens of wildlife cameras deployed in various areas around DGFC and a set of images are taken whenever an animal triggers a break in their infrared (IR) sensor.

The Kinabatangan is a protected wildlife reserve, with thick and humid forest, making it very difficult to walk through and even more difficult for hardware to survive the conditions. Cameras are placed along the river and up to 1km into the forest, capturing images when triggered and saving them onto SD cards. These SD cards are collected and stored at the field centre, where the images are manually collated and processed. The cameras are designed to have a battery life of three months but, due to the humidity, a battery life of three weeks is more realistic. In 2010, twenty cameras were deployed and half of them were inspected every two weeks, on a rotating basis. In that time, each camera can record more than a thousand pictures and the dynamic nature of the rainforest, such as the sun through leaves, falling trees and reflections in the water can cause the camera to trigger when an animal has not walked past. The events are known as *false triggers*, and they can make up to 70% of the images on an SD card and each of these must be manually processed.

Each trigger of the camera results in a set of three images being captured, this number

has been chosen by the researchers at DGFC after some experimentation. Three images allows for more than one image of slow-moving animals to be captured and, typically, allows for the middle image in the sequence to capture the body of fast-moving animals; with the first and last capturing the head and rear respectively.

We have used this scenario to test our hypothesis and implement a WSN that automates the collection, transmission, processing and storage of images, using LK to classify the data and prioritise the flow of information through the network, making more efficient use of the limited power and bandwidth available.

1.2 Research Contributions

Here we outline the main research contributions explained in this thesis. We believe that two main contributions have been made and our experimental results serve to support these contributions. Our primary contribution is that we propose a **novel tiered network architecture where each subsequent tier possesses increased knowledge-processing ability**, K-HAS, that utilises the local knowledge of its surrounding environment, users and previously sensed data to process observations within the network and prioritise the data according to the inferred classification.

Knowledge is pushed out to the edge of the network to allow the nodes that capture observations to prioritise the data based on its content. The knowledge processing capabilities increase with each tier as the data moves toward the centre of the network, allowing more detailed inferences to be made and data to be given a higher priority. This smart utilisation of bandwidth allows data to be delivered in an order that is more useful than chronological, delivering the most valuable observations first and using human feedback to learn how important these observations were.

In addition, K-HAS uses a feedback loop to dynamically update the knowledge base on every node throughout the deployment so it is able to react to changes within the data recorded, and the network, in near real-time.

Experimental results from the Malaysian rainforest showed that current long-range wireless sensor technology is not yet at the point where nodes could be deployed in a harsh environment and left without human intervention for months at a time. A modified K-HAS system LORIS (Local knowledge Remote-sensing Ontology-based Informatics System) combined tiers that use knowledge-processing at the centre of the network, with commercially available, wireless cameras. This showed that knowledge-processing automates the handling of sensed data when it is received and can be used to infer patterns for future observations. LORIS was designed because we could not find robust, open sensing nodes that can send data over long distances and handle high humidity and extreme temperatures. Because of this, we used off-the-shelf (OTS) wireless cameras that ran a closed system, while providing some access through proprietary software. We then combined two of K-HAS' tiers so that processing only took place at the base station, matching a more traditional WSN topology; while still making use of local knowledge gained from previously sensed data.

We model the ideal implementation of K-HAS in a simulation environment, along with variations on the knowledge processing capbilities of each node, We use these simulations to show that LK and GK can prioritise sensed data effectively and that this prioritisation increases the efficiency of the network by delivering data based on its importance, rather than chronologically.

Our second contribution is ~~what we believe to be the first~~ an **ontology that combines, and extends, ontologies used in multiple domains to create an extension ontology for WSNs and ~~scientific~~ ecological observations**. To formally define the structure of K-HAS and the data standard used, we developed an ontology that combined existing ontologies for sensor networks and ~~scientific~~ ecological observations. This ~~modular~~ ontology can be used as a whole or in parts for WSNs that use some, or all, of K-HAS' architecture. On top of this, we extended these ontologies by adding specific terms relevant to the different node types and users within the K-HAS architecture. ~~The extensibility of the ontology allows parts to be removed and replaced where necessary.~~

We apply this ontology to our proposed architecture, and simulations, in order to define a data standard for sensed data sent within the network, as well as terms to use in observations as well as identify the roles of both humans and sensors within the network.

1.3 Thesis Structure

The rest of this thesis is structured as follows. Chapter 2 provides background on wireless sensor networks and the use of knowledge-based technologies in that context. Chapter 3 explains technical decisions we made and the findings when running experiments in the Malaysian rainforest. Chapter 4 introduces the K-HAS architecture we have proposed and explains the purpose of each tier. Chapter 5 details the ontology we have proposed to support K-HAS and shows how current ontologies do not sufficiently cover all of the concepts involved with a *scientific observation*. Chapter 6 details our implementation in the Malaysian Rainforest and the changes we had to make in order for it to be feasible. Chapter 7 describes how we modelled K-HAS, and other scenarios, in a simulation environment. The results are then explained and we explain how different scenarios are suited to WSNs, in a multitude of environments, with different requirements. Chapter 8 then concludes this thesis and summarises our contributions and findings, as well as highlighting work that could be undertaken to take this project further.

Chapter 2

Background

Using knowledge in a WSN is related to existing research into sensor networks that utilise context-awareness in order to improve their effectiveness or adapt their sampling rate. An example of a context-aware sensor is an accelerometer attached to a node that is able to determine what the readings of the accelerometer indicate. For example, a smart phone with an accelerometer may use context-awareness to determine if the user is running, walking or going upstairs. Some of these actions may be more important than others and, thus, they can be prioritised. However, researching context-aware WSNs alone would limit our knowledge of WSNs in general and affect decisions we make when designing our own architecture. There is research that is valuable for all types of sensor network, such as: hardware design, routing protocol, tranmission medium choice or middleware used.

This chapter is split into the following sections. Section 2.1 outlines the issues surrounding WSN design and deployment. Section 2.2 details relevant existing routing protocols for sensor networks that are used for a number of different purposes, from extending network lifetime by scheduling sleep patterns to storing sensed data on nodes and responding to queries. Section 2.3 highlights commonly-used sensor middleware. Section 2.4 shows some examples of existing WSNs that are related to our motivating scenario. Section **??** introduces research into local and global knowledge and Section 2.5 summarises the results of our research.

2.1 Wireless Sensor Network Issues

WSNs have been used in a number of domains, for a range of different purposes, from habitat monitoring [**?**] to military purposes [**?**] and healthcare [**?**]. While these applications are different, the technology behind each is similar. Each requires the use of nodes with sensors attached and each node requires a power source and storage devices.

According to [15], there are at least eight factors that affect the design of sensor networks, but we focus on a subset that are the most relevant to our research problem. Those we have given less consideration to are: production costs, scalability and topology. While these are important, productions costs are not a concern for us in the research stage and scalability has been considered when studying existing networks. We have considered the following points in greater detail:

2.1.1 Fault Tolerance

WSNs typically contain a large number of nodes and any node can fail for various reasons, from a lack of power, filling its storage capacity, to factors in the environment causing the hardware to fail. While the hardware architecture of sensor nodes is typically similar, the variation between each deployment means that the device itself must be adapted to its environment. For example, [**?**] used a custom protective casing for their nodes so that they were able to survive being in the open while ensuring that the transmission range was not affected.

2.1.2 Hardware Constraints

A sensor node typically consists of: a platform that contains the memory and processing power, a sensor (or sensors) and a transceiver that uses a wireless standard, such as Wi-Fi or Zigbee. Cost and size are the most significant considerations when

designing a WSN. According to [**?**], the expectation of a sensor node is a matchbox-sized form factor. However, ten years ago, there was much research focussed on *smart dust* [**?**]. Smart dust is small, inexpensive, disposable nodes that can transmit until their power reserve is depleted, [16] highlights that it is a requirement for the nodes to cost less than USD10. In [1], a decade on from the first WSN papers, smart dust has not been realised and the focus has instead been on larger, more powerful nodes that have reduced in cost and grown in power. The Gartner Hype Cycle for 2013 [**?**] showed that smart dust is still in early innovation stages and may not be fully commercialised for another ten years. To counter this, research has been focussed on using software solutions to maximise the battery life in these more powerful, more expensive nodes, accompanied by the use of renewable energy sources.

2.1.3 Energy Constraints

Commonly, sensor nodes do not have access to a constant power supply and must run on a battery that is, generally, a similar size to the node. Larger batteries must usually be contained in a separate enclosure to the node and this makes nodes less compact and their deployment more difficult. This means that the nodes must be as efficient as possible, knowing when to transmit data and when to sleep. The lifetime of a sensor network is highly dependent on the battery life of each node and, unlike other mobile devices, they cannot typically be recharged [15]. Much work has been done on power efficient routing protocols, as well as the control of which attached devices are active [**? ? ?**].

The limited resources on the nodes mean that the sensing devices, and transceivers, attached must consume as little power as possible. Some routing protocols implement turning off wireless radios and scheduling a wakeup across the network [**?**], but the cost of turning off a device can waste just as much energy as leaving it on and sampling at a lower rate, if not more [**?**].

The use of energy in a node is dependent on how active the sensor(s) are, how much it transmits and receives, the transmission medium used as well as the environment it is in.

2.1.4 Transmission Medium

Widely used, general purpose transmission media, such as Wi-Fi, are viable solutions in WSNs when a high data rate is required and power is readily available. However, research has shown that Wi-Fi is extremely power-hungry and [**?**] shows that Wi-Fi consumes almost 9 times more energy, while transmitting, than other standards, such as Zigbee. Bluetooth is a more power-efficient standard that is becoming increasingly popular for sensing devices that are part of the 'Quantified Self' movement [**?**], with wearable devices that report measurements, such as heart rate, steps taken and calories burned. With the advent of the new low-power Bluetooh 4.0 , also known as Bluetooth Low Energy (BLE), this standard is supposed to allow months of continuous use on a coin-cell battery [**?**]. However, the theoretical max range is 100m and, using the same frequency, as Wi-Fi (2.4GHz) means that it is as susceptible to path loss and reduced transfer rates. [**?**] shows that a 2.4GHz Wi-Fi antenna is capable of transmitting up to 350m, while a considerably lower frequency of 41MHz was able to achieve links of 10km. The use of 2.4GHz frequencies in wet conditions have been shown to reduce the performance by up to 28% [**?**], while humidity in the air can reduce the performance by up to 78% [**?**]. New low-power, low-frequency standards have emerged in recent years and allow for a considerably longer range and increased battery life, at the cost of transmission speeds. Digimesh is an example of this and, while it can achieve 250kb/s using 2.4GHz, it has much slower speeds of 125kbps when using the 900MHz spectrum. However, it does offer a range of, up to, 64Km [20].

2.1.5 Environment

The environment that a node is deployed in can have a great impact on almost all aspects of a WSN, such as range or the operational lifetime of the node itself. Harsh environments that are not easily accessible make it difficult to place nodes and protected environments may limit where nodes can be placed. In [**?**], nodes were deployed within glaciers and had to survive extreme temperatures, lasting without human intervention, for months at a time. [**?**] attached collars to Zebras that had to withstand high speed movement, impact, dust and high temperatures. The deployment of any node requires extensive research as to the environment that it will be deployed in and adjustments must be made to ensure it is able to survive for extended periods without continued maintenance. Section 2.1.4 also shows that environment does not simply affect the hardware, but humidity can reduce the transmission range significantly, as well as moisture collecting on wireless antenna can reduce the range for days at a time.

Limited range in a WSN can be addressed by using *intermediate nodes* - nodes tasked with forwarding data from other sensing nodes that would otherwise be out of range - between disconnected sensing nodes. While they do not need to sense the environment directly, they can be vital in ensuring data from all areas of the network are ~~deliverd~~delivered. In the case of [**?**], a car was used to drive near to zebras that had not passed close enough to the base station in order to transmit their data. The car acted as the intermediate node by collecting data from the zebra's collar and relaying it back to the base station. In a normal WSN, this has the downside of requiring more nodes, which increases the cost and the chance of a node failing, but it can connect clusters in a geographical region that would otherwise be unable to send sensed data, or they can provide multiple routes through the network to help preserve battery life.

2.2 Routing Protocols

Routing protocols specify how nodes in a WSN are organised, as well as how they transmit data throughout the network. In [14], the more popular routing protocols are surveyed and split into three of the main identified categories: data-centric, hierarchical and location-based. We use the aforementioned categories, as well as flat, to highlight some of the key protocols that are relevant to our work. Flat routing protocols are for WSNs that involve many nodes deployed over a large area, all sending data to a single endpoint. Data-centric is a protocol that focusses on sensed-data, where nodes advertise their contents and query other nodes to fulfill requests made by a user. Location-based protocols use the physical region that nodes are deployed in to request and send data. Hierarchical protocols are for heterogeneous networks that perform more than one task. For example, a network spread of over hundreds of miles could be split into individual networks, or it could follow a hierachical structure where clusters of nodes send data to an gateway node (also known as a *cluster head*); a node that may not directly sense the environment but acts as an endpoint for a group of nodes [14]. The data then hops across other nodes in order to reach a final endpoint. The protocols have the task of ensuring that a network is performing at its best, providing the best lifetime and ensuring reliable and consistent delivery of data. This must reduce *flooding*, where nodes send every message to every link, aside from themselves, effectively flooding the network with unnecessary messages, and find a way to deliver data to the endpoint using the most efficient path possible.

2.2.1 Flat

Initially, this was the most common structure for a WSN, dozens of nodes spread out over a geographical area, with one or more neighbours, sending observations to a single endpoint.

MCFA

The Minimum Cost Forwarding Algorithm (MCFA) is a flat routing protocol that works by assigning costs to each node, based on how many hops they are from the endpoint [**?**].

Each node has a path-estimate of the cost of transmission from itself to the base station. The base station sends out a broadcast message and it is received by all nodes in range. The message contains a cost from the base station (initially zero) while every node has their cost set to infinity. The cost is stored at the node, incremented and sent on to all nodes in range. If the received cost is less than the current cost stored on the node, then the cost (and the neighbour) is updated and passed on.

Each message has a cost associated with it, which is based on the hops it has completed so far. A node that receives the message forwards it only if it's cost matches the sum of the source node's cost (contained within the message) and the message's current cost. This ensures that all messages are sent through the minimum cost path, without storing explicit path information on each node.

This approach allows for dynamic reconfiguration of the network, as well as a reduced overhead due to not having to maintain a global routing table on each node. The assumption with MCFA, however, is that the direction of routing is always towards a fixed endpoint.

2.2.2 Data-centric

Data-centric routing protocols are not like traditional WSNs where nodes are given addresses; they use a method that involves the advertising, or querying, of the data that has been sensed and those with the relevant data can respond to the request.

SPIN

Sensor Protocols for Information via ~~Negotation~~ Negotiation (SPIN) is one of the first data-centric protocols and attempts to address the issue of flooding the network whenever new data is sensed by addressing the data through metadata [**?**]. SPIN works on three messages passed between nodes:

1. ADV - A message sent by a node when it has sensed new data, advertising what it has recorded.
2. REQ - Sent by nodes that received an ADV to request the data.
3. DATA - Message containing the sensed data.

When a node has sensed data, it sends an ADV message to all nodes within range. If any of those nodes are interested in the data, then they respond with a REQ message, at which point the DATA message is sent to nodes that responded.

SPIN eliminates the need for a global view of the network topology, as nodes only need to know their single hop neighbours. However, SPIN does not guarantee equal diffusion of data throughout the network as a node may be interested in the data sensed at the other edge of the network, with only nodes that are not interested in between. This would mean that those nodes would not request the data or pass it on.

SPIN-IT

An extension to SPIN, SPIN-IT uses a slightly different approach to receiving data and was developed solely for the transfer of images [**?**], using metadata to fulfill requests.

Nodes use the existing message structure of SPIN, but REQ messages are used as queries, sent to all nodes in transmission range. The receiving nodes keep these requests and generate a new REQ message, thus allowing nodes to store temporal paths. When a

REQ reaches a node that has the desired data, it responds with a ROUTE-REPLY message. This message is used because images are large and resource-constrained WSNs would have a much shorter lifetime if a lot of unnecessary transmissions were made. The ROUTE-REPLY is used in case multiple nodes, in range of the requesting node, have the data it has requested and it can then choose the optimal route. As each node keeps a history of REQ messages, these can be used to trace the requested data back through the network, to the originating node, without the overhead of maintaining a global routing table.

COUGAR

A slightly different data-centric approach is the proposed COUGAR protocol, viewing the network as a distributed database. Cougar is similar to SPIN because it does not forward data as soon as it is sensed, instead COUGAR uses a query language that abstracts the underlying network structure and uses received queries to generate a plan that utilises in-network processing to provide an answer based on the sensed data stored on all deployed nodes [**?**].

For example, in a building monitoring network, a user could query a base station for offices that are unoccupied. The base station then sends that query to all nodes in range and it is then dispersed throughout the network. Nodes that have data that can satisfy the query send back their results and these are combined as they move back through the network to the base station. The user then receives that data, along with the nodes that have provided it.

Within the network, a *leader* is selected and this node is used to aggregate the data from nodes that were able to fulfil all, or some, of the query. At risk of failure, each query should result in a *leader* being dynamically selected and it must have sufficient resources to be able to satisfy the request. This protocol was only proposed, and much of the technical detail has yet to be completed, but the concept of treating the network as a distributed database is a novel idea and this is one of the first protocols to suggest

the use of a query language that could be used by people without specific technical knowledge, which, in this case, is technical knowledge of query languages and how the routing protocol works.

2.2.3 Hierarchical

Hierarchical networks are WSNs that contain nodes of different classes; nodes at the edge of the network are typically clustered into groups and served by a gateway node. This gateway could be in charge of aggregating the data, processing the data, or simply forwarding it to an endpoint. Clusters of nodes allow the network to be spread out over a wider geographical area and gateway nodes can use a different transmission method to provide long distance links to the base station. Gateway nodes serving a cluster of nodes means that the network can scale easily as well, simply by adding a new cluster to the network.

TEEN

The Threshold sensitive Energy Efficient sensor Network (TEEN) protocol is designed for reactive sensor networks, networks that require instant reactions to changes sensed in their environment [**?**]. TEEN recognises that transmission is the most power hungry action for a node so each node is coded with a hard and soft threshold. The hard threshold is a value that makes nodes transmit the reading to their cluster head. Similarly, the soft threshold is a small change in the value of the sensed attribute that causes further transmissions.

During the initialisation of the network, the base station sends information about the thresholds and sensing attributes to all cluster heads in the network; the cluster heads then forward this on to all nodes in their cluster. When a node senses data over the hard threshold, it transmits to the cluster and only transmits again when new sensed values

are greater than the hard threshold and the difference between the current sensed value and the previous is greater than the soft threshold [**?**].

Clusters are assigned for a period of time and then new clusters are selected by the base station, at which point new attributes and thresholds are broadcast to all nodes. This kind of protocol allows the network to be dynamic after deployment and allows user input based on the data that has been sensed in the previous cluster times.

For example, a network could be tasked with sensing humidity in a rainforest but the thresholds have been set such that nodes are transmitting readings that are not of interest. A user can change these thresholds and they will be pushed out to the nodes at the time that the next clusters are chosen, without any need to visit the node or configure them individually.

2.2.4 Location-based

Instead of using the physical addresses of nodes, or the data they store, location-based protocols are based on the region that nodes are deployed in. Location-based routing relies on the fact that each node is aware of its own location and is also aware of the destination's location.

Span

Span is a protocol where nodes are selected as *coordinators* based on their positions. A node can decide to be a coordinator based on the amount of energy it has and the number of neighbouring nodes it would benefit if they were able to use it as a bridge [25].

An example of this would be node B placed between node A and C. C and A are unable to communicate directly so, when node B wakes up, it decides whether it should become a coordinator. It knows that it has sufficient energy levels and it can provide

connectivity for a previously disconnected area of the network, so it chooses to become a coordinator, staying awake and routing sensed data to other coordinators, which form the backbone of the network.

Results showed that using Span, in a system that transmits using 802.11, provides an network lifetime increase of more than a factor of 2 over networks that just use the 802.11 protocol.

GEAR

The Geographic Energy-Aware Routing protocol (GEAR) is similar to SPAN in that it makes routing choices based on both energy-awareness and location. Each node maintains an *estimated cost* and a *learning cost* of forwarding a packet through its neighbours. The estimated cost is calculated using the distance to the packet destination and the energy remaining on the node whereas the learning cost is the estimated cost that takes holes in the network into consideration [**?**].

GEAR is designed to perform in two phases: forwarding a packet towards a region and disseminating a packet within a region. When sending a packet towards a destination, GEAR either sends a packet on to the node in range that is closest to the destination or, if such a node does not exist, then a hole is identified. If a hole is identified then the node that minimises a cost is selected.

To disseminate a packet within a geographic area, GEAR uses algorithms based on the density of the network. Recursive geographic forwarding is typically used but this can result in an endless loop if the density of the network means that the region is unable to contact the destination. In that case, restrictive flooding is used.

2.2.5 Conclusion

Routing protocols can define the topology of a network, how data is sent and have an impact on the network lifetime by determining when nodes should sleep, when they should transmit sensed data and when (or if) they should request updates on the network topology. Researching these types of routing protocol allows us to determine the situations in which each category would be used and whether they fit our requirements. The routing protocol must be considered when developing a network architecture, so we have researched the four main categories and the more commonly-used protocols within those categories. If we chose a flat network structure, then MCFA would be the better choice but our architecture changes significantly if we choose to use a data-centric protocol.

Using this background knowledge, we can select a protocol that fits with our needs, or combine useful aspects of many. Data-centric protocols are useful for battery conservation and networks that do not require real-time reports. Location-centric is useful as it does not require specific node addressing but location-aware nodes are more expensive and power-hungry. Hierarchical, however, allows for in-network data-aggregation or processing but requires the use of a heterogenous network with a more rigid topology of nodes split into clusters.

With the knowledge of the requirements of a WSN architecture, we can pick the routing protocol that best suits these, or combine traits from multiple protocols in order to develop a hybrid protocol that fits our requirements exactly.

2.3 Sensor Middleware

Acting as a bridge between the hardware and the user, sensor middleware is software that abstracts the underlying network from the user and provides a means of accessing sensed data and administrating how the network performs [**?**]. These middlewares

must not be specific to a single network and provide support for as many different sensor nodes as possible. In [**?**], a middleware is said to provide standardised services to many applications and perform operations that make effective use of limited system resources.

In [**?**], a middleware should include four major components: programming abstraction, system services, runtime support and Quality of Service (QoS) mechanisms. In this section, we will discuss the challenges surrounding middlewares for WSNs and highlight some existing middleware that are particularly relevant to our research problem and motivating scenario.

2.3.1 Issues

WSNs present a range of new challenges to existing middleware, due to their resource constraints, deployment environments and more. However, there has been research into the key issues that must be addressed in order for middleware to be considered suitable. While there have been a number of surveys into these challenges [**? ? ?**], we will detail those that we believe to be most relevant to our work. Some of those that, while considered, have not been a primary issue are: dynamic network organisation, security and application knowledge.

Energy Constraints

It is rare that nodes in a WSN would have a constant power source, unlimited memory and a casing that can survive a harsh environment without decaying. In order to ensure that the lifetime of nodes is maximised, middleware needs to offer a power scheduling system that makes efficient use of the hardware on the node, generally disabling the radio after a set interval has elapsed, or use a combination of lower power sensors to provide sensed data of a similar quality [**?**].

Ideally, a middleware will be able to coordinate nodes through wireless communication, making efficient use of transmissions and dynamically modifying sleep schedules based on the power remaining.

Heterogeneity

Not every node in a network will have the same capabilities, manufacturer or hardware. WSN middleware needs to provide a standard interface to the applications that are making use of it, whilst at the same time accommodating differences arising from hardware coming from different manufacturers and perhaps having significantly different capabilities. Some middleware have been built for a specific set of hardware [**?**], however this homogeneity can provide an increase in the performance and efficiency of the network by only supporting a limited number of devices.

Real-world Integration

WSNs are often tasked with recording phenomena that are time-crucial, so a sensor middleware should provide a real-time interface to the data that it has sensed [12]. Ideally this data would be available outside of the network, though the use of an API.

Quality of Service

This issue is perhaps the most complex as QoS could apply to almost all aspects of the networks, such as efficiently using bandwidth, 100% uptime for nodes, guaranteed packet delivery or access to data stores. Some of these requirements are managed by the implementation of the routiong protocol, the middleware should be able to monitor deployed nodes and report on their current status, as well as identify failures.

2.3.2 Existing Middleware

In this section, we identify existing middleware, explain how they address the issues highlighted in Section 2.3.1 and highlight how they relate to our research. While there are a lot of existing middleware, our research did not show any that used information abouts its environment or knowledge from previously sensed data to process data that is currently being sensed. We did, however, find some middleware solutions that utilise context and rules to administrate the network.

GSN

The Global Sensor Networks middleware (GSN) has been developed to manage heterogeneous sensor networks and be suitable for those without any technical knowledge [12].

GSN provides hardware abstraction through the use of *virtual sensors*, a data stream that abstracts implementation details from the actual sensed data. A virtual sensor can be comprised of many streams and it can even consist of many virtual sensors.

Virtual sensors are described using XML, with tags that consist of metadata for the sensor, the structure of the incoming data stream, SQL queries for processing the incoming data and querying times. What makes GSN stand out is that virtual sensors do not have to be sensors deployed within your network, or even sensors at all, some examples of GSN show virtual sensors being added that read in data from the weather websites. This also means that the underlying structure of the network is irrelevant to GSN, as well as the physical locations of the nodes. Unlike some middlewares that have an expectation of how data will be routed, GSN is decoupled from the routing protocol, allowing them to act independently.

GSN is completely open source and the Java code can be modified to suit a specific deployment.

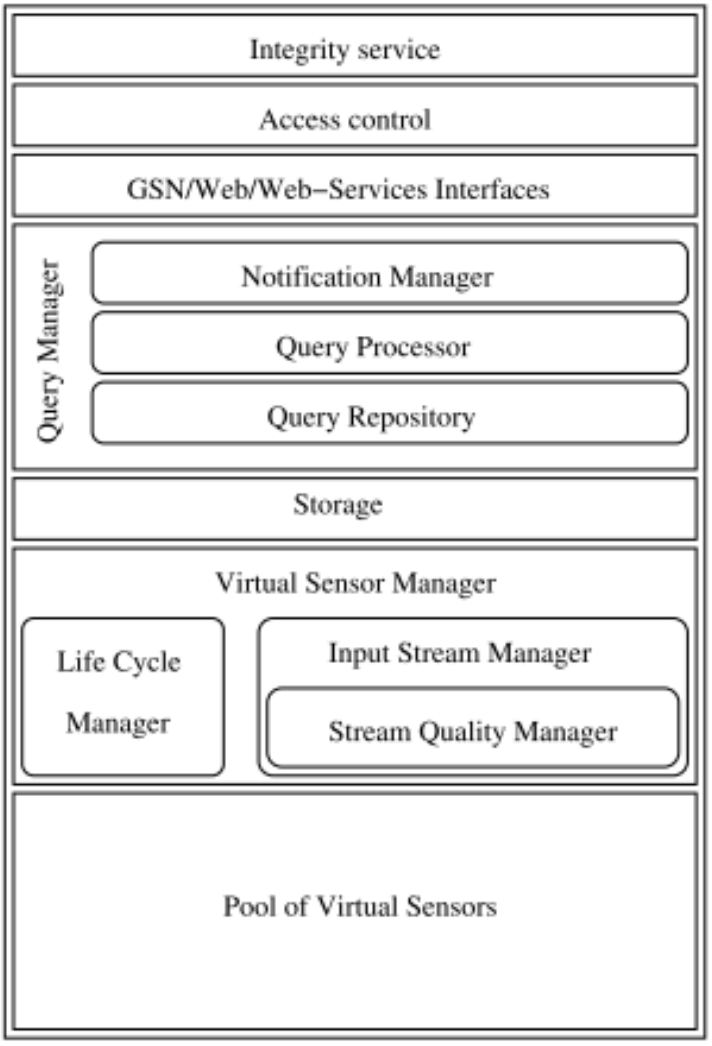


Figure 2.1: GSN Architecture [12]

Figure 2.1 outlines the architecture of GSN, showing that the virtual sensors are stored on a central node and their inputs are managed and stored. GSN also comes bundled with a web interface to show all active sensors and their most recent recordings, as well as the implementation of web services to access the data outside of the interface.

Data from virtual sensors pass through the virtual sensor manager to the storage layer. Once the data has been stored, the query manager is invoked and queries are loaded from the repository and executed by the manager. The results of the queries are then handled by the notification manager and also made available to the web interface. No-

tifications can be extended to support many different forms of communication, such as SMS, email or web services.

Virtual sensors do not natively support all hardware, although new virtual sensors can be described using XML, and there may be a need to implement an entirely new virtual sensor. In this case, technical knowledge is required, and new sensors can be implemented through use of the Java programming language. This provides more control over the use of XML and allows users to specify how sensed data is stored in a database, use external libraries to receive proprietary data, specify processing workflows before the data is stored or implement new notification methods for the users of the network.

To show the simplicity of a basic virtual sensor, [11] describes a temperature sensor that we reproduce here in Listing 2.1. The file is human-readable and does not require specialist knowledge when compared with programming languages, with tags that have self-explanatory names. In this example, the output structure shows that only a temperature reading is received and that the data should be stored permanently. The stream source specifies the content of the stream and the query details the standard query that should be used to extract data from GSN.

```
1  <life-cycle pool-size="10" />
2  <output-structure>
3    <field name="TEMPERATURE" type="integer"/>
4  </output-structure>
5  <storage permanent-storage="true" size="10s" />
6  <input-stream name="dummy" rate="100" >
7  <stream-source alias="src1" sampling-rate="1" storage-size="1h">
8  <address wrapper="remote"> <predicate key="type" val="temperature" /> <
       predicate key="location" val="bc143" /> </address>
9  <query>select avg(temperature) from WRAPPER</query>
10 </stream-source>
11 <query>select * from src1</query>
12 </input-stream>
```

Listing 2.1: Example Virtual Sensor

The modularity and flexibility of GSN makes it different to existing ~~middlewares~~ middleware as it has not been designed for any specific hardware and modules of the middleware can be replaced, such as the database.

FACTS

One such example is the FACTS middleware, an approach that uses a fact repository to coordinate nodes. Rules can then be implemented to process sensed data and fired when certain conditions are met [**?**]. More traditional sensor middleware controls the network and manage sensed data but this rule based approach allows for more flexibility, where rules can control the transmissions and process the data upon receipt.

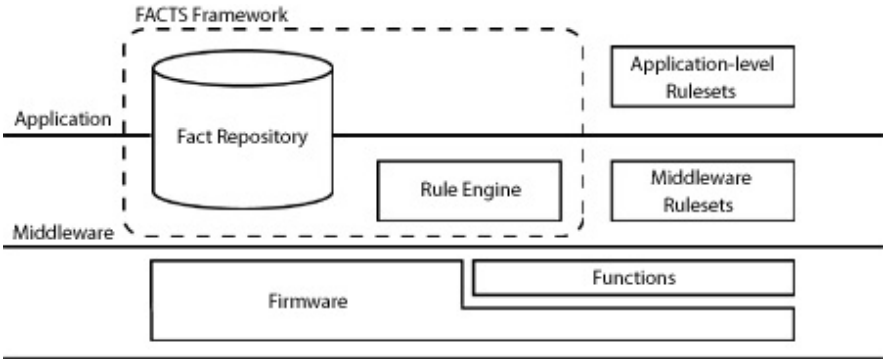


Figure 2.2: FACTS ~~Architecturei~~ Architecture [?]

Figure 2.2 shows the FACTS architecture, with the middleware holding the ~~rulesets~~ rule-sets and a distributed fact repository. Data within the network is stored as facts, providing a standard data format throughout the network and hardware abstraction. When new facts are received, ~~ususally~~ usually because of new sensing data, the rule engine checks to determine whether any rules should be fired. The ~~ruleset~~ rule-set definition language (RDL) is used here and each ~~ruleset~~ rule-set contains a group of relevant rules. Each rule is given a priority so that, if more than one rule is triggered by a fact, then the higher priority rules are fired first.

Listing 2.2 shows a FACTS rule, written in Haskell, that determines which geographic areas are covered by nodes. The name of each rule has no prefix, statements are prefixed with '->' and conditions are prefixed with '<-'. The *sendRange* rule runs after a timer expires and removes that timer. Lines 4 to 12 set a *rangeFact* fact that contains the range it expects to be able to cover. The rest of the rule sends the fact to all nodes within range and sets a new fact to show that it has sent its range information.

The (xyMinCovered) is a simplified rule that, upon receipt of range information from neighbouring nodes, checks whether the range of the neighbouring node overlaps with its own range and the final *determineCoverage* rule then stores that coverage information in a fact as knowledge of whether or not it is the only node to cover a particular geographic region. This can then be used to inform future routing and sleeping decisions.

```
 sendRange
 <- Exists Timer.expiredSlot
 -> Retract Timer.expiredSlot
 -> Define "rangeFact"
 -> Set ("rangeFact" "xMin")
   (posXSlot - System.txRadiusSlot)
-> Set ("rangeFact" "xMax")
   (posXSlot + System.txRadiusSlot)
-> Set ("rangeFact" "yMin")
   (posYSlot - System.txRadiusSlot)
-> Set ("rangeFact" "yMax")
   (posYSlot + System.txRadiusSlot)
-> Send 0 System.txPowerSlot
   ("rangeFact" [(("rangeFact" "owner")
   == nodeIDSlot)])
-> Define "rangeSendFact"

xyMinCovered
<- Exists "rangeSendFact"
<- Eval ((posXSlot - System.txRadiusSlot)
   < ("rangeFact" "xMin"))
<- Eval ((posYSlot - System.txRadiusSlot)
   < ("rangeFact" "yMin"))
-> Define "xyMinCoveredFact"

determineCoverage
<- Exists xyMinCoveredFact"
-> Define "coveredFact
```

Listing 2.2: Coverage Algorithm in FACTS Rules

While FACTS itself does not utilise any local knowledge, the repository is used as a source for all previously sensed data and would be an excellent source of knowledge to assist with the classification of future readings. Also, the ability to add new ~~rulesets~~rule-sets, without technical knowledge of the hardware of each node means that users of the network have the ability to add knowledge in the form of less technical, high-level rules.

ITA Sensor Fabric

The ITA Sensor Fabric is a collaboration project between IBM, the US Army and the UK ~~MoD~~Ministry of Defence. Sensor Fabric, or Fabric, is a two-way messaging bus

and set of middleware services connecting network assets to each other and users [**?**].

The core difference between the Fabric middleware and others is that not every node is sensing all of the time, sensor nodes are tasked when there is a requirement and they stop as soon as that task has been fulfilled. Similar to sinks in a traditional WSN, Fabric utilises Fabric nodes, which run the following three pieces of software:

1. Message Broker - Provides the communication infrastructure.
2. Fabric Registry - Holds information about the current deployment, such as all nodes deployed, all assets, routing information and tasks. Deployed in the form of a database.
3. Fabric Manager - The main service on the node to track the status of connected sensors, establish communication channels, provide a container for processing, plug-ins and to extends the capabilities of the Fabric.

Fabric runs on a Publish/Subscribe model, a sensing requirement is sent to a messaging broker as a subscription and this is distributed through all Fabric nodes and, thus, all sensor nodes. Sensor nodes then publish their data and the relevant data is sent to all applications that have subscribed to the data.

The plugin structure of Fabric makes it stand out from existing middlewares, allowing its functionality to be extended through web interfaces.

Because Fabric has been developed for military purposes that cross countries, policy enforcement has been implemented to restrict access to the granularity of sensed data but these access levels do not simply apply to a military context. Using our motivating scenario, researchers and professors should see animal images whereas the Sabah Wildlife Department should see images of hunters and people in the forest.

Sensor Web Enablement

The Open Geospatial Consortiums's (OGC) Sensor Web Enablement (SWE) is a set of standards to allow developers to make sensors and sensed data repositories accessible via the Internet [**?**]. While this is not a middleware, these standards can work with existing middleware in order to publish their sensed data to the Internet. The SWE framework consists of:

- Observations & Measurements (O&M) - General models and XML encodings for observations and measurements.
- Sensor Model Language (SensorML) - Standard models and XML schema for describing the process within sensor systems.
- PUCK - Defines a protocol to retrieve a SensorML description and other information from a device, enabling automatic sensor installation, configuration and operation.
- Sensor Observation Service (SOS) - Open interface for a web service to obtain observations and sensor descriptions for one or more sensors.
- Sensor Planning Service (SPS) - An open interface for a web service that a user can determine the feasibility of ~~collection~~ collecting data from one or more sensors and submit collection requests.

These components can be integrated into existing middleware, clients and sensors and not all need to be used, some middleware, such as MufFIN [**?**], have just used SOS and O&M but 52North have developed an open source implementations of all components of SWE [23].

Internet of Things

In [**?**], the Internet of Things (IoT) is described as the "concept of pervasive things... which, through unique addressing ~~shemes~~schemes, are able to interact with each other and cooperate with their neighbours to achieve common goals." The number of devices that are part of the IoT grows daily, from thermostats and smoke detectors [8] to activity monitors [4]. Some of these devices have their data siloed away and it is only accessible through a web interface or a mobile app, but most have an Application Programming Interface (API) that allows the raw data to be access and queried. Because of this, sites like If This Then That (IFTTT) [6] have been created to link together various services and use simple if statements to act on changes in the data. IFTTT uses *recipes* that follow the rule of: `IF Service1 changes THEN perform action on Service2`. For example, `IF I take a picture using my phone THEN upload it to Twitter` or `IF a new item appears on my RSS feed THEN add it to my reading list`. While, IFTTT is not a traditional middleware, it does monitor user's connected services, some of which may be sensors around the home or attached to their person.

A more traditional middleware for the IoT is Xively, a cloud platform that allows devices to upload their sensor readings to an endpoint and view readings from others [10]. Feeds can be public or private and Xively can be used with almost any Internet-enabled devices, such as webcams, temperature sensors or alarms. Other sites, such as Dweet.io [3] and Open Sen.se [9], have also been in operation for a few years that provide a similar service to Xively. Open Sen.se combines the recipes of IFTTT with the storage of Xively to create a middleware that supports actions being performed when certain sensed data is received. For example, a temperature sensor could trigger an alert on Open Sen.se that would alert a user via e-mail, or through social media. This combination of devices with online services and applications allows a middleware to do more than simply store data, it can execute rules over the data, provide visualisations and create links between devices separated by hundreds, or thousands, of miles.

2.4 Biodiversity and Environmental Monitoring Sensor Networks

In this section we will cover existing WSNs that are related to our motivating scenario or, more specifically, biodiversity focussed WSNs that have been deployed to monitor wildlife and/or the environment. WSNs for habitat, and wildlife, monitoring are especially important because these are areas that often need to be untouched by humans. Areas with high human disturbance can influence the abundance of species and some habitats, i.e. underground burrows, may be impossible to monitor without destruction.

One of the most well known WSNs to monitor habitat is the network deployed on Great Duck Island, an island off the coast of Maine, USA. A hierarchical network of 32 nodes was deployed to monitor a bird, known as the leach's storm petrel [**?**]. This network used a clustering approach for groups of nodes to send data to a gateway node, which would then route it back to the base station. The base station, located a few kilometres away on the island, has internet access and uploads the data to allow users to browse and process the data.

A multihop approach was used here as they found that, for sufficient coverage, single hop connectivity would not cover all of the island. Acrylic enclosures were developed to ensure the nodes were weatherproofed for the conditions of the island, while maintaining the functionality of each sensor and not impeding transmission range. While the nodes, their casing and their sensors have been designed specifically for the deployment on Great Duck Island, the success of the network, running for 123 days in the early stages of WSN research [**?**], shows that this approach can be used elsewhere with similar effects; allowing hard to monitor and/or inaccessible areas to be continuously monitored.

On a smaller scale, INternet-Sensor InteGration for HabitaT monitoring (INSIGHT) is a single-hop WSN that allows remote access for data and reconfiguring of nodes [**?**]. Using off the shelf hardware, their findings show that their nodes could survive for 160

days on a single battery, supporting their claim that a single hop network allows for a longer network lifetime.

The key feature of this network is the ability for humans to remotely set reporting thresholds for sensor nodes. This means a user can prolong the lifetime of nodes by limiting the threshold they report on, as well as the fact that these thresholds are a way for users to add knowledge, albeit primitive, into a network.

While there is research on cameras used to monitor animals [**?** 13], these networks are generally cameras deployed with their memory cards manually retrieved and processed. In recent years, however, the use of wireless technologies and image-based WSNs has increased, [**?**] uses wireless cameras to monitor the movement of animals between roads. Using commercial hardware and controlled sleep scheduling, this solution employs the use of nodes to detect movement and wake up more power-hungry camera nodes. While the nodes are wireless, the distance of the network from civilisation means that the data does still need to be collected manually and uploaded to a computer.

Due to the advent of smartphones and tablets, as well as the improvements in 3G technology, projects taking advantage of more modern technologies have grown in popularity. Using 3G enabled cameras, [**?**] have deployed a number of devices in locations all over the world, such as: Kenya, Indonesia and the USA. The images captured are transmitted to a server and a website allows the general public to not only see the images in near real-time, but to classify the images as well. This ~~crowdsourcing~~ crowd-sourcing of collective knowledge lets people, that may not have domain knowledge, vote on an image and those votes are used to make classification easier.

Over the past fifteen years, WSNs have grown from a concept to a real solution for monitoring the habitats, movements and eating habits of wildlife all over the world. Whether it is using GPS collars to monitor the movement of cattle [**?**], monitoring animal habitats on a remote island or using cameras to capture the animals themselves, the popularity of these networks has grown considerably and advances in technology

have allowed these networks to be deployed in places that humans cannot access with ease.

There are a number of situations where we may want to record data in an environment that is not safe for humans, such as a volcano [**?**], or that may be difficult for electronics to survive. In these cases, special considerations must be made during the design and deployment of the WSN, in order to ensure maximum network lifetime with reliable readings.

GLACSWEB

Glacsweb is a sensor network to monitor the rate ~~that~~ at which glaciers are melting [**?**]. Deployed in Norway, specially designed sensors have been drilled into glaciers to monitor pressure, temperature, orientation and strain. Due to the high pressure and exposure to moisture, a polyester casing was used so that, once bonded, the node inside would be protected from its environment, but also preventing it from being recoverable.

Reventador

Volcano Reventador is in northern Ecuador and a WSN has been deployed on there to monitor eruptions. Similar to Glacsweb, weatherproof enclosures were used to prevent ash and moisture from breaking the sensors and long-range external antenna mounted to a pole was used to achieve communications over large distances when wireless communications have proven to be difficult [**?**].

Rainforest

There have been many studies on how the rainforest affects the range and quality of wireless links [**?** **?** **?**]. In [**?**], the humidity was shown to reduce 802.11 range by up to 78% and [**?**] explains that periods of rainfall reduce the link quality up to 100% in

some cases, resulting in the loss of a hop and this could prevent data from reaching the endpoint.

These examples highlight some of the difficulties of deploying WSNs in harsh environments. Environments vary from freezing glaciers to humid rainforests to dry deserts with extreme temperature variation and the sensors cannot disturb their surroundings, be too conspicuous or require regular human attention that would disturb the area and affect the wildlife.

2.5 ~~Local and Global Knowledge~~

~~The environment of a sensor network is rich and varied and we believe that knowledge of this environment and patterns in the data sensed can be used to inform the network on decisions surrounding the transmission and processing of newly sensed data. As our research began, we simply called this knowledge but, as our work continued, it became apparent that it could be catgorised further.~~

~~While we believe that we are the first to use the concept of local and global knowledge within the wireless sensor network domain, the terms have been around for many years. In 1999, a book that referred to local knowledge as *indigenous knowledge* defined local knowledge as 'systematic information that remains in the informal sector, usually unwritten and preserved in oral traditions rather than text' [**?**].~~

~~Over the past twenty years, local knowledge has been used in various contexts, from researching lending and the credit market [**?**] to extracting local knowledge from natives to improve farming techniques [**?**]. This research, as well as work that will be covered later, showed us that there are two kinds of knowledge: global and local.~~

~~It was from agriculture research that we were able to refine our definition of local knowledge, [**?**] defines local knowledge as 'knowledge that farmers have derived locally through experience and experimentation'. They also say that indigenous knowledge is~~

different in that it is culturally specific. From this definition, as well as our work with our motivating scenario, we were able to generalise the definition and expand upon it.

We now define local knowledge as *knowledge of an area, held by a domain expert, that has been gained through experience or experimentation*. This then means that global knowledge is *knowledge of an area that can be accessed by anyone, without the need to visit the area directly*. The weather of a region is global knowledge because it can be found through a variety of media, whereas the saturation levels of the soil in a field would be local knowledge as it would require experimentation to gain that knowledge. Local knowledge could become global if it was readily available but some of these data are held in tribes of people without Internet access.

Using these definitions, we believe that **utilising local and global knowledge within a sensor network can inform routing decisions to make better use of the bandwidth in resource-constrained WSNs by sending data that the node believes to be important first, rather than chronologically**. Patterns in the data, and knowledge of the environment surrounding a node, can allow a node to infer what data may be valuable, automate the processing of adding context to the sensed data, learn from previously sensed data and utilise global knowledge of ongoing projects within the network to determine what data is thought to be of a higher priority.

2.5 Conclusion

In this chapter we have explored the components that make a WSN, as well as some existing deployments that are relevant to our motivating scenario. Sensor middleware have come a long way in the past decade and increased capabilities for sensor nodes have allowed for more intense processing to be carried out before any data is seen by a human. Context is now used on sensor nodes to infer the activity they are undertaking or even to determine when it should wake to sample.

Local knowledge is not a new concept and it has been used for many years to extract

information from indigenous people and in industry to adapt their processes to a local area, such as farming. However, we believe that our work is the first to apply local knowledge in the context of WSNs. Context-aware sensor networks have gone some way to support our hypothesis that some knowledge can increase the functionality of a network and enhance the quality of the sensed data, but we believe that the addition of LK and GK will only increase this functionality further.

The two primary components of a WSN that could be injected with local knowledge is the middleware and the routing protocol, each providing different benefits. Existing work has shown that the use of context-awareness in sensor middleware allows them to make dynamic, global changes to the network (Section 2.3), such as power management, whereas routing protocols affect the data that is sampled and sent to the endpoint(s) of the network, such as adaptive thresholds.

One issue in proving this hypothesis is the deployment of a network that utilises local knowledge. The deployment environment of our motivating scenario is not only interdisciplinary, it is in a region that is humid, dynamic and dense. Existing research has shown that the deployment of a WSN in these conditions means that range will be greatly reduced [**?**] and changes in humidity can prevent communication altogether, as well as moisture affecting the hardware itself. To deploy a network, hardware must be adapted and the right medium must be chosen in order to maximise link quality and minimise dropped connections.

In Chapter 3, we describe how we used this research, along with our own findings, to choose suitable hardware for our network. As well as this, we used the results from previous rainforest range tests in the literature, to design experiments that would aid us in choosing, and verifying, our choice of ~~tranmsission~~ transmission medium.

Chapter 3

Exploratory Experiments in a Rainforest Environment

In this chapter, we explain our motivating scenario in more detail, explore the sensor hardware and software that we researched and outline the results of experiments we undertook in the Malaysian rainforest. As highlighted in Section 1.1, we have been working with Cardiff University School of Biosciences to design and deploy a WSN that utilises local and global knowledge, using an area of rainforest in Malaysia owned by the Sabah Wildlife Department, called Danau Girang.

The structure of this chapter is as follows. Section 3.1 explains what we aimed to deploy in Danau Girang and what our considerations were. Section 3.2 introduces sensor hardware that is in use today and details the choices we made. Section 3.3 details the transmission medium choices we tried and also shows the results of experiments performed in both the UK and Malaysia. Section 3.4 explains some of the software applications we developed in order to process sensed data on the nodes. Finally, Section 3.5 summarises our findings and explains the choices we made for the sensor nodes we used in Danau Girang Field Centre (DGFC).

3.1 Danau Girang Field Centre

Based in Sabah, Malaysia, Danau Girang Field Centre is located in Lot 6 of the Lower Kinabatangan Wildlife Sanctuary (LKWS), surrounded by secondary rainforest that had been logged up until the 1970s. Experiencing a wet season from October to February and a dry season from March to September, the LKWS can receive more than 500mm of rainfall during the rainy season, dropping to lows of around 150mm during the dry seasons [**?**], and up to 100% humidity all year round.

Figure 3.1: The Main Buidling at DGFC

Danau Girang is in a unique location, situated in a rainforest corridor (figure 3.2) that joins two areas of rainforest together, with the corridor surrounded by palm oil on each side. Because of this, animals use the corridor to move between the rainforest regions and some use it to enter the palm oil plantation for new feeding grounds. This gives the DGFC insight into the movement patterns of these animals in the corridor as well

as in the rainforest itself, with a wide variety of species that are not commonly seen in other tropical regions of the world. Due to the remote nature of the centre, power is provided by a set of diesel generators which, typically, provide power from 10 am to 1 pm and 5pm to 11pm daily. Wireless Internet access is provided by satellite with speeds approximating 56kbps, although the upload speeds are considerably faster than downloads.

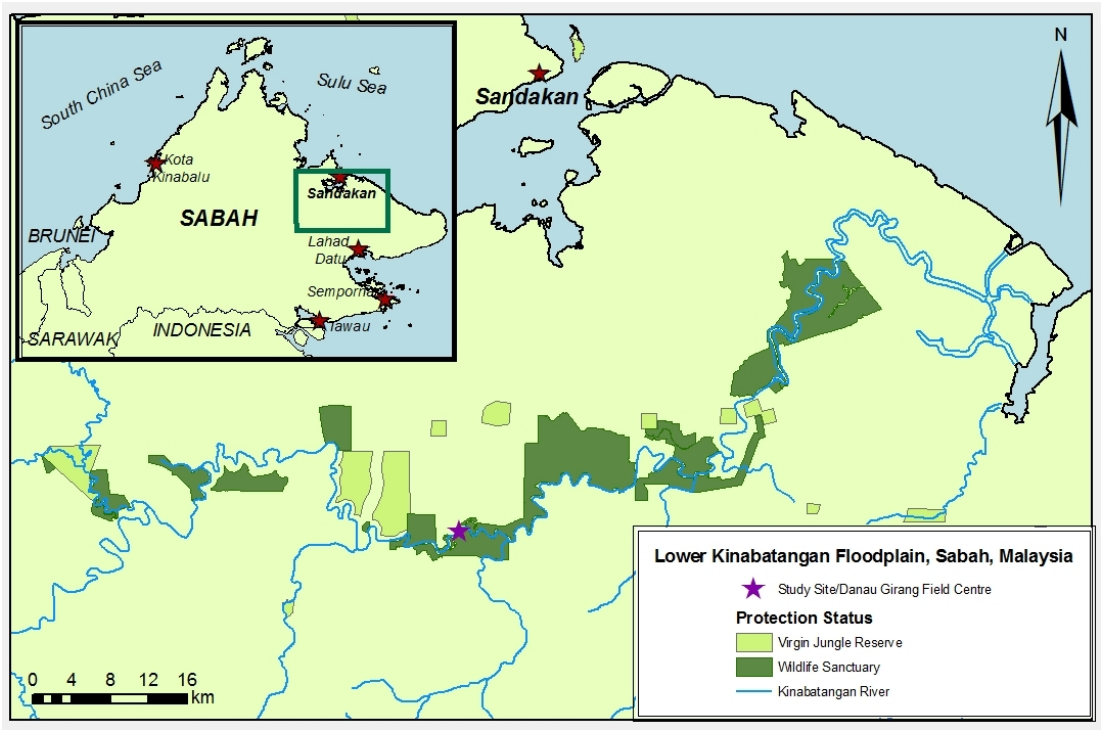


Figure 3.2: ~~Aerial View~~ Map of Danau Girang ~~Field Centre and the Surrounding Area~~in relation to Sabah Malaysia.

The corridor monitoring programme is a scheme that has been in place for more than five years, using wildlife cameras to track the movement of animals through the rainforest corridor and to capture species that are rare or unique to South-East Asia, such as the Bornean clouded leopard. Currently, Reconyx Hyperfire HC500 (Figure 3.3) cameras are being used [1]. These are standalone cameras that capture a user-defined number of images which are saved to an SD card. A trigger is caused by an infrared (IR) motion sensor when an object causes the IR beam to break.

Figure 3.3: Reconyx HC500 Camera

Images must be manually collected every two weeks from the cameras and the batteries are changed at that point as well, although a typical charge should last three months. In the field, the batteries last a matter of weeks shortly after the cameras are deployed, this is most likely because, although the cameras are equipped with watertight casing, they are still exposed to moisture when the case is opened every two weeks and 100% humidity seems to have an impact as well. Silica gel is used to prevent moisture inside the camera. We believe that humidity also reduces the battery life, as the charge drops from three months to around three weeks within the first few months of usage. However, the lack of constant power availability could also affect the charge of the batteries. Because power is not available all day, the batteries are charged for a period of a few hours at a time. Not only does this reduce the number of charge cycles that these batteries can experience but it also means that they are never fully charged within one time period of power availability. Solar recharging is being researched to counteract this. Each camera is secured to a tree and~~at higher risk~~ , when placed in higher risk

locations, such as known elephant paths, have protective cases as well.

In 2010, twenty cameras were deployed for six month periods and then relocated based on the needs of the projects at that time. Currently, there are now ninety cameras with a view to expand and dozens of projects within the field centre use the images gathered from these cameras. Initially, it was the job of visiting research students to collect the images but, since the number of deployed cameras has grown, full-time staff have been taken on to maintain them.

Images that have been collected are stored on an external hard drive in the field centre and a spreadsheet is updated with information about each image, such as date, time, camera, filename and location. Research students and, later, staff, manually inspected each image and added their classification to the spreadsheet. Because so many images came through the field centre each week, this was a long process and it was not uncommon for it to be a few weeks before the images were classified. Research students in charge of the cameras were usually on projects lasting 6 months so, until staff members were recently employed, their methods of cataloguing were quite different. Sometimes, each student created a different spreadsheet for their time at ~~DG~~ DGFC and ignored the classification of images that did not contain the animal their project was focussed on. While this has improved now, staff members are tasked with combining these spreadsheets from the past few years and classifying all the images that have been left empty.

With image processing, we can automate the classification process. Using common tools and widely accepted methods, we can use existing images (that have been classified) to create templates and match new images to these templates and classify them, when possible. Humans can then check the classification and provide feedback on its validity~~, but it~~ . This removes the wait of weeks for a classification and automates the cataloguing, classification and storing of the images, when received at the field centre. This method should also allow us to alert researchers when our system believes an image of particular interest, such as a rare animal, has been taken.

Our belief is that we can use the ~~LKWS~~Lower Kinabtangan wildlife sanctuary (LKWS), and the locations of the existing cameras, to deploy a WSN that uses local knowledge gained from the researchers at DGFC to automate the collection of images, improve the battery life by not exposing the internals of the camera to the elements so often and, most important, prioritise the flow of data through the network by in-network processing.

Three annual visits, in 2011, 2012 and 2013, each lasting three weeks, have been made to DGFC to test out hardware, software and wireless choices, in an effort to optimise the network. These visits have also been used to extract local knowledge from the area and researchers, by semi-structured interviews and watching them work.

3.2 Hardware

Before any visits were made to DGFC, meetings with staff members of the field centre were held in order to gain a better understanding of the environment and the project. This is where we were alerted to the humidity of the region and the fact that the failure rate of the Reconyx cameras has been as high as 30%.

Reconyx cameras have no external interface support and the only way to access the images is through the removable SD card. Because of this, there is no way of attaching external sensor hardware to the existing cameras. In-network processing is an important requirement for our WSN and this did limit our choices to nodes that are computationally capable, e.g. the Pandaboard [7], than more common sensors, ~~such as~~ e.g. the IMote 2 [**?**].

In this section, we detail our research into suitable sensor hardware that met the following requirements:

1. Able to perform processing of images and metadata
2. Common interface availability (Serial, USB)

3. Wireless enabled

4. Battery-powered

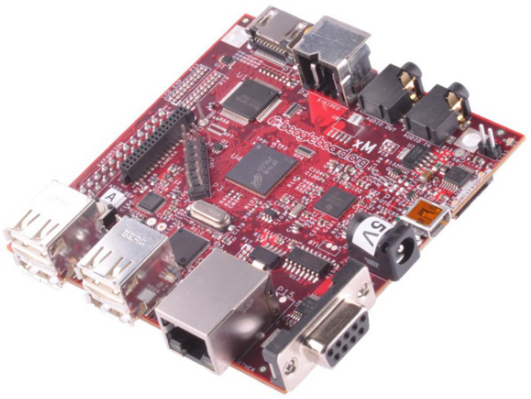
5. Expandable memory

It should be noted that there are many devices out there that met the above criteria but we selected devices that had a large development community. At the time of writing, there are now many more, but this research was carried out in 2010, when the market for low-cost, high-power, small form factor single board computers (SBCs) was starting to grow. While we did research on many more devices, the three listed below are the devices we physically trialled and tested. This section also details the modifications we made to any devices we tested in order to ensure they would survive in a humid environment.

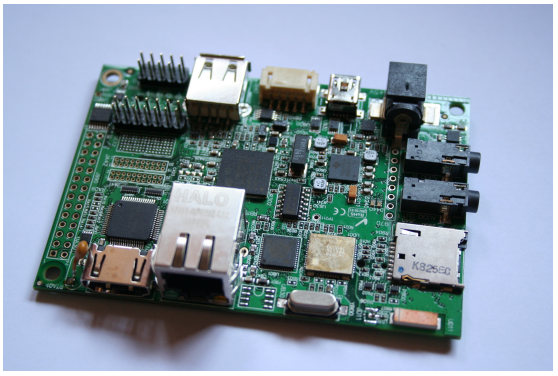
3.2.1 Pandaboard

Texas Instruments supported the development of a reference Single Board Computer (SBC) that had ~~specs~~ specifications similar to that of a modern smartphone (Figure 3.4a) and was capable of running desktop Linux, known as the Pandaboard [7]. A dual core 1GhZ ARM processor with 1GB of RAM, support for external storage, expansion ports, USB, Wi-Fi and Bluetooth in a board the size of two credit cards can be a powerful addition to a data heavy WSN, especially one that deals with images.

There is no mention of Pandaboards in the literature being used in WSNs but the low power of the system and advanced capabilities make it suitable for processing and transmitting data simultaneously.



(a) Pandaboard



(b) IGEP v2

(c) Waspmote

Figure 3.4: Sensor Node Hardware

3.2.2 IGEP v2

The IGEP v2 (Figure 3.4b) is another ARM based SBC that uses a 1GhZ single core processor with 512MB RAM and similar connectivity features to the Pandaboard, but around half the size. This does result in a reduced power draw and the device is still capable of running desktop Linux.

Due to the smaller size, and easier commercial availability, the IGEP has been used as a sensor node to record, process and send readings from multiple devices, such as air temperature, GPS and oxygen saturation as part of environmental monitoring [**?**]. In their research, they found that the IGEP achieved 9.1 hours of uptime using a 4000mAH battery, a capacity used in many modern smartphones.

3.2.3 Waspmote

The Waspmote (Figure 3.4c) is a general purpose sensing board that is designed to allow plug-and-play connectivity for multiple sensor modules. The node can be programmed with C++, using the Standard Development Kit (SDK) developed by the manufacturer and uploaded via USB. The processing power is not comparable to the more powerful SBCs, but the 600mAh battery is reported to last three months and the size is much smaller [**?**].

One of the primary benefits of the Waspmote is that they are commercially available with an actively maintained programming environment.

3.2.4 Discussion

The capabilities of each of these boards vary greatly from running a subset of C on an embedded OS to running full desktop Linux. Each visit we made to Danai Girang allowed us to test these devices and determine their eligibility for nodes that would

survive in the Malaysian rainforest and ~~yield usable range~~discover usable radio range data, which we discuss in the next section.

The ~~tradeoff~~ trade-off with these devices is that the Waspmote nodes, with limiting processing capabilities, is reported to yield a battery life of 6 months. Our tests yielded approximately 3 months, with minimal sleep scheduling, so this number does seem reasonable. However, the IGEP and Pandaboards are able to run any software that has been compiled for an ARM processor and do so within an unrestricted operating system. While this does allow them the flexibility to process data with the same power as a five year old desktop computer, our battery life tests drained the board within 6 hours when using 4 D cell batteries. We managed to increase this number by loading a power-efficient modification of the Arch Linux distro, removing any support for displays or running of a GUI, disabling unused ports and disabling the radios for as much of the uptime as possible. With these changes, we were able to achieve approximately three weeks of running with sleep periods.

Most SBCs, while having a reduced battery life, contain common I/O ports, such as: USB, VGA, Ethernet and Serial. This extensibility enables us to add additional wireless capabilities. For example, sensor nodes may require the use of a long range wireless medium for inter-node communication but the endpoint of a network uses Wi-Fi to allow users to connect to it and upload sensed data to the Internet; this can be done by adding a long-range wireless radio to an I/O port and providing software support.

3.3 Transmission Media

In this section, we explain the experiments we carried out to test the performance of different transmission media in the UK as well as the Malaysian rainforest. While range is the most important feature, a data rate that can handle hundreds of large readings in a day is a requirement, as wildlife cameras can generate ten images each time they are triggered and a high definition image can be up to 1MB in size. Our motivat-

ing scenario is focussed on the transmission of sets of 3 images (each around 800KB) for every trigger, where a sensor can trigger hundreds of times in a day.

3.3.1 Wi-Fi

Wi-Fi was already available on our initial test platforms and the high data rate made it suitable for sending a large volume of images in a short period. We knew that current cameras deployed in DGFC were up to 1km apart and we did not expect to cover that range completely, but we did expect to achieve that coverage by using intermediate nodes.

Research, outlined in Section 2.1.4, showed that the rainforest could reduce the range by up to 78% and the ideal maximum range of 2.4GhZ Wi-Fi is 100m [**?**].

We tested Wi-Fi range using two IGEP boards. We selected IGEP boards because they could be powered by 4 D Cell batteries and run a lightweight Linux operating system that allowed us to access information at the network layer, while still running higher level applications. The IGEP nodes we used did not have any additional hardware and the nodes were tested without the use of an external antenna. A Java application was written to periodically scan for available networks and store those results in a text file. One IGEP board was set as the base station and attached to a tree, at the same height it would be if it was attached to a camera, and another was walked to specified points around the base station at defined locations. These locations were chosen to include as many distances as possible and as many different forms of obstacle between the searching node and the base station, such as: line of sight (LOS), medium vegetation or thick trees.

This experiment was run in a wooded area in the UK and in the rainforest at DGFC. The specified maximum range of 802.11g is 120m. When considering attenuation and obstacles we were expecting the signal to be reduced by up to 50% in the UK. However, we found that we received a maximum range of 30m, with LOS. Figure ~~??~~ 3.5 shows

the results we experienced, while testing in the UK, some of the drops in signal can be attributed to dense foliage and readings that were not LOS, but a maximum range of 31m, with an SNR of 29.5 dBm, is less than we expected, as the UK does not experience high humidity often.

The graph shows a drop at 22m: this was due to unusually dense foliage that restricted the LOS between the base station and the receiving node. With five runs of this test we observed the same results. The primary aim of this experiment was to show the viability of Wi-Fi and to ensure our application functioned as intended, which it did. Further experiments could have been run to remove the anomaly but the results of the experiments in Danau Girang were the focus.

Despite the poor range from the tests in the UK, it was consistent with other studies reporting signal degradation of up to 78% in areas with dense foliage. We visited Danau Girang in 2011 to gather the requirements of the network and ensure the hardware is able to survive the humidity. Range experiments were run in the rainforest to see if a more humid environment impacts range any further, Figure ~~??~~ 3.5 shows this.

~~UK Woodland Malaysian Rainforest~~

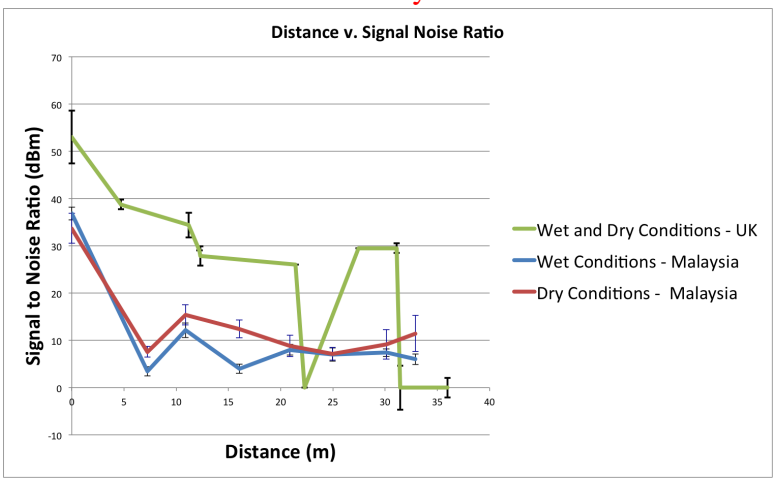


Figure 3.5: Signal-to-Noise Ratio for Wi-Fi

~~Comparing Figures ?? and ??~~ Figure 3.5 shows that the maximum distance to receive a signal is approximately the same in Malaysia as it is in the UK. There are more signal drops but this seems to be due to denser foliage blocking the line of sight. However, it

does suggest that the humid environment of the rainforest does not have a significant impact on the received signal. It is clear that the denser rainforest does impact the signal-to-noise ratio in a much shorter distance from the base station but a link is still made, allowing for a successful transmission of data.

Poor Wi-Fi performance led us to research alternative methods to increase the range without impacting the environment the network is to be deployed in. We considered using intermediate nodes, not attached to cameras, to account for the lack of range but, because some cameras can be up to 1km apart, we would need more than 30 nodes to create a connection between two locations.

We also researched wireless technologies that are more common in sensor networks. This does mean that the data rate is not as high as Wi-Fi and error correction in packet streams is not always as robust, but it is more suited to sensor networks, using less power and providing longer range.

Finally, we considered using the researchers or animals at Danau Girang, as 'data mules', creating temporary links between nodes while they are in the forest. However, the trip to Danau Girang yielded the information that researchers generally do not cover those distances in the forest and data delivery would be sporadic.

Although the range of Wi-Fi is poor, for our requirements, in both Malaysia and the UK, it did show that the results we experience in the UK are very similar to the results in Malaysia. ~~This means that tests run in the UK should be indicative of what we can expect in Danau Girang.~~

3.3.2 DigiMesh

Due to the poor range results of Wi-Fi, we created a second prototype of the network, using DigiMesh. DigiMesh is a proprietary wireless protocol, based on the 802.15.4 standard and designed for devices with limited power. Using the same frequency as Wi-Fi, DigiMesh has been reported to provide 7km of range, with a data rate of 250kbps.

In our prototype implementation, we are using Waspmote sensor boards [**?**], a general purpose node that is capable of transmitting through various communication mediums. Our Waspmotes are provided with DigiMesh modules and a 2GB SD card to store sensed data.

When testing the range of the Waspmotes, we followed a similar method to that which is outlined in Section 3.3.1. One board is in a fixed location and running a C++ application to poll for nodes in the network. Once a node is found it sends a message to the node every 10 seconds. The second board is set to scan the network and receive packets as soon as a base node is found; this node is then moved to different locations.

The receiving node prints out variables related to the received packet, such as: Received Signal Strength Indicator (RSSI), source MAC address and packet ID. However, not all packets are received so the RSSI can display 0 if there are errors in reading or if packet collision occurs. We found this to affect the results and have just used the two nodes to identify the maximum distance they can be apart, while maintaining a stable connection.

Initial experiments were run in a moderately vegetated area in the UK which yielded 497m of range. Limitations with buildings preventing us from testing any further but the signal strength still proved to be strong.

The initial results for the range tests were positive and DigiMesh does seem to be a viable solution to account for the lack of range when using Wi-Fi. As the frequency is the same as 802.11g, thus licensing it for worldwide use, we expected similar results in Danau Girang.

Experiments were run in 2 areas of the rainforest around Danau Girang and the results yielded were not the same as we experienced in the UK~~, and~~ ; the thick vegetation had a significant impact on the range, reducing it by almost 50%.

In more open areas of the rainforest, we achieved 199m on average, more dense regions of the forest reduced this to 102m on average. While these results are not as high as

we achieved in the UK, they are still suitable to use DigiMesh in the deployment of a WSN. ~~This~~ The low range could be because we were using low-gain antennas and little configuration had been made on the DigiMesh radio.

3.3.3 Adapting and Optimising for Harsh Environments

All of the hardware that we used for experiments had their components exposed, as shown in Figure 3.4 and would have become compromised if moisture came in contact with them. To protect them from this, we used waterproof cases with a protective foam inside, known as Pelican cases, to keep the nodes watertight, but still allowing airflow to displace the heat generated, shown in Figure 3.6. External antenna can be fed through the lip of the case, using thin cable, ensuring that the range of the transmissions is not affected by the case.

Figure 3.6: Pelican Waterproof Case

After experiencing poor range results, we considered running antennae up through the trees to the canopy so there would have been no foliage to block signals and the humidity would have been lower. However, this was not possible because of conservation issues and the fact that animals could easily damage the antenna wire.

Typically, WSNs in urban areas can use mains power and more remote networks utilise solar, or other renewable energy sources. In an area of dense foliage and high canopy cover, solar power is not a viable energy source. While DigiMesh has a lower transmission rate than Wi-Fi, its power requirements are lower, which maximises network ~~lfietime~~ lifetime for WSNs that do not have access to constant power or a renewable energy source.

3.4 Software

In this section, we explain the image processing software we developed for the real time processing of images within the network, as well as the retrospective processing of images that have been taken during previous deployments.

3.4.1 Triton

Triton is a C++ program that makes use of the Open Computer Vision (OpenCV) library for performing popular image processing functions [**?**]. Triton takes in a set of images, combines them and builds a Gaussian background model [**?**], a common background subtraction method commonly available in many computer vision packages. Using this model, the foreground is extracted from the image and this is scanned for objects. If an object is found, it is extracted to a new image and saved as a black and white template, to minimise space and assist with future classifications. An object identified from a set of images means that the set is then marked as interesting, i.e. it is believed to contain something. If no object is extracted, it is marked as empty

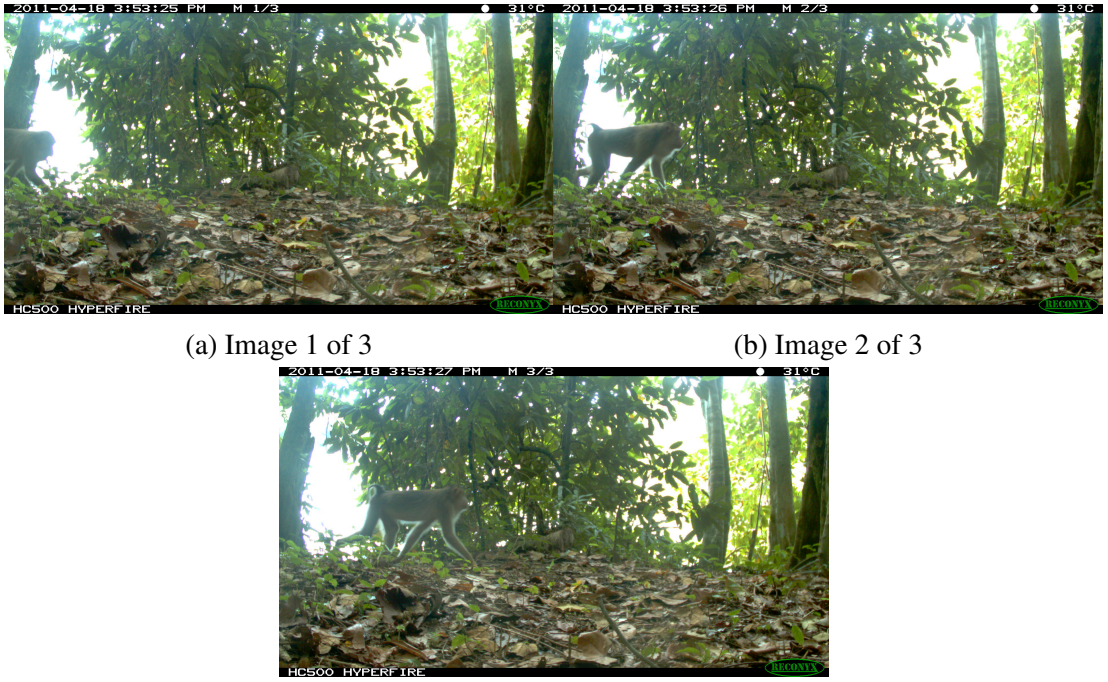
Primarily, we developed this tool to detect the large number of empty images captured around Danau Girang, due to trees falling, movement of the Sun, fast animals or dirt on the lens. However, Triton proved to be quite effective when finding images of interest and it was modified to be used with our sensors. In this section, we explain how Triton was tested and compared with human observations. To ensure a valid comparison, we also compared both results to a random classifier.

During a typical three month deployment along the Kinabatangan River more than 40,000 images can be taken. A large proportion of these images can be classed as *false triggers*; occurring when a camera is triggered by the motion detector but there is no object of interest. These can be caused by changes in the light, falling trees or animals moving too fast to remain in the shot in the delay between the motion detector triggering and the camera capturing an image.

We organised the images collected by the original camera that captured them and separating them into sets of three, because that is the number of images that the cameras were configured to take at each trigger. From this, we built a Gaussian background model [**?**] for each set and used the resulting model to detect objects in the foreground, classifying the detected foreground as the Region Of Interest (ROI), and extracting it.

Figure 3.7 shows a sequence of images taken by a Reconyx HC500 camera. The dynamic background and light levels make it very difficult to extract an object of interest. A background model is built from these images, which contains everything that is believed to be in the background. In this case, it should be the ground, the trees and the visible sky. The background is then subtracted from the image, leaving the foreground, and ROIs, larger than a user-defined threshold, are identified. The largest ROI is then extracted from the image and saved separately, shown in Figured 3.8.

From our visit to Danau Girang in 2011, we collected images from two different three month deployments, with the same cameras placed in different sites around Danau Girang for each deployment, giving us just over 70,000 images to test our approach. The images are sorted by the camera location and the date of collection. We process



(a) Image 1 of 3

(b) Image 2 of 3

(c) Image 3 of 3

Figure 3.7: Sequence of 3 Images Captured on a Reconyx Trigger

every 3 images as one set, building a background model of all three images.

We manually processed all of the images initially, marking images that are empty as false triggers. We then processed the images, using our application, and a resulting processed image is created from every set of 3 images. If nothing is detected in a set then no image is created and that set is logged as empty.

There are four classifications that can be made with images sets:

True positive: An ROI is extracted that contains the animal in the set.

False positive: An ROI is extracted that contains nothing of interest.

True negative: A camera is triggered with nothing of interest in the image and no ROI is extracted.

False negative: An image containing an animal has no ROI extracted.

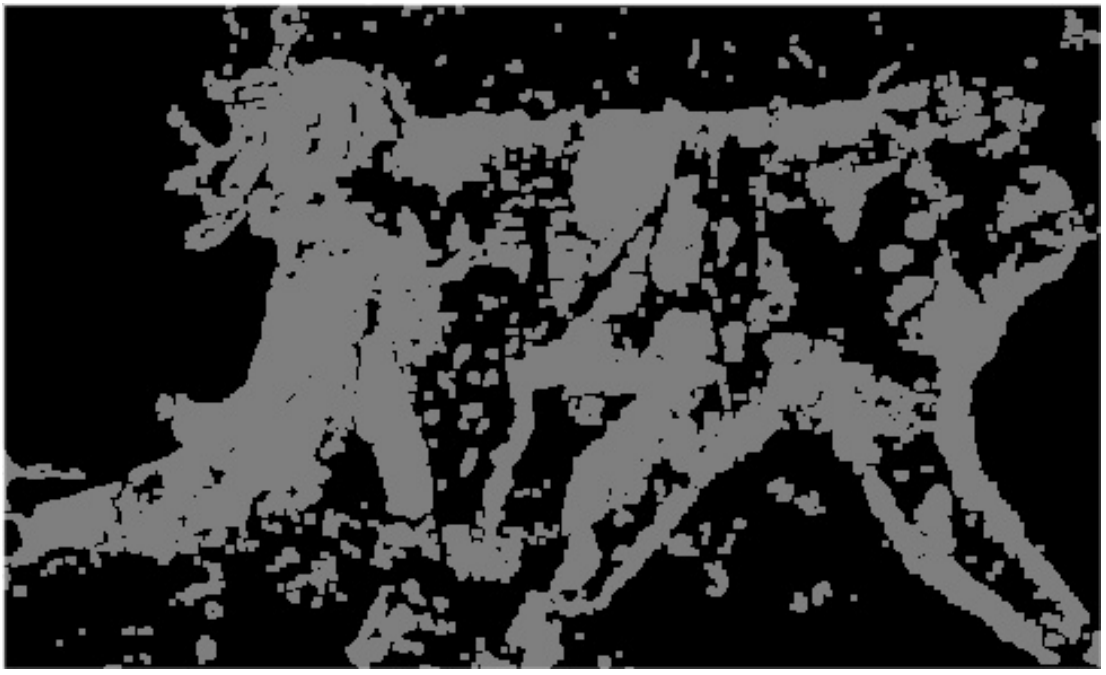


Figure 3.8: Resulting Processed Image from Original Set of 3

The processed images are then compared with our manual findings. The accuracy of our application is calculated by the following ~~equation~~equations:

$$Accuracy = (T_p + T_n)Ex_{tp}/TotalSetsAc_{tp}) * 100 \tag{3.1}$$

~~Where T_p~~

$$Accuracy = (Ex_{tn}/Ac_{tn}) * 100 \tag{3.2}$$

Where Ex_{tp} is the number of true positive sets extracted~~and T_n is the~~ , Ac_{tn} is the actual number of true positive sets, Ex_{tn} is the number of true negatives extracted and Ac_{tn} is the actual number of true negative sets.

Table 3.1 shows experiments run on images from a randomly selected camera deployed in Danau Girang, testing on 300 sets of 3 images. These images were manually processed and classified as interesting or empty, of the 300, 94 sets were identified as interesting. Using Triton, 77 of those interesting sets (true positives) were extracted,

True Positive	True Negative	False Positive	False Negative	Total Image Sets
77	201	5	17	300

Table 3.1: Classification Results

with another 17 false positives extracted. 201 true negatives were correctly identified and a further 5 were identified as false negatives. As Triton processed these sets, we tested the time it took to complete a set of images ~~, determine whether it was interesting or empty~~ on a Pandaboard and we found the mean time to be 43 seconds.

Out of the 94 interesting sets, 77 were identified correctly, giving an accuracy of 82% for finding sets of interest. Furthermore, of 206 empty sets, 201 were found, which gives a 98% accuracy at finding empty images.

These preliminary results show that our method is effective at detecting ~~false positives but is less effective at detecting false negatives. It appears that these misclassifications primarily come~~ images of interest (True Positives) and it appears that misclassifications (False Positives) primarily came from black and white images taken at night, images where minimal movement of the animal has caused a trigger and images where an animal has caused a trigger but it has been too fast moving to be in the second two images.

After a longer deployment, we would be able to build up a more substantial background model to account for some of the animals being less dynamic in images and we expect this to decrease our error rate thus reducing the number of false negatives.

The results of Triton were compared to a random classifier, implemented in Python, that ran through a directory of images and generated a random number for each file. If the generated number was less than 0.25, then it was marked as true positive, between 0.25 and 0.49 marked it as false positive, between 0.5 and 0.74 was a true negative and between 0.75 and 1 was a false negative. Table 3.2 shows the outcome, with 72 *true positives* extracted, resulting in an accuracy of 80%. 73.3 *true negatives* from a total of 206 were found, giving an accuracy of 36%. Although this set does not compare to the

True Positive	True Negative	False Positive	False Negative	Total Image Sets
72	73.3	78.3	75.3	300

Table 3.2: Random Classifier Results

number of images collected in a six month deployment, human classification is a time consuming process and to have 900 classifications is complex. Although these could be crowdsourced, the reliability becomes questionable, in part due to the specialist nature of the images. This does show that our approach is quite close to the accuracy of a human when finding images of interest, but is able to do so in a fraction of the time, processing hundreds of images every minute.

Triton, coupled with a set of human eyes at the heart of the network, should be an effective approach for prioritising images through the network, even when a classification cannot be made through the use of local knowledge. More importantly, in an area where the environment is as dynamic as the Malaysian rainforest, 98% accuracy for detecting empty images can be crucial, if only for saving network bandwidth.

Future Work

Triton is currently capable of processing a set of images and extracting the largest object of interest, if one is detected. The primary benefit of this is that it requires no training initially and works, with fairly good accuracy, from the time of deployment. However, functionality can be extended by storing the extracted images and associating it with the actual content, i.e. animal name, future extracted images can then be matched to the templates, within a threshold, to assist with classifications within the network. Currently, the templates are extracted, stored and associated with their contents once classified by a human, but Triton does not use these. If Triton detected an interesting image, it could then search a folder of templates and use OpenCV to compare the images and provide a cursory classification based on the closest match.

Extracted images are stored in black and white and are only a few kilobytes in size, this

will be useful as we expect that a large number of templates would be required in order to accurately identify an object of interest. For example, in our motivating scenario, an animal could be any distance from the camera, its legs could be in different positions or its angle it faces towards the camera could be different, giving many possible images. This process could be optimised by only matching regions between extracted images, such as the head or body, but this would require further experimentation.

3.5 Conclusion

In this chapter, we have shown the current hardware choices available for more computationally capable sensors and detailed our experimental results on how rainforest environments impact wireless transmissions. Although wireless technologies such as Wi-Fi provide a high data rate, their range is limited and not suitable for sparsely located nodes covering a large area; especially in a humid environment. ~~Commerical~~ Commercial Wi-Fi solutions are becoming more ~~widepsread~~ widespread in open, public places, such as: shopping malls, airports and ~~universitites~~universities, but these places allow for multiple routers, constant power supply and a greater range as routers are placed as high as possible to maximise coverage. Newer wireless technologies, designed for long-range communication in sensor networks, are becoming increasingly more viable and, while they do have lower data rates and less robust protocols, they are more suitable for a resource constrained WSN that requires minimal power draw when transmitting. Using general-purpose hardware also means that there is no protection against water, humidity and animal or human intervention. We adapted existing waterproof cases to suit our needs and preliminary short term deployments showed that they are able to prevent moisture from entering the case. We have also developed an image processing application, Triton, that attempts to extract an object from a set of images that it believes to be the subject. Image sets that do not have any subject are classed as empty and should have no object extracted while those with a subject are interesting and should have an object extracted. Testing Triton on existing images at

DGFC yielded 98% accuracy for detecting empty image sets and an 82% accuracy for finding interesting sets. A key feature of Triton is that it is not calibrated for a particular background or environment, so these results are especially promising.

Chapter 4

Wireless Sensor Network Architecture Design

In this chapter, we explain our proposed network architecture that uses local and global knowledge to make informed routing decisions and to classify sensed data within the network. Our approach, K-HAS, uses a three-tiered, hierarchical approach with each subsequent tier providing increased knowledge processing capabilities.

We aim to show that sensors capable of processing knowledge will provide a more efficient network and be able to prioritise the delivery of interesting sensed data. We also believe that human input is a valuable way for such a network to learn about classifications it has made correctly, and incorrectly, and use that knowledge to inform future classifications. To prove this, we have developed an architecture that uses different levels of knowledge processing throughout the network, the Knowledge-based Hierarchical Architecture for Sensing (K-HAS).

The rest of this chapter is structured as follows. Section 4.2 outlines the main aims of K-HAS and what it is capable of that typical sensor networks are not. Section 4.3 details the software and data standard choices we made for our implementation of K-HAS. Section 4.4 provides a sample ~~walkthrough~~ walk-through of K-HAS, relating to our motivating scenario and Section 4.5 concludes the chapter.

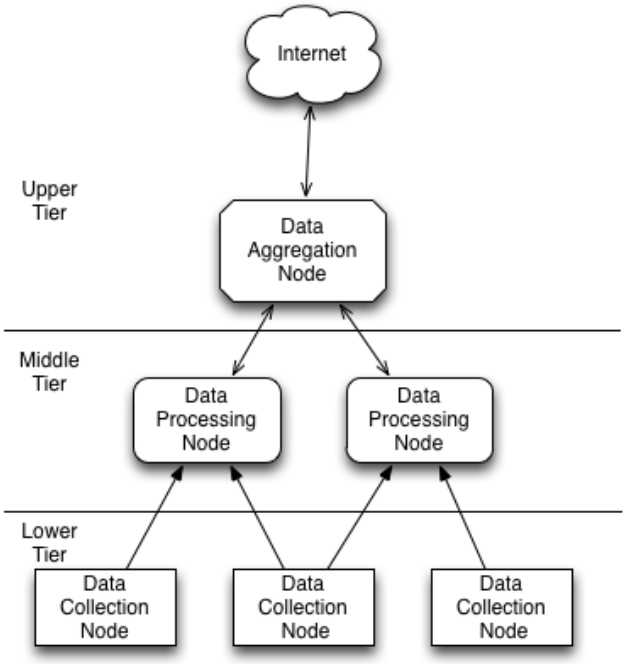


Figure 4.1: Overview of the K-HAS Architecture

4.1 The Local Knowledge Problem

While large in size, it is typical that a WSN would use low cost nodes with limited power, memory and computational capabilities; causing them to lack the ability to be aware of their surroundings or the data that they are sensing [15]. This means that, unless fixed by a routing protocol or a technician, data is delivered on a chronological basis and is then filtered at the base station, usually manually. Some WSNs store all of the data on the node and users of the network use a 'pull' model to query for data from nodes [**?**], but this requires some technical knowledge and, although it does increase the battery life of the nodes, it is a manual process again.

The environment that a WSN is deployed in is often a rich source of data to be sensed (such as inside a bird's nest to sense temperature, humidity and movement), which often contains patterns that can be used to improve the performance of the network. For example, if a node knows that it is has only been triggering between the hours of 6pm

and 5am for the past few weeks, it can then learn to enter a deep sleep outside of those hours or use that time to transmit data it has been storing while it assumes it will be inactive, based on previous days. Alternatively, this knowledge can be used to prioritise data throughout the network so that the most important data is received first, instead of the most recent. An example of this could be two camera nodes deployed facing the entry and exit of a building, tasked with looking for intruders between 5pm and 8am. If the camera facing the exit is triggered at 5:01pm and the camera on the entrance is triggered at 5:05pm, then the knowledge that the security guard leaves through the exit between 5:00pm and 5:08 pm will allow the entrance camera to prioritise its capture as more important, as it is an irregular occurrence.

This knowledge can be categorised as either *local* or *global*. Local knowledge is the knowledge of an area that has been gained through experience or experimentation [**?**] For example, a native to the Amazon may know that three of the locations in which the nodes are to be deployed are flooded for two weeks of the year, rendering their readings useless for that time period and increasing their risk of failure. This is local knolwedge, as it cannot be gained without experiencing the flooding in that area, or measuring local water levels.

While we believe that we are the first to use the concept of local and global knowledge within the wireless sensor network domain, the terms have been around for many years. In 1999, a book that referred to local knowledge as *indigenous knowledge* defined local knowledge as 'systematic information that remains in the informal sector, usually unwritten and preserved in oral traditions rather than text' [**?**].

Over the past twenty years, local knowledge has been used in various contexts, from researching lending and the credit market [**?**] to extracting local knowledge from natives to improve farming techniques [**?**]. This research, as well as work that will be covered later, showed us that there are two kinds of knowledge: global and local.

It was from agriculture research that we were able to refine our definition of local knowledge, [**?**] defines local knowledge as 'knowledge that farmers have derived locally

through experience and experimentation'. They also say that indigenous knowledge is different in that it is culturally specific. From this definition, as well as our work with our motivating scenario, we were able to generalise the definition and expand upon it.

We now define local knowledge as *knowledge of an area, held by a domain expert, that has been gained through experience or experimentation*. The weather of a region is global knowledge because it can be found through a variety of media, whereas the saturation levels of the soil in a field would be local knowledge as it would require experimentation to gain that knowledge.

Using this definition, we believe that **utilising local and global knowledge within a sensor network can inform routing decisions to make better use of the bandwidth in resource-constrained WSNs by sending data that the node believes to be important first, rather than chronologically**. Patterns in the data, and knowledge of the environment surrounding a node, can allow a node to infer what data may be valuable, automate the processing of adding context to the sensed data, learn from previously sensed data and utilise global knowledge of ongoing projects within the network to determine what data is thought to be of a higher priority.

To show this, we have developed a network architecture for WSNs that utilises knowledge from the data it senses, as well as its deployed environment. It is called the Knowledge-based Hierarchical Architecture for Sensing (K-HAS) and this thesis will show how K-HAS addresses the problem of delivering the most important data first and improving the overall efficiency of the network.

4.2 K-HAS

K-HAS has been designed as an architecture for WSNs that ~~uses~~ addresses the above problem by using knowledge to classify sensed data and adapt to changes in the structure of the network. By pushing knowledge bases out to the edge of the network, all nodes in the network have some awareness of the data they are sensing, as well as how

important it is, based on the current projects that the network is involved in. This is achieved by using rules with different levels of granularity based on the knowledge processing capabilities of that tier. Figure 4.1 shows a high level overview of K-HAS, showing the flow of data from each tier.

4.2.1 Data Collection

The data collection (DC) tier is very similar to sensor nodes commonly used in a WSN, using hardware that has similar, limited processing power and storage to nodes such as: the I-Mote and Waspmote. These DC nodes are deployed at the edge of the network and tasked with sensing their environment, pre-processing the sensed data and using each other to relay data to the next tier.

DC nodes are capable of performing processing on data, such as the time it was recorded and its size, but their limited knowledge processing capabilities allow them to have an increased battery life and reduced size, making them suitable for a variety of deployments as they can be integrated easily with existing devices (i.e. cameras) and be easily disguised, or hidden.

Knowledge Base

Reduced knowledge processing capabilities and low memory restrict the knowledge that these nodes can hold and they are limited to a static knowledge base that is encoded at the time of deployment. DC nodes run an operating system designed for embedded devices, the size of the operating system is minimal when compared to an operating system used on modern desktop computers and provides much more limited functionality, such as sleep scheduling, basic ~~filesystem~~ file system access and the capabilities to run a variation of popular languages.

For example, TinyOS [**?**] is an operating system designed for sensor nodes and uses a dialect of the C language, called nesC, that is optimised for the memory constraints

of most sensor nodes. The operating system exposes components that allow basic commands, such as reading from the sensor interface or sending a message. Commands are executed as a request to perform some action and events signal the completion of that action. Intensive tasks, such as data processing, can be scheduled to run at a later time, ensuring that the low-power node remains responsive and preventing any calls from blocking other events.

The Waspmote nodes, explained in Section 3.2.3, do not run any operating system and any event handling, networking or memory management is performed solely by the single file application that is uploaded to the node through the ~~bootloader~~boot-loader. Performing intensive tasks, such as local data processing, can drain the battery of the node, cause it to fill the available memory or even cause the node to go down. DC nodes do not have the safeguards in place that most operating systems and applications do, operational code must be lightweight, responsive and ensure that events do not run for longer than intended.

Some nodes within this class do not have the capability to process text files within the file system and, therefore, must have the knowledge added to their operational code. DC nodes only perform simple operations on the properties and content of the data that they sense, such as the time it was recorded, the location and its size. For more complex data, such as images and video, DC nodes do not possess the computational power required to process them and instead use the metadata associated. Unlike modern rule engines, these static rules do not use forward chaining and the outcome of one rule does not cause the rules to be fired again. Listing 4.1 shows an example of some of the rules in the knowledge base and highlights that the core structure of this file is simply a list of if statements that can be executed one after the other. Forward chaining is not used here because processing times need to be kept to a minimum and a chain of rules being fired could easily overwhelm the node. If any rules are fired that suggest the observation may be interesting, it is prioritised through the network and potential classifications are encoded within the observation before it is sent on.

```
1 if month of observation is JUNE AND time of observation is between 17:00 and
      19:00 and active otter project is TRUE
2   add classification to observation('Potential Otter sighting')
3   prioritise observation as interesting
4 if temperature is 37 AND time of observation is between 01:00 and 05:00 and
      active leopard project is TRUE
5   add classification to observation('Potential Leopard sighting')
6   prioritise observation as interesting
```

Listing 4.1: Pseudocode DC Node Rules

When the data is recorded by the DC node, the knowledge base is fired and inferences are made about the contents of the data. Each DC node has a static knowledge base loaded onto it before it is deployed, which is based on the local knowledge of the area that it is to be deployed in. For example, a node deployed on the bank of a river would have a different knowledge base to a node deployed in the fields of a plantation.

Once a trigger has been processed, the data is packaged and then sent on to the Data Processing (DP) node.

4.2.2 Data Processing

DP nodes act as cluster heads of the network, serving a subset of all deployed DC nodes. When data is sensed, it is forwarded through all DC nodes to the DP node that is tasked with serving the originating DC node. These nodes have more knowledge-processing capabilities than a DC node and do not perform any direct sensing.

Due to the greater capabilities, DP nodes have a much shorter battery life than DC nodes and a network typically consists of fewer DP nodes. These capabilities, such as increased memory and higher processing power, allow DP nodes to run a rule management system (such as Drools [**?**]), that would not be possible on nodes that run on an OS for ~~embdedded~~ embedded systems, that is able to handle more complex rules and process more than just files, they can perform the same tasks as most modern computers, such as: image processing, audio processing or reading metadata from files that requires extra libraries. When a DP node receives data, it processes everything associated, this includes metadata, the data itself and the inferences made by the DC node. If

the DC node has inferred that the data is of a higher priority, then this data is processed first. This is done by prioritising the data at two stages: once it has been received and when it is about to be sent.

In our current ~~implementation~~design, DP nodes use two different radios, a Zigbee radio to allow long range communication from DC nodes and a Wi-Fi radio that provides short range communication that allows for higher data rates.

Knowledge Base

In our motivating scenario the network is image-based, this means that the DP node would perform image processing, as well as processing the image metadata. The increased knowledge processing capabilities allow DP nodes to run rules dynamically, learning from the sensed data and providing classifications that change based on changes in the environment. For example, if a DP node has not seen an elephant before, and it is not aware of the object in the image, then it will await a human classification. The node will then record the time period that it receives elephant pictures, i.e. June to July, and become more alert the following year. Similarly, the node will know not to look for pictures of nocturnal animals during the day. This local knowledge allows processing power to be saved and, thus, time; this ensures that the processing of sensed data is optimised as much as possible in order to reduce the time it spends in the network.

The rule engine used should allow for rules to be dynamically inserted into the rule base, so that rules can be updated through network communication.

Upon receiving sensed data from a DC node, the rule base is fired on the ~~metadata~~ meta data of each file received. If the rules determine that the data is of interest or, in the best case scenario, provides a classification, then the data is packaged and sent on to the Data Aggregation (DA) node.

4.2.3 Data Aggregation

Placed at the root of the network, these are nodes with the same knowledge-processing capabilities as DP nodes (although they typically have greater memory, processing power and a continuous power source) and would be accessible by users of the network. When DA nodes receive sensed data, it is unpacked and stored with a link to the node that the data originated from. Compared with a standard WSN, these nodes can be compared with a base station, or endpoint.

Any information added by the DP node is parsed and classifications are extracted. If a classification is found, it is stored and the DA node checks for any active projects that contain the classification. If a match is found then all users involved in the project are informed via their preferred method of communication. Using the motivating scenario as an example, the people involved with projects could be researchers and professors and they may be looking for images of leopards, requesting to be informed via Twitter.

All sensed data received, regardless of whether it has been classified, is accessible through a web interface hosted by each DA node. The interface shows all of the sensed data from each deployment, along with the associated classification. More importantly, it allows users to classify the data using a voting system. Users have roles which give them different privileges within the system. Normal users are able to vote and the majority vote is seen to be the current classification. While this is not a necessary feature of a DA node, it does allow the network to utilise the knowledge of human experts to inform future classifications.

However, privileged users are able to confirm a classification and prevent any further votes. Once a classification has been confirmed, it is then sent back to the DP node it originated from. If the classification made by a user is different to the one inferred by the node, then it updates its knowledge base and acknowledges receipt.

This ~~system is~~ section of K-HAS is yet to be implemented but we believe that this system will be vital in the early stages of deployment as ~~this is~~ it will be used to build

up a knowledge base. The more user classifications there are, the more accurate the network will be in the future. After a few months of classifications, K-HAS ~~is then~~ would then be able to use the knowledge base to make more informed classifications, requiring fewer classifications/confirmations from users. The exception to this is sensed data that cannot be matched to the knowledge base.

Data aggregation nodes should not be hampered by limited battery life that deployed nodes would experience as we expect them to be placed in a base station with power availability and access to the Internet. Therefore, DA nodes would typically be desktop computers with a constant power supply.

Knowledge Base

DA nodes do not typically experience the resource constraints that DC and DP nodes must compensate for. Because of this, they hold a global knowledge that contains a history of all observations made by all nodes, as well as the location and deployment times of all nodes in the network. When nodes are deployed, this is updated, by users, to show the location of the node and when it was placed.

Every classification sent from DP nodes is stored in a database and contains all information about an observation: date, time, originating node, route taken within the network, location, sensed data, classification of sensed data. They also hold information for all projects running, for example: a project to track elephant movements within the forest. When sensed data that could be related to the project is received, such as an image with an elephant classification, users associated with that project are informed through email, or other means.

While DC do not store any of the observations they capture, and DP nodes only store part of the observation that can be used in future classifications, DA nodes store the complete observation made by every DC node, as well as any extra knowledge that is added by users upon receiving the sensed data. What is stored by the the DP node is

dependent on the deployed purpose of the network. For example, in our motivating scenario, we store: the processed image, the classification, the node it originated from and the date and time it was captured.

As well as this, the functionality of DA nodes can be extended to provide administrative operations on the network, such as the recording of node locations, time of deployment and viewing all active nodes. This allows the DA node to monitor active nodes and alert users if a node has not sent any data in a while. The longer a K-HAS network runs, the more knowledge a DA node gains, both from users that classify observations and from changes in the sensed data. This knowledge is then relayed back to DP nodes, updating their knowledge bases as to what ~~clasifications~~ classifications were correct and which need updating for future observations.

Feedback Loop

The feedback loop is a protocol within K-HAS, that uses human input, and other sources, to update DP nodes. When a DP node classifies sensed data, it stores some of that to assist with future classifications. What is stored depends on the type of sensed data. For example, images would mean that the DP node would store the resulting processed image, its classification and information about the time it was taken, the camera that took it and the location. When that is sent to the DA node, a human would then look at the image and mark the classification as correct, or modify it if it was not. Once that classification has been finalised, the DA node sends either a confirmation or a modification to the DP node that sent it, updating its knowledge base. This protocol seeks to reduce the number of incorrect classifications the longer the network is deployed and allows the nodes to be dynamic and 'learn' throughout the lifetime of the network. It also allows nodes to adapt quickly to new data, if a DP node is unable to classify an image of an animal it does not yet have template images for, the knowledge of a human expert can provide those templates and the feedback loop will deliver that knowledge.

In the current design of K-HAS, the feedback loop has only been tested in a development environment and has not been fully implemented, we believe that the actual implementation of the feedback loop will allow the network to adapt dynamically to changes during its deployment.

4.2.4 Sequence Diagram

In figure 4.2, we have outlined the interactions that take place between nodes when an observation has been captured. Some parts of the architecture have not been fully implemented and those are highlighted throughout this chapter. This diagram provides a visual representation of how the tiers of K-HAS interact as well as how K-HAS prioritises sensed data and how K-HAS could utilise knowledge of professionals to improve the accuracy of its classifications.

4.3 Technological Components

In this section, we describe the technologies used in our designs of K-HAS and how they integrate in order to use local knowledge based on their respective knowledge processing capabilities. The majority of components, both hardware and software, used in K-HAS are used so they are applicable for any WSN, but some choices have been made to remain in line with our motivating scenario and, thus, are more specifically suited for the capture of scientific observations.

4.3.1 Data Standard

To pass sensed data through the network, we first had to choose a standard format that would allow us to encode the sensed data, as well as enrich it with inferences made through processing. Darwin Core (DwC) is a body of standards with predefined terms

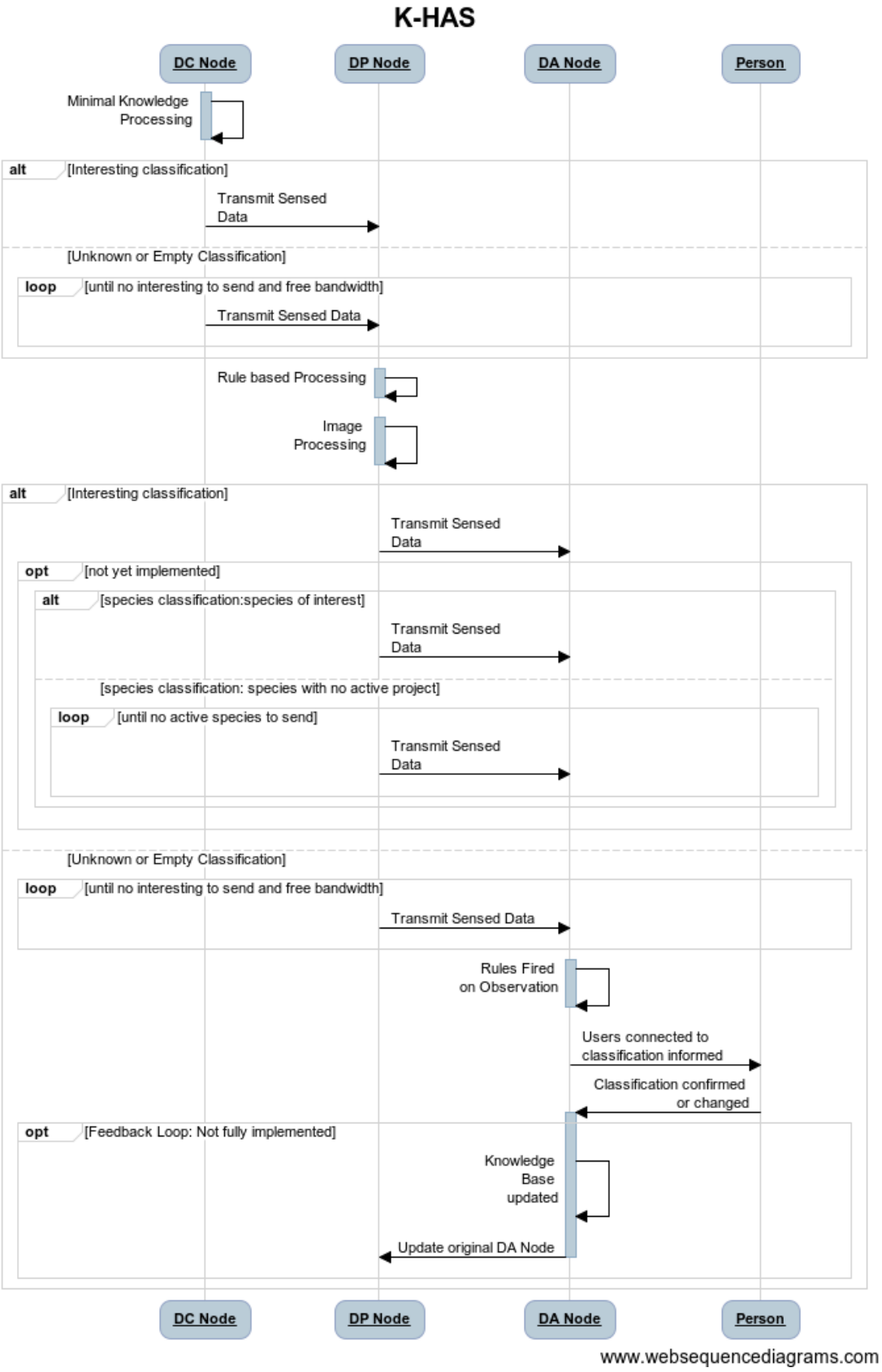


Figure 4.2: Sequence Diagram for K-HAS upon Capture of an Observation

that allows for the sharing of biodiversity occurrence information through the means of XML and CSV data files [**?**].

The Global Biodiversity Information Facility (GBIF)[5] indexes more than 500 million Darwin Core records published by organisations all over the web, allowing datasets that were previously siloed from the public to be accessed by both human and machine. The primary purpose of Darwin Core is to create a common language for sharing biodiversity data that is complementary to and reuses metadata standards from other domains wherever possible [**?**].

DwC Archives follow a star file structure, where a record can contain many occurrences, which is the recording of a species in nature or in a dataset. In an occurrence, there is an *event*, a recording of a species in space and time, enriched with other terms such as *identification* and *location*. DwC is ~~that~~ the standard set of terms that can be used, while a Darwin Core Archive (DwC-A) provides the structure for data recorded using these terms. The core files in a DwC-A are:

1. EML.xml
2. Meta.xml
3. Data files

While DwC does not have the extensions available to OBOE, an extensible base ontology designed for ecological observations that is explained in detail in section 5.1.1, it is extremely concise for recording observations within the biological diversity domain and aims to be a standard reference for sharing these observations.

The record shown in figure 4.3 represents a DwC-A that conforms to the star schema. The ecological metadata language (EML) document contains all of the details about the project, such as who is involved, the institution code, contact details and the project(s) related to the observation. Listing 4.2 shows a fragment of the EML file and a complete DwC archive can be found in Appendix **??**.

```
<alternateIdentifier>e71fda1c-dcb9-4eae-81a9-183114978e44</alternateIdentifier
    >
<title>Images from Danau Girang during the PTY Project 2011-12</title>
<creator>
  <individualName>
    <givenName>Christopher</givenName>
    <surName>Gwilliams</surName>
  </individualName>
  <organizationName>Cardiff University</organizationName>
  <positionName>PhD</positionName>
  <address>
    <city>Cardiff</city>
    <administrativeArea>Cardiff</administrativeArea>
    <postalCode>CF24 3AA</postalCode>
    <country>Wales</country>
  </address>
  <phone>(+44)2920 123456</phone>
  <electronicMailAddress>C.Gwilliams@cs.cf.ac.uk</electronicMailAddress>
<onlineUrl>christopher-gwilliams.com</onlineUrl>
</creator>
<pubDate>2012-07-26</pubDate>
```

Listing 4.2: Darwin Core Ecological Metadata File Fragment

The descriptor file is an ~~xml~~ XML document that contains the column headers in the attached files and the mappings of those headers to DwC terms, an example file is shown in Listing 4.3. This file shows that this is an archive containing the sighting of an individual, and that the sighting has been split into two files. The largest benefit of Darwin Core is its modularity. Extension files can be added to enrich the data for each occurrence. In this example the extension file is named as *images.csv* and contains image-based evidence to support the observation.

```
<?xml version="1.0" encoding='utf-8'?>
<archive xmlns="http://rs.tdwg.org/dwc/text/" metadata="eml.xml">
<core encoding="UTF-8" linesTerminatedBy="\n" fieldsTerminatedBy=","
    fieldsEnclosedBy='' ignoreHeaderLines="1" rowType="http://rs.tdwg.org/dwc/
    terms/Occurrence">
<files>
<location>set.csv</location>
</files>
<id index="0"/>
<field index="0" term="http://rs.tdwg.org/dwc/terms/occurrenceID"/>
<field index="1" term="http://rs.tdwg.org/dwc/terms/basisOfRecord"/>
<field index="2" term="http://rs.tdwg.org/dwc/terms/recordedBy"/>
<field index="3" term="http://rs.tdwg.org/dwc/terms/associatedMedia"/>
<field index="4" term="http://rs.tdwg.org/dwc/terms/eventDate"/>
<field index="5" term="http://rs.tdwg.org/dwc/terms/eventTime"/>
<field index="6" term="http://rs.tdwg.org/dwc/terms/locationID"/>
<field index="7" term="http://rs.tdwg.org/dwc/terms/scientificName"/>
<field index="8" term="http://rs.tdwg.org/dwc/terms/identifiedBy"/>
<field index="9" term="http://rs.tdwg.org/dwc/terms/dateIdentified"/>
</core>
<extension encoding="UTF-8" linesTerminatedBy="\n" fieldsTerminatedBy=","
    fieldsEnclosedBy='' ignoreHeaderLines="1" rowType="http://rs.gbif.org/
    terms/1.0/Image">
<files>
<location>images.csv</location>
</files>
<coreid index="0"/>
<field index="1" term="http://purl.org/dc/terms/identifier"/>
</extension>
</archive>
```

Listing 4.3: Darwin Core Descriptor File

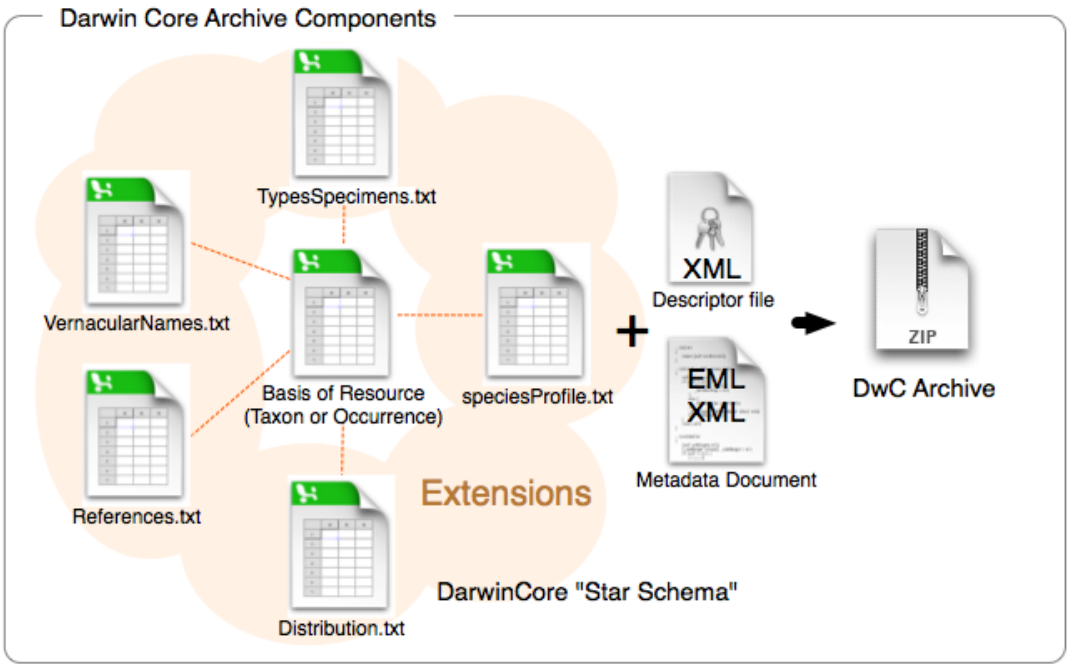


Figure 4.3: A Darwin Core Star Archive [?]

Listing 4.4 shows the comma-separated central file (Basis of Resource) containing the core details of the observation, i.e. the animal observed, and the column headings map to the descriptor file. Other linked files are typically linked by the unique ID of the observation, containing information that extends the observation and provides further context.

```
1 eventID,basisOfRecord,recordedBy,eventDate,eventTime,locationID,scientificName
      ,identifiedBy,dateIdentified
2 3,MovingImage,1,2010-12-22,16:14:35,1,,,
```

Listing 4.4: Darwin Core Occurrence Data

All of these files are then archived and sent as a ZIP folder throughout the network. If the sensed data is media based, then the media is included as well. Sofwtare libraries to process DwC archives are included on both DP and DA nodes.

Darwin Core is suited to K-HAS because its use in ~~scientific~~ ecological observations matches our motivating scenario and the archive can be easily created by a DC node, as it does not require any heavy processing and all of the files are commonly used formats.

4.3.2 Middleware

The knowledge-processing capabilities of DA and DP nodes are the same and this is part of what makes K-HAS different from most other WSNs; both types of node run the sensor middleware, but each for different purposes. DA nodes use the middleware for administrating the network, receiving and archiving sensed data and allowing users to provide classifications. DP nodes use it for the receiving, sending and controlling the flow of processing of sensed data before it is passed on.

Existing suitable middlewares have been detailed in Section 2.3.2 and our requirements for K-HAS were partially determined by the expertise of the users in our motivating scenario. Below is a list of our three core requirements:

Portability Heterogeneous WSNs utilise nodes with different architectures and capabilities, if middleware is to be used on the nodes it must be able to run on these varied devices.

Usability Users of K-HAS should not be expected to have knowledge of computer science or the underlying architecture, this network should be usable by almost anyone. The same must be said for the middleware as well.

Extensibility A closed-source middleware can be used, but it must then support all sensor nodes and data types, as well as receive regular updates. Open-source, or extensible, middleware can be used to add support for newer nodes.

GSN is a Java-based open-source middleware. New generic sensors can be added through XML files, while more complex sensors can be added through custom Java classes. GSN is covered in more detail in Section 2.3.2. Because GSN can run on any architecture that supports the Java Virtual Machine (JVM), it meets our portability requirements and the web interface to provide administrative functionality makes it usable by those without any domain knowledge. Finally, the ability to add new sensors

through XML means that it can be extended by almost any user of the network with very little guidance.

4.3.3 Knowledge Capture

GSN is packaged with a web interface that allows users to see all nodes deployed and view the latest sensed data received. The web interface is targeted towards users with domain expertise and has limited functionality focussed towards sensor administration. However, it does make GSN accessible by more than one computer, as well as a variety of different architectures. For example, the admin webpage could be accessed on the machine that runs GSN, or from a tablet computer connected to the same network. We used the same approach to develop a web-based tool that provides access to all sensed data, as well as a simple interface for performing tasks, such as uploading new rules or updating the location of nodes.

All sensed data is read from a MySQL database and users can view the metadata from each observation, such as location, date, time and temperature, as well as the data itself. From this, users are able to classify the data based on their role. Shown in the ontology in Chapter 5, K-HAS uses roles to control active projects and classifications; there are administrators and researchers. Researchers are involved in projects and receive notifications when relevant data has been received. They have access to the web interface and can vote on classifications for sensed data. Administrators lead projects and can create/complete them, but they also have the ability to finalise classifications. K-HAS follows a knowledge hierarchy (Figure 4.4), with administrators at the top and DC nodes at the bottom. While there are more DC nodes in the network, their knowledge bases are more limited and classifications are trusted less than classifications made by DP nodes. Although there are fewer DP nodes, their knowledge bases are more detailed. Researchers and admins do not share a level on the knowledge hierarchy because we assume that administrators would be more experienced domain experts. For example, researchers could be students at Danau Girang, whereas an admin would be

a professor with more experience. When an administrator makes a classification, all prior classifications are ignored and the feedback loop protocol is used to update nodes.

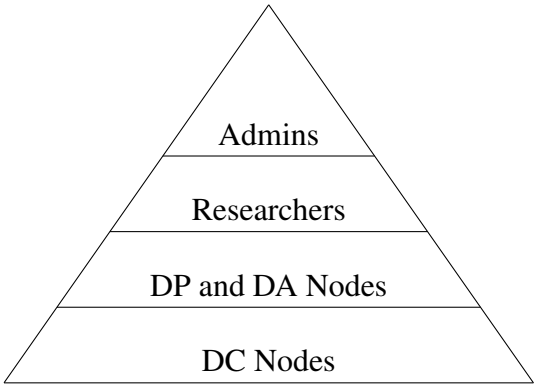


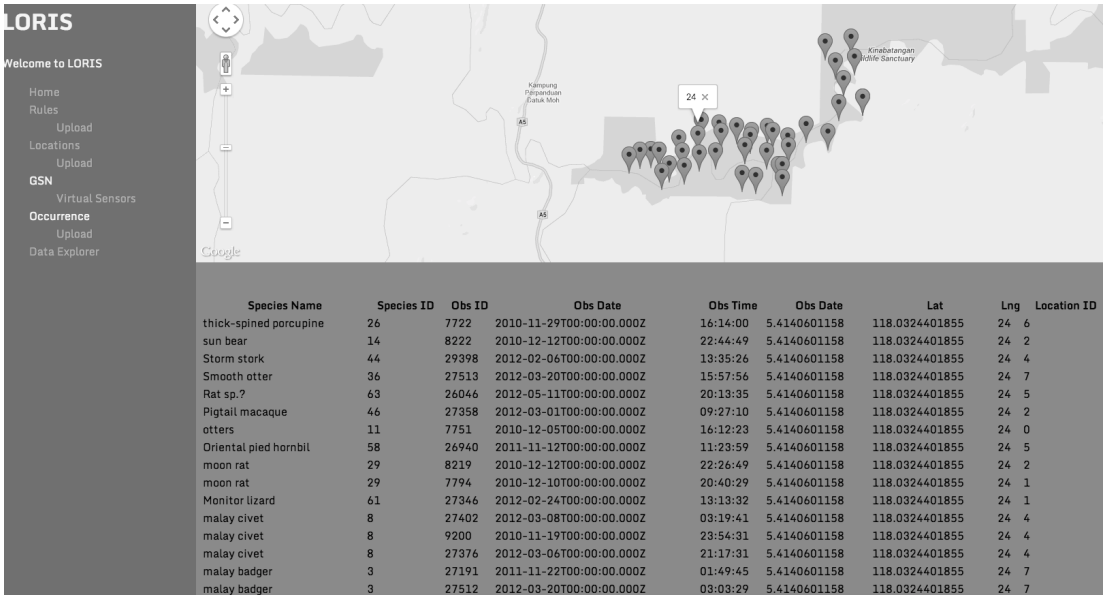
Figure 4.4: Knowledge Hierarchy for K-HAS

Figure 4.5 shows an observation where users can vote on the contents. An administrator can then confirm that classification and prevent further votes from being cast. This type of moderation means that it does not have to be specialists voting on sensed data that cannot be classified by DP nodes.

Figure 4.5: Web Interface for Observations

Viewing data for new observations is useful for gaining classifications and alerting

members of a project, but viewing older sensed data allows patterns to be identified in order to create new rules. Figure 4.6 shows a map of all deployed nodes in the area surrounding the field centre in our motivating scenario. When users select a node, a table is populated with all of the classified observations that it has captured; this can then be used to extract patterns from the data and create rules. For example, the ~~two~~ three observations of the Malay civet are only seen late at night, if further observations also showed this, then we could create a rule defining the active hours of the Malay civet and, potentially, list days that it is likely to pass. These rules can then be written and uploaded to the knowledge base.



Species Name	Species ID	Obs ID	Obs Date	Obs Time	Obs Date	Lat	Lng	Location ID
thick-spined porcupine	26	7722	2010-11-29T00:00:00.000Z	16:14:00	5.4140601158	118.0324401855	24	6
sun bear	14	8222	2010-12-12T00:00:00.000Z	22:44:49	5.4140601158	118.0324401855	24	2
Storm stork	44	29398	2012-02-06T00:00:00.000Z	13:35:26	5.4140601158	118.0324401855	24	4
Smooth otter	36	27513	2012-03-20T00:00:00.000Z	15:57:56	5.4140601158	118.0324401855	24	7
Rat sp.?	63	26046	2012-05-11T00:00:00.000Z	20:13:35	5.4140601158	118.0324401855	24	5
Pigtail macaque	46	27358	2012-03-01T00:00:00.000Z	09:27:10	5.4140601158	118.0324401855	24	2
otters	11	7751	2010-12-05T00:00:00.000Z	16:12:23	5.4140601158	118.0324401855	24	0
Oriental pied hornbil	58	26940	2011-11-12T00:00:00.000Z	11:23:59	5.4140601158	118.0324401855	24	5
moon rat	29	8219	2010-12-12T00:00:00.000Z	22:26:49	5.4140601158	118.0324401855	24	2
moon rat	29	7794	2010-12-10T00:00:00.000Z	20:40:29	5.4140601158	118.0324401855	24	1
Monitor lizard	61	27346	2012-02-24T00:00:00.000Z	13:13:32	5.4140601158	118.0324401855	24	1
malay civet	8	27402	2012-03-08T00:00:00.000Z	03:19:41	5.4140601158	118.0324401855	24	4
malay civet	8	9200	2010-11-19T00:00:00.000Z	23:54:31	5.4140601158	118.0324401855	24	4
malay civet	8	27376	2012-03-06T00:00:00.000Z	21:17:31	5.4140601158	118.0324401855	24	4
malay badger	3	27191	2011-11-22T00:00:00.000Z	01:49:45	5.4140601158	118.0324401855	24	7
malay badger	3	27512	2012-03-20T00:00:00.000Z	03:03:29	5.4140601158	118.0324401855	24	7

Figure 4.6: Web Interface for Classified Sensed Data

4.3.4 Knowledge Base

The Drools rule engine is a Java-based engine that uses forward chaining inference for the processing of rules [**?**], which means that rules are used to make meaningful inferences about data. Unlike DC nodes, which have limited knowledge processing capabilities, Drools is able to chain rules together and a rule that may not have been triggered at the start of processing may be triggered later if another inference is made.

For example, a rule that is specifically for small mammals may not be triggered until an inference has been made that the image may contain small mammals based on the time and location of the observation.

Drools is able to dynamically update its knowledge base, adding rules and firing them on observations that have already been loaded, as well as newer ones. This allows DP nodes to adapt to new rules and local knowledge whilst they are deployed. The use of *drl* files use a mixture of Drools and Java syntax to define rules, allowing them to modify, or create, Java objects. For example, a rule could be triggered on the receipt of sensed data and create a DwC object from the received data, process it and perform checks on the result that would trigger different rules based on that result. This is one of the main reasons we chose Drools, as it can work with GSN and DwC Java objects, as well as the ability to run on any architecture that supports Java. In order to add rules to the Drools system, knowledge of the Drools syntax and, ideally, Java is required, which does make it a part of the network architecture that requires some specialist knowledge.

The functionality of Drools is extensive and the engine is very powerful, however, it does require specialist knowledge to use and manipulate rules. Using a custom developed Drools web interface , detailed in Section 4.3.3, we created a simplified interface that uses a custom REST API for Drools, allowing users to create sessions, add rules, load data and fire rules, returning the output to the interface. Users can view, and load, existing drl files, shown in Figure 4.7. I

Once a file has been selected, users can view the rules and use the controls on the webpage to perform common operations. The *Load Darwin Core* button loads all DwC archives that are stored in the MySQL database into the current Drools session and the *Fire* button runs all loaded rules on the loaded data. If any of the rules trigger, then the output is presented to the user in the same page, allowing them to act on the results. For example, the location of observations could provide a narrowed down list of potential classifications, allowing an administrator to remove votes that do not match the list.

Figure 4.7: Web Interface for Drools Operations

Users can write new rules based on patterns gleaned from observations received during a K-HAS deployment, which can be found using the web interface and querying the database. These patterns can be encoded as rules into a *drl* file and uploaded to have an immediate effect on the active knowledge base in the network.

Currently, these implementations have been developed for our motivational scenario, but much of these tools are general enough to be repurposed in order to apply to a variety of different WSNs. The Drools API can be used for any kind of sensed data and the web interface would require minor changes to be extensible.

4.3.5 Routing Protocol

The routing protocol we have selected for K-HAS is not fixed for every deployment but, from our research, we modified the commonly used Mininum Cost Forwarding Algorithm (MCFA), outlined in Section 2.2.1, as it allows for changes in the topology

of the network and does not require every node to have a global view of the network.

On deployment of all nodes, a DA node sends out a packet containing the number zero, representing the number of hops to the DA node. Nodes in range receive the packet, store it along with the identifier of the originating node and then send it once it has been incremented by one. The next nodes receive that packet and do the same until the edge of the network is reached. If a node has a number stored that is higher than the one it receives then it is replaced and sent on until the nodes at the edge of the network are reached.

When a node wants to send data, it queries neighbouring nodes and sends to the node with the lowest hop count to the root. We modified this to run in accordance with our tiered architecture as we expect the topology to remain the same for much of the deployment.

In MCFA, nodes do not store path information and messages are broadcast to all nodes when they are sensed.

Firstly, our protocol runs in two modes: configuration and running. During the configuration mode, we use MCFA. Whereas MCFA does not store path information, we use the method ~~desribed~~ described above to store both the hop count and the nearest neighbour. This process is carried out until all nodes of the network have a hop count (and neighbour) and a final pass is made by all nodes to find, and store, their neighbour with the lowest count. The main difference is that, unlike MCFA, not all nodes process all packets. If a packet originates from a DA node, then it is only stored by DP nodes but DP node packets are stored by both DC and DP nodes. DC nodes store it and send all data through their nearest DP node and DP nodes store other DP neighbours to delegate processing to, should a situation arise where they have too much data to process and cause a bottleneck. Configuration mode can then be run at a set interval throughout the deployment of the network, or ~~initiiated~~ initiated manually.

Whilst in running mode, nodes do not query for the neighbour with the lowest hop

count, or broadcast the sensed data to all nodes in range, they send to the node stored as their nearest neighbour. If that fails, then a query is sent out to find other available nodes in range and then sent to the one with the lowest hop count. If the nearest neighbour node is unavailable for more then three attempts, then it broadcasts a request to run the configuration mode again.

4.4 ~~Walkthrough~~Walk-through

In this section we will explain the steps involved in the capture, and processing, of an observation when using the K-HAS architecture. Each tier is responsible for performing different actions upon the observation to ensure it is received by the DA node with an inference as to what it may contain.

Not all of the features described in this section have been implemented within K-HAS and some features have only been tested on a small set of observations. For example, the image classification down to species level is a concept that we have not implemented but template matching is a commonly used practice within image processing [**?**].

4.4.1 Scenario

This ~~walkthrough~~ walk-through will use our motivating scenario and the type of sensed data will be images of animals in the Malaysian rainforest. In this example, we have a collection of wildlife cameras, with nodes attached to them, deployed in the forest. Projects for the rare clouded leopard and sun bear are currently active at Danau Girang. The clouded leopard is a nocturnal carnivore that uses existing paths and hill trails to travel through the rainforest and the sun bear is the smallest bear in the world and sightings are rare. It is also nocturnal and claw marks can be seen on trees that they have climbed. All of this information has been encoded onto the DP nodes and DC

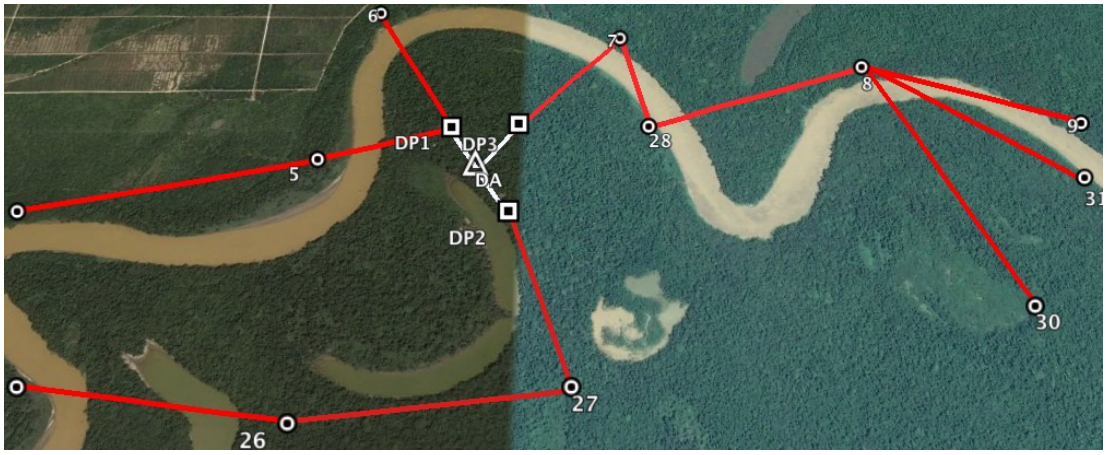


Figure 4.8: Section of Proposed Topology for K-HAS in Danau Girang

nodes know that images taken at night will be of a higher priority, as well as to prioritise further images at night from DC nodes that are deployed on ridges or existing trails.

In order to explain the K-HAS architecture, we need to show the planned topology for the network in Danau Girang, which is based on the positions of the cameras in 2010. Figure 4.8 shows a section of the proposed topology around the field centre, with the rest of the network spreading out along the river on both sides. The triangle icon shows the location of the DA node (at the field centre), with DP nodes (square icons) placed near the DA node because of the poor range of Wi-Fi (Section 3.3.1). Circle icons represent the DC nodes and they link to the nodes with the fewest hops to a DA node, as long as that route includes a DP node.

4.4.2 Data Collection

A DC node is deployed along a ridge in the rainforest and consists of a wildlife camera with a wireless node attached. At 0200, the infra-red sensor detects movement and the camera triggers a set of 3 images to be captured. The DC node creates the DwC archive for the observation. Terms that describe the observation, such as time, date, species identified and location, ~~are~~ is added to the meta.xml file and links to CSV files that contain the data for each term. Any field that can be completed, such as time,

Figure 4.9: ~~clouded~~ Clouded leopard Image Capture

Figure 4.10: Processed Image of clouded leopard

location and date, ~~are~~ is added to the set.csv file. A separate CSV file is created that holds the filename of each image that was taken. The image is shown in Figure 4.9.

The DC node runs its rules on the metadata of the images and infers that the image may contain a clouded leopard, this is because the image was taken in the early hours

of the morning and the camera is deployed on a ridge. DC nodes run simple, non-chaining rules based on the file metadata and details about the node, such as location, but these rules are fixed from the time of deployment and are not updated until the network is redeployed. When ~~bandwith~~ bandwidth is restricted, DC nodes ~~us~~ use a queue to prioritise both their own observations and those ~~receivied~~ received from other nodes. Observations marked as interesting are moved to the front of the queue and any others are sent afterwards.

The inference is included in the archive and is compressed. The node then sends it through every DC node between the originating node and the DP node assigned. To achieve the long range communications in the forest, Digimesh is used; the low transfer rate does mean that an archive can take several minutes to send but it allows for a range of up to 1km.

4.4.3 Data Processing

The DP node receives the observation and it is unzipped and processed by the Darwin Core library. If the data has been preprocessed by a DC node and marked as interesting, then the processing is prioritised, otherwise it is added to the queue. In this case, the DC node believes the observation is interesting and may contain a clouded leopard, so it would be processed before other observations that may be queued and not marked as interesting. The images are read from the filenames provided in the CSV and processed using two methods. The EXIF tags in the image are extracted and the images themselves are processed using the Open Computer Vision (OpenCV) library. A unique feature of the DP node is that it uses two radios to allow links to both DA and DC nodes. DC nodes send archives using Digimesh, to achieve long range ~~communcation~~communication, and DA nodes use Wi-Fi, to provide a faster transfer rate than Digimesh and a more standard method that allows other devices to connect, such as mobile phones or laptops.

EXIF

EXIF (Exchangeable Image Format) tags are written to images at the time of capture. Examples of these tags can be time, date or camera serial number. The capabilities of the camera do affect how detailed the EXIF is, for example, a camera with GPS capabilities will enrich the image with the location.

Wildlife cameras have more functionality than common digital cameras, with details like moon phase, temperature and/or GPS location. Some devices even include the saturation, brightness and hue of each image. These capabilities allow the EXIF to be extremely detailed and this metadata can be used to find patterns in pictures that, when accompanied by local knowledge, assist with the classification of sensed data.

In this example, the knowledge base on the DP node is aware that clouded leopards and sun bears are nocturnal, but clouded leopards have previously only been seen when the temperature is between 30 to 35°C and only when the moon is not full. However, data on sun bear is not as complete and the knowledge base only shows that the bear is nocturnal and can be seen at any time of night in any area of the rainforest. The DP node identifies that the moon phase is not full and that the temperature is 32°C, from this it determines that the image could contain either animal and it cannot make a final conclusion.

Image Processing

Our Triton program, described in Section 3.4.1 is run on the set of three images. These images are converted to black and white and combined to build a background model for the complete set. The detected background is then removed and the final image is then searched for objects, where objects in the foreground will be shown with white pixels. The largest object is then found in the image and extracted to create a template, shown in Figure 4.10.

Processed images of previously sensed images are stored on the DP node and associated with the confirmed classification, confirmed by a human or a node. Although the memory available on a DP node is typically around 32GB, this could easily fill in a matter of months if 3 full HD images were stored for every observation. Storing a single black and white template that contains a portion of the image is much more efficient and can still easily be associated with the classification made. The extracted image is then compared with the existing images, using the knowledge base to prioritise templates for comparison. In this example, nocturnal animals are prioritised and especially nocturnal animals with active projects associated. If the DP node has received an observation from the same DC node recently, then it will check for a classification on that and check for a match there first.

As explained in Section 3.4.1, species classification has not been implemented in K-HAS and templates are currently only associated with human classifications but this walkthrough describes how the process would be carried out if classification were to be implemented. These findings are written into the set.csv file of the DwC archive (Listing 4.4), using the identifier of the DP node as the 'person' that identified the image and the scientific name for the clouded leopard as the species identified in the image. The archive is then zipped and sent on to the DA node, sending observations of interest first and delaying observations that have been found to contain nothing of interest.

This observation is the first trigger from the DC node in the past few hours, so there are no recent classifications. However, processing of the metadata showed that the image was taken at night, so the DP node ~~uses~~ would use its knowledge base to match the images to templates of nocturnal animals first. Triton ~~then finds~~ could then find a match to an existing template of a clouded leopard and completes its classification.

Classification

The metadata processing of the image shows that it could be any nocturnal animal that is known to come out when the moon is not full and the temperature is 32°C. This is not a complete classification but the image processing has found a match. ~~These findings are written into the set.csv file of the DwC archive (Listing 4.4), using the identifier of the DP node as the 'person' that identified the image and the scientific name for the clouded leopard as the species identified in the image. The archive is then zipped and sent on to the DA node, sending observations of interest first and delaying observations that have been found to contain nothing of interest.~~

4.4.4 Data Aggregation

Upon receiving a DwC archive, it is unarchived and processed by Darwin Core libraries, called by the middleware running on the node. The resulting archive is then inserted into a MySQL database and the files themselves are stored in a directory that maps to the DC node that captured the original observation. At the field centre, three users of the system have subscribed to updates for observations of clouded leopards.

As the archive is processed by the library, the species is extracted and this triggers a rule to notify the subscribers. The rule then queries the database and finds their preferred method of communication. In this case, one is a lecturer and wants to be emailed while the two remaining are students and want to be notified via Twitter. An email is generated that contains the time, location and content of the observation, with the images attached, and sent on to the lecturer. The students are sent a short tweet that tells them a clouded leopard has been spotted and a link to the middle image in the sequence is provided; the middle image is used because local knowledge has shown that it is the most likely to contain the full subject in the image. In order to maintain privacy, a 'direct message' can be sent on Twitter so this ~~the~~ message is not public.

The middleware on the node supports a web interface to allow users to perform ad-

ministrative functions on the network, such as deploy a new node, on top of this there is a custom made website that shows all observations for every DC node. This allows users to log on and classify the images. In this case, the lecturer receives the email notification, reviews the attached images and clicks on the link to access the website to inform the DP node that the classification was correct. Due to the administrator position the lecturer has on the system, he is able to stop any users voting on the image and to simply confirm the classification.

When a user classifies the observation, they see that a clouded leopard has been spotted in the same area on the same day for the past 5 weeks and they create a rule to automatically classify images from this camera that have a similar time (within an hour) and have an object extracted from them by the image processing. The user can then upload the rules through the same web interface and it will instantly become active on the system.

If there was no classification, then users would be able to vote on the contents and use the classification with the highest vote, or the classification made by an administrator. Once a classification has been made, it is stored in the database and written to the archive. This triggers the DA node to send that classification on to the DP node that sent the original archive. In this case, the DA node informed the DP node that it has been confirmed as a clouded leopard and the DP node then stores the extracted image in the directory of clouded leopard templates, to assist with future classifications. This updated template causes the rule base of the DP node to be updated so similar data processed in the future would be correctly identified. The longer the network is deployed, the more knowledge DP nodes could gain and the more accurate their classifications can be. For example, a change in season could cause a new, previously unknown animal to migrate to the rainforest. With human assistance, the animal can be identified and determined whether it is of interest. This knowledge can then be stored and sent on to DP nodes to prioritise and classify correctly the next time that it is captured. The feedback loop protocol is currently only a design and has been

minimally implemented as a proof of concept, we believe it can improve the accuracy of classification in a WSN whilst ndoes are still deployed but it would require testing in the future.

4.5 Conclusion

In this chapter we have explained the architecture we have developed to allow knowledge to be encoded and utilised within a wireless sensor network. Using tiers of nodes, with varying levels of knowledge-processing capabilities, we can process observations within the network and deliver data that has, where possible, already been classified. Working as more of a subscribe-push method, users do not have to check a DA node for new data, instead it is sent to them if it has been found to be part of a project they are subscribed to. If not, then the data is accessible to all users of the network through a web interface.

Using GSN as the application middleware allows sensors to be added, modified and maintained by those without technical knowledge and ensure future interoperability. However, it is not a standardised approach and using an architecture that implements the standard OGC SWE[**?**] would allow for interoperability through the use of standards not just for sensor networks but the Web as well.

One of the key features of K-HAS is that it is not a static deployment. The knowledge that the network holds at the time of deployment will rarely be the same as the knowledge held after a few months. Humans enrich the existing knowledge base and the nodes are able to make inferences about the data they are sensing, improving their classifications the longer they are deployed. When developing this architecture, GSN was the most robust architecture of those that we researched and tested, the support for many databases, administrative interface, native support for many widely used sensors meant that it was a better choice than a middleware that adopted the OGC SWE standards, especially as we were not interacting with SWE systems in our motivating

scenario. However, while GSN is stable and mature, its large codebase does mean that there are dated features, such as using SOAP instead of REST and an unintuitive web interface that does not utilise web sockets. A middleware with similar automation on receipt of sensor data could be used instead of GSN that did follow the standards set out by the OGC. We believe that this should require few changes to the core K-HAS architecture as it currently stands.

In Chapter 6, we explain how we implemented a variation of K-HAS in our motivating scenario and Chapter 7 shows our evaluation of the K-HAS architecture, but Chapter 5 will first outline the development of an ontology to support the architecture described here.

Chapter 5

An Ontology for Knowledge-Based Wireless Sensor Networks

In this chapter, we explain the ontology we have proposed to formally define the components within K-HAS, the structure of the sensed data, the users involved with the network, as well as the format of the sensed data that is passed through the network.

This ontology is for those wanting to deploy a WSN that uses knowledge to classify the ~~scientific~~ ecological observations recorded, or even to select parts of the ontology that meet their requirements and implement those. Terms have been used to map to widely-used ontologies in the domains of ~~scientific~~ ecological observation, sensors and people. However, our proposed architecture (Chapter 4) defines new terms that do not exist in current ontologies, thus we extend the alignment of these existing ontologies to create an ontology that encompasses the ontologies as well as defining new terms to match our architecture.

5.1 Background

K-HAS was developed in order to provide a generic architecture for wireless sensor networks to utilise the local knowledge contained within their environment to process sensed data and, therefore, make more efficient use of the network bandwidth by prioritising sensed data that is deemed to be more valuable. We have defined local

knowledge as knowledge of an area that has been gained through experience, or experimentation.

Before we were able to implement K-HAS, we needed a high-level model that defined relationships between each tier of the architecture, as well as the data standard used to transport sensed data from DC nodes to an endpoint. Developing an ontology for K-HAS means that we can do this, as well as provide a computer-readable model for all of the classes and components used by K-HAS.

Making it computer readable has allowed us to reuse classes in the development of software for each tier. For example, we were able to develop a common Darwin Core java library that is used in this ontology and in our GSN middleware to unzip and process received archives.

During the development of the K-HAS ontology, we researched existing ontologies that were commonly used in the domains that K-HAS covered. These included scientific observations and sensor hardware.

Looking into these existing ontologies, we found that there had been many surveys on representing sensors in the semantic web: [29], [**?**], outlines the existing work. These surveys clearly highlighted that these ontologies had been split into two branches; observation-centric and sensor-centric.

Observation-centric ontologies, such as OBOE [**?**] and O&M [**?**], focus on the data that is sensed, and its content; whereas sensor-centric ontologies, such as SSN [30] and SensorML[**?**], detail the components that make up a sensor and the operations they perform to turn sensed data into an output.

~~We found one existing ontology, the Standard Upper Merged Ontology (SUMO), that linked these concepts together to show the flow of data within the network and the role that each sensor plays in delivering this data, discussed later in Section **??** and Section 5.1.3. The reason there are so few could be because other WSNs do not integrate structure, hardware and sensed data in the same way as K-HAS or due to the fact that~~

~~most WSNs do not process their sensed data until they reach a base station, so they would not need to model their data structure within the network. Because of this, we developed an aligning ontology that reuses existing ontologies, where possible, and introduces new classes that allows these ontologies to interlink. The result is an ontology that covers the flow of sensed data from the point of capture to the point it is received, processed, reviewed and stored at the end point of the network. Not only does the ontology cover how the data changes as it flows through the network but also the roles and capabilities of each tier.~~

K-HAS has been developed to be used with any sensors and is not specific to wildlife cameras, therefore we also looked into sensor-based ontologies that concentrate on the hardware and the individual capabilities of each device within the network.

5.1.1 Observation-Centric Ontologies

OBOE

The Extensible Observation Ontology (known in reverse as OBOE) is a popular suite of ontologies used to represent scientific observations [**?**]. Initially starting as base ontology for ecological observations, it has now grown into a suite of extensions that make it suited for chemistry, bioinformatics, anatomy and others. OBOE is represented by OWL-DL [**?**] and allows the characteristics of a generic scientific observation to be linked to domain specific characteristics.

OBOE focuses on the concept of an *observation*, which is made up of an entity, a measurement and a characteristic [**?**]. An example of an observation could be a researcher observing an animal (the entity) and recording the gender (the measurement) as male (the characteristic). A single observation could then consist of multiple observations within it, such as gender, location, species and the number of species observed.

Figure 5.1 shows the basic structure of a core OBOE observation, outlining the five key

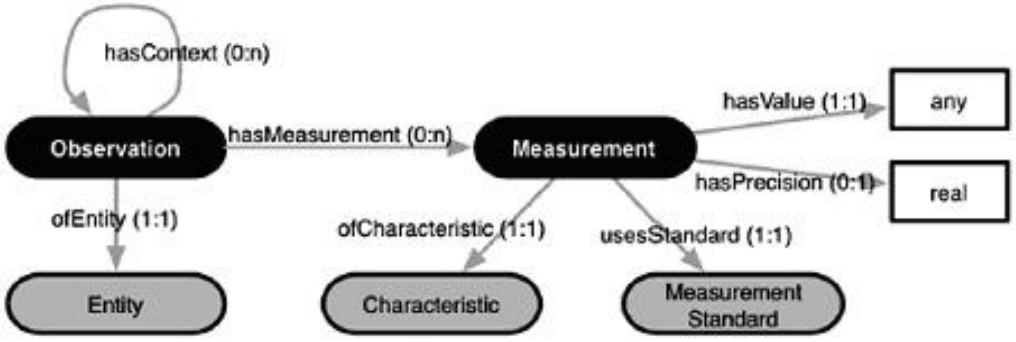


Figure 5.1: The Core of an OBOE Observation [?]

classes that are linked by seven properties. While this is the core structure of OBOE, domain specific extensions have been implemented by utilising 'extension points' that are part of OBOE's core. OBOE follows the O&M Standard (below) very closely, providing extensions to the core classes that allows more information about each to be encoded, adding context and enhancing the value of an observation.

The primary benefits of OBOE are that it is generic enough to cover almost all types of scientific observation and domain extensions allow for more specific details to be stored.

O&M

The Open Geospatial Consortium (OGC) Observations and Measurements Standard aims to provide a framework suitable for recording any observation made by a sensor, regardless of the domain [**?**].

A key feature of O&M is that *observation* and *measurement* are not just classes. They also denote an action. An observation is an action that causes a result, yielding a value and a measurement is a set of operations that provide some result(s).

O&M provides a conceptual model, as well as XML encoding for observations and measurements. The listing 5.1 shows an observation of a vehicle in a given time and

place. Similar to the encoding of a measurement with the standard and protocol used in OBOE, O&M provides support for the recording of the procedure used to gain the measurement for the observation.

```
<?xml version="1.0" encoding="windows-1250"?>
<om:GeometryObservation gml:id="geom1610"
xmlns:om="http://www.opengis.net/om/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xsi:schemaLocation="..Specialization_override.xsd">
  <om:samplingTime>
    <gml:TimeInstant>
      <gml:timePosition>2009-09-16T17:22:25.00</gml:timePosition>
    </gml:TimeInstant>
  </om:samplingTime>
  <om:procedure xlink:href="urn:ogc:object:procedure:ifgi:GPS"/>
  <om:observedProperty xlink:href="urn:ogc:def:phenomenon:OGC:Shape"/>
  <om:featureOfInterest xlink:href="urn:ogc:object:feature:vehicle"/>
  <om:result>
    <gml:Point srsName="urn:ogc:crs:epsg:4326">
      <gml:pos>40.7833 -73.9667</gml:pos>
    </gml:Point>
  </om:result>
</om:GeometryObservation>
```

Listing 5.1: An Observation of a Vehicle encoded in O&M

The listing shows that an observation centres around a *feature of interest* that can be a physical object and the measurement is the detection of the vehicle at the recorded location. Figure 5.2 shows a basic diagram of the ontology. While the event of a feature is linked, it is clear that the main focus is on the observation and the measurement associated with it.

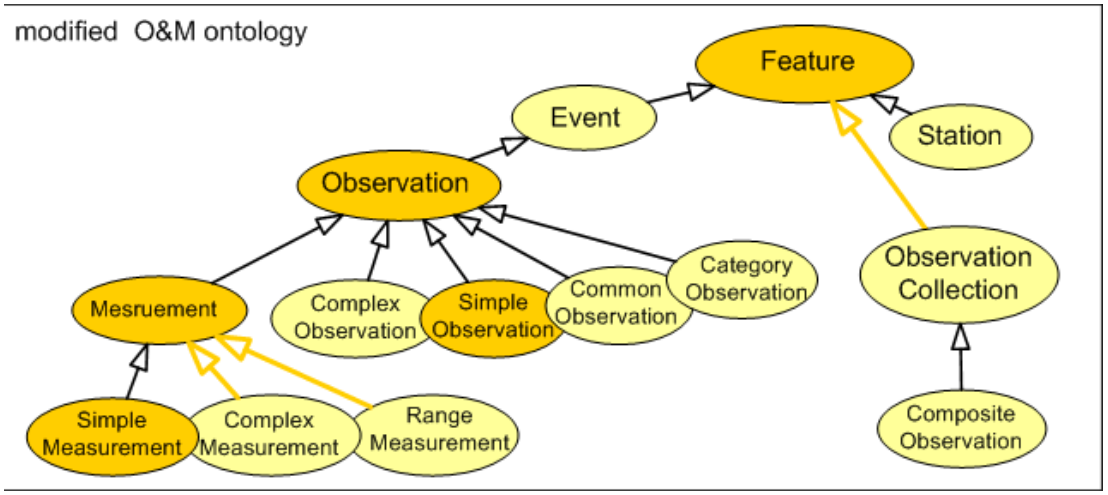


Figure 5.2: The OGC O&M Ontology [?]

The structure of an observation within O&M is focussed on the action and the result, this makes it suited for sensor networks across many domains that perform a wide range of observations. Because the users of the O&M standard are spread across many domains, each with their own terms and definitions, the creation of an ontology for the standard aimed to remain as generic as other observation-centric ontologies.

Darwin Core SW

In order to represent DwC occurrences, covered in Section 4.3.1, in an ontological format, work has been done to represent Darwin Core terms, as an ontology, in OWL. Darwin-Semantic Web (Darwin-SW) [19] is the project that aims to do this and many of the core terms associated with an occurrence have already been formalised and Figure 5.3 shows the entity relationship model of these terms.

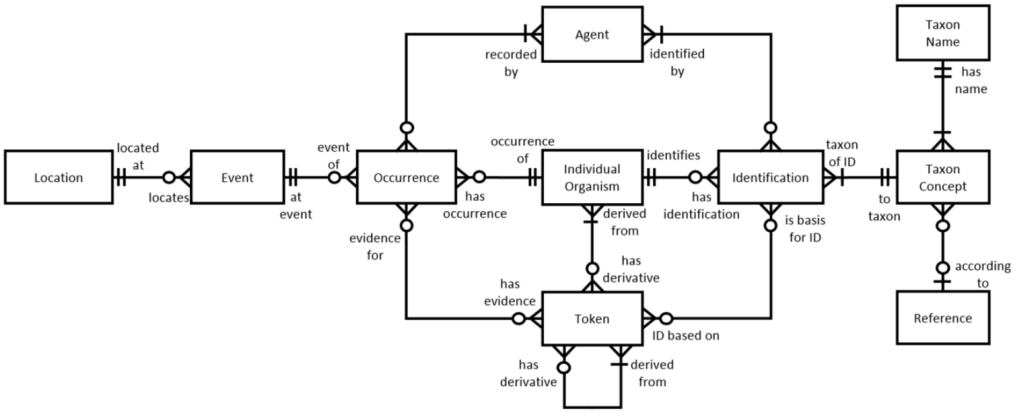


Figure 5.3: Darwin-SW Entity-Relationship Model [19]

While Darwin-SW does not represent all of the classes within the DwC namespace, it contains the core classes required to record an occurrence, to a more-detailed level than OBOE. The downside of this is that the specificity of the terms limits DwC to ecological observations, making it far less generic than OBOE and other alternatives. There are positive aspects that, for those that need to record ecological observations, DwC allows for a greater level of detail and can be mapped to OBOE with ease.

Listing 5.2 shows a fragment of a DwC occurrence of a living specimen which, ~~is~~ in this case, is a tree ~~.~~[18]. Darwin-SW, and Darwin Core, terms are used to describe the identification of the tree, such as the by whom it was identified, the date and the taxon. The full source can be seen in Appendix **??** and includes links to existing related occurrences, other images of the individual and relationships to other resources.

```
<dsw:hasIdentification>
  <rdf:Description rdf:about="http://bioimages.vanderbilt.edu/vanderbilt
      /12-126#2002-04-10baskauf">
        <dcterms:description>Determination of Ginkgo biloba L. sensu Flora of
            North America (1993) for the individual http://bioimages.
            vanderbilt.edu/vanderbilt/12-126</dcterms:description>
        <rdf:type rdf:resource="http://rs.tdwg.org/dwc/terms/Identification" /
            >

        <!-- In lieu of stable external identifiers for taxon concepts, Im
            defining some onsite -->
        <dsw:toTaxon rdf:resource="http://bioimages.vanderbilt.edu/
            taxonConcepts#183269-fna1993" />

        <dwc:identifiedBy>Steven J. Baskauf</dwc:identifiedBy>
        <dsw:idBy rdf:resource="http://bioimages.vanderbilt.edu/contact/
            baskauf" />
        <dwc:dateIdentified>2002-04-10</dwc:dateIdentified>

        <!-- Relationship of the identification to other resources -->
        <dsw:idBasedOn rdf:resource="http://bioimages.vanderbilt.edu/baskauf
            /10554"/>
  </rdf:Description>
</dsw:hasIdentification>
```

Listing 5.2: Darwin-SW Representation of an Identification

~~SUMO - Sensor Data Ontology~~

~~The Suggested Upper Merged Ontology (SUMO) is an ontology that links sensor hardware and sensor data ontologies to assist with searching and evaluating distributed and heterogenous sensor networks [**?**]. SUMO combines the Sensor Data Ontology (SDO) and Sensor Hierarchy Ontology (SHO), explained in Section 5.1.3, with the ability to 'plug in' extension ontologies.~~

~~The SDO describes the sensing properties of a device beyond the hardware and the context of the sensor can be enriched with information about spatial and/or temporal observations. Similar to GSN (Section 2.3.2), SUMO uses the notion of virtual sensors where a group of sensors can be described together to provide abstract measurements.~~

~~In [**?**], the example of a humidity sensor, temperature sensor and wind speed sensor being collectively described as *weather sensors*.~~

5.1.2 Sensor-Centric Ontologies

Sensor-centric ontologies are more focussed on the structure of the sensor, the network and the sensing processes involved.

SensorML

The Sensor Markup Language (SensorML) Standard has been developed by the OGC, and complements the O&M Standard, to enable the discovery and tasking of internet-connected sensors [**?**].

SensorML provides an XML schema for describing a sensor, its capabilities and the processes available. At the core, SensorML comprises of:

1. Component - A physical process that transforms information from one form to another.
2. System - Model of a group of components.
3. Process Model - Atomic processing block used within a Process Chain.
4. Process Chain - Composite block of Process Models.
5. Process Method - Definition of the behaviour of a Process Model.
6. Detector - Atomic part of a Measurement System.
7. System - Array of components, relates a Process Chain to the real world.
8. Measurement System - Specific type of System, mainly consisting of sampling devices and detectors.

9. Sensor - Specific type of System that represents a complete Sensor.

These definitions outline the core concepts of SensorML, a *system* that performs one (or more) process(es) and is comprised of a group of *components* [**?**]. A SensorML document allow for a general, formal specification of a *sensor* and its capabilities. The document describes a *component*, outlining what data it reads in and the output once it has been processed. Several of these *components* can the be used to create a *system* and the primary goal of SensorML is to describe the process of how an observation came to be, focussing on the technical features of the node.

5.1.3 Observation and Sensor-centric Ontologies

SSN

The Semantic Sensor Network (SSN) Ontology is the most fitting ontology for our requirements, as it covers systems processes and observations. Developed by the W3C after an extensive review of existing ontologies [30], the SSN ontology is designed to allow focus on a variety of perspectives, such as the sensor within the network or the data that has been observed.

Figure 5.4 shows the model for the SSN ontology, displaying the relationships that connect each class, as well as their associated properties. It also highlights the modular approach that has been taken, separating the system from the process and the observation.

The observation pattern of SSN is centred around ‘Stimuli-Sensor-Observation’, which can be simply described by an event causing a sensor to trigger and create an encompassing observation to store details about the event, as well as the device that recorded the event. While SSN is focussed on sensors, capturing the measurement capabilities of each sensing device that makes up a system and specifics on its lifespan, the development of the ontology arose from the review of sensor-centric ontologies as well,

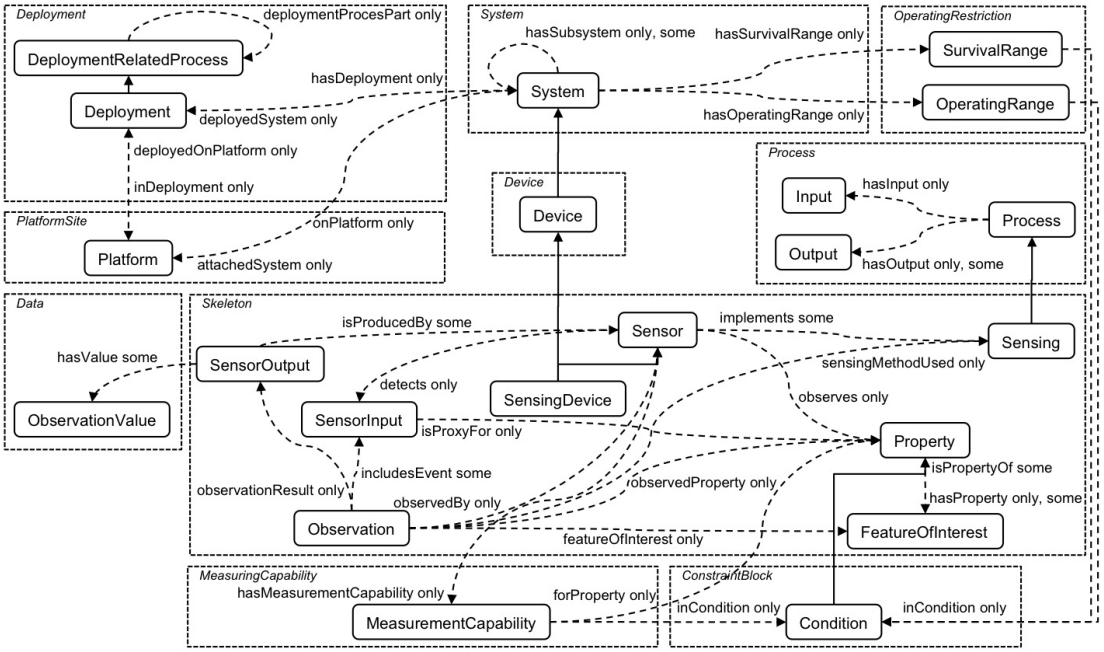


Figure 5.4: SSN Ontology

recording data about observations ~~is~~ in a structure very similar to O&M.

SUMO ~~- Sensor Hierarchy Ontology~~Extension

The ~~SHO component of SUMO~~ Suggested Upper Merged Ontology (SUMO) is a general purpose ontology [**?**] that provides general-purpose terms and is intended to be extended for domain specific ontologies. In [**?**], SUMO has been extended to link sensor hardware and sensor data ontologies in order to assist with searching and evaluating distributed and heterogenous sensor networks. The work combinesthe Sensor Hierarchy Ontology (SHO), the Sensor Data Ontology (SHO) and the ability to 'plug in' extension ontologies.

The SHO describes the hardware of a sensor node, as well as its accuracy, transmission medium and data processing capabilities. The SDO, however, describes the sensing properties of a device beyond the hardware and the context of the sensor can be enriched with information about spatial and/or temporal observations. Similar to GSN (Section 2.3.2), SUMO uses the notion of virtual sensors where a group of sensors can be

described together to provide abstract measurements. In [?], the example of a humidity sensor, temperature sensor and wind speed sensor being collectively described as *weather sensors*. The SHO ontology is another extension of SUMO, from [?], that represents the hardware of a WSN, including the node itself, data transmission units, data processing units and individual sensors. The data model describes a sensor with metadata describing features such as measurement type, transmission range and physical properties.

~~**Findings**~~

5.1.4 Findings

There are many ontologies that are suited for sensor-based scientific observations and ontologies that allow for the description of sensor hardware are very complete. However, in our research, we were unable to find a complete ontology that allowed for a sensor to be described as well as specifics of the observation. SSN and SUMO were the closest that we found for this, but SSN was still very hardware focused and aimed at a technical audience. SSN primarily describes the hardware of each node and their capabilities, whereas our requirements are for both hardware and sensed data to be represented in a single ontology.

While ~~SUMO is~~ the SHO and SDO extensions of SUMO are classed as both an observation-centric and sensor-centric ~~ontology, because it does link~~ ontologies, they are separate ontologies that model both sensor hardware and sensed data and, while ~~SUMO's development~~ the development of these ntologies did follow a similar approach to the one described in this chapter (combining ontologies) it is more focussed on retrieval of sensed data through queries constructed using both hardware details and properties of the sensed data and does not fulfil all of the requirements we identified when developing K-HAS, such as representing users. However, an important feature ~~of SUMO is~~ is the Extension Plugin Ontologies (EPO), support for domain specific plugins to integrate with

SUMO and integrate with the SHO and SDO and this would also be a useful feature for K-HAS.

We found that many of these ontologies satisfied many subsections of K-HAS completely, meaning that it would be unnecessary to reproduce these concepts. From these findings we have developed an ~~alignment~~ aligning ontology that connects ontologies across multiple domains to support our proposal of K-HAS, creating an ontology that is both sensor-centric and observation-centric.

The SSN ontology is a modular ontology created by combining concepts from researching existing, commonly used ontologies and allows for domain specific concepts to be imported. Some of the main uses cases for the SSN ontology are provenance and data discovery, which are also key within K-HAS. However, the tiered structure of K-HAS did not map directly to SSN and it proved difficult to represent the flow of knowledge through a network, as humans can also perform similar operations on observations and the observations are enriched as they pass through the network. However, while we did not use the ontology directly, many of the concepts can be mapped directly and it would be possible to modify K-HAS to leverage the modular nature of SSN and extend it.

5.2 Method

Before we began development, we researched existing methods that have been used to develop current ontologies. From this, we found three well-documented methodologies; the methodology used to develop the Toronto Virtual Enterprise (TOVE) ontolgy, the Methontology and the methodology used to develop the Enterprise Model ontology.

The methodology by Uschold and King [**?**], used to develop the Enterprise ontology, is a four step method that was most suited to the processes we expected to follow. The steps are not covered in great detail in the original paper but, research since provides greater detail for each step.

5.2.1 Identify Purpose

The aim of this step is to identify why the ontology is being built and what requirements it is supposed to fulfil. This includes considerations, such as the audience, the intended purpose and the specificity of the ontology.

Other methodologies have used more structured methods to informally identify the requirements of the ontology. Gruninger and Fox [**?**] propose competency questions; these are questions that are used to identify the problems that the developed ontology is developed to solve. These questions can act as a benchmark, showing that an ontology is sufficient if it can solve the questions raised here.

5.2.2 Build the Ontology

Building the ontology can be broken down further into 3 smaller steps: capture, code and integrate existing ontologies.

Capture

The capture stage involves defining the concepts and terms that the ontology will model, as well as how they map to the real world. This can be done in one of three ways:

1. Top Down - Starting with the core conepts, create the more specialised classes until you have identified all subclasses.

2. Bottom Up - This process begins with the more specialised classes and grouping them into the more general classes towards the top of the heirarchy.

3. Middle Out - A combination of the two, this process involves specialising, and generalising, the classes identified in the middle layer of the hierarchy.

As for the capture of the knowledge used to identify the classes needed, this is an area that has been documented, but primarily provides recommendations. [**?**] suggests interviews with domain experts, iteratively brainstorming with a group actively involved in the development of the ontology. This stage has often been referred to as the knowledge-acquisition stage.

Code

This step involves coding the terms identified in the previous step into a formal language, such as the Web Ontology Language (OWL) [**?**].

Integrate Existing Ontologies

This step can be carried out at the same time as the step above, so that the ontology is developed with existing ontologies in mind. This also allows overlap to be identified, and incorporated, early. Being aware of commonly-used ontologies within the domain, before development begins, is a more logical approach and does avoid the need to create new terms for existing concepts.

Existing ontologies can be integrated by importing them and linking the existing terms to the terms identified in the ontology that is being developed. However, there is also the method of developing the ontology completely, so that it is consistent without the need to rely on existing ontologies and creating 'sameAs' relationships between the terms identified and only the required terms in the existing ontologies.

In [**?**], it is recommended that, when importing external ontologies, the whole ontology is not used. Rather, one should aim to extract only the required fragments from the external ontology that are relevant to the concepts in the developed ontology.

5.2.3 Evaluation

For the evaluation, Uschold and King adopt the definition of [**?**]: ‘to make a technical judgement of the ontologies, their associated software environment, and documentation with respect to a frame of reference... The frame of reference may be requirements specifications, competency questions, and/or the real world’.

There are some well documented methods for evaluating ontologies in the literature, [**?**] proposes using the competency questions, used in the first step, to ensure that they can all be fully answered by the finished ontology.

5.2.4 Documentation

Although this step is listed at the end of the development cycle, it seems more fitting to document all major aspects of the ontology as it is being developed. Documentation of the ontology should include: any assumptions, all concepts introduced, all ontologies that have been incorporated (by whatever means) and any primitives used for the definitions of concepts.

5.3 Results

This section details our results when using the Uschold and King methodology, accompanied by more recent research for particular steps, to develop the K-HAS ontology.

5.3.1 Identify Purpose

The need to create the K-HAS ontology was partly due to the reasons outlined by Gruninger and Fox [**?**], we had identified a problem as we were developing a sensing

architecture that utilised local knowledge: how could we formally represent the flow of knowledge and sensed data throughout a wireless sensor network?

To determine the scope that K-HAS should cover, we identified a set of competency questions that represent what we expect K-HAS to cover within the domain. In order to present this, we used an approach similar to that of [26], which can be seen in Table 5.1.

Competency Questions
Find all **Occurrences** of an **Individual**
Find all **Occurrences** of an **Individual** at a **Location**
Find all **Occurrences** within a specified **Date** and **Time** range
Find all **Sensors** that have recorded an **Occurrence** of an **Individual**
Find all **Locations** of an **Individual**
Find the storage location of a **Project**
Find all **Projects** containing an **Individual**
Find all **People** involved in a **Project**
Find all the **Evidence** that supports an **Occurrence**
Find all the **Types** of evidence that supports an **Occurrence**

Table 5.1: Competency Questions

These questions allow us to identify the core concepts that the ontology needs to represent, as well as serving as a tool to evaluate the completed ontology.

5.3.2 Build the Ontology - Capture

This part of the development cycle is the identification of the concepts and their implementation. The first step is capture the knowledge that will be used to identify the core concepts within the ontology.

To capture the knowledge for the ontology, we used a combinatorial approach of those outlined in the literature. We interviewed domain experts, as well as overseeing a basic implementation of a system, and we iteratively brainstormed the concepts throughout the development cycle.

Over the course of eighteen months, which involved 2 field visits and several brainstorming meetings, we identified the core concepts that the ontology would need to contain, as well as the properties that would link them. From this, it would seem that we followed the *Top-Down* approach, but the first visit with domain experts also allowed us to identify some of the more specialist classes early on in the development cycle. Thus, it seems that in practice we followed a more *Middle-Out* approach.

Table 5.2 outlines the core concepts we have identified for K-HAS to be complete, as well as the definitions we have used for the architecture. When integrating existing ontologies, the definitions of similar terms would need to match with our definitions or we would not deem them the same as the K-HAS concepts.

Mapping these concepts, a diagram of the base K-HAS ontology is shown in Figure 5.5. The next step is to create the ontology and map the concepts identified to existing ontologies within the same domain space(s).

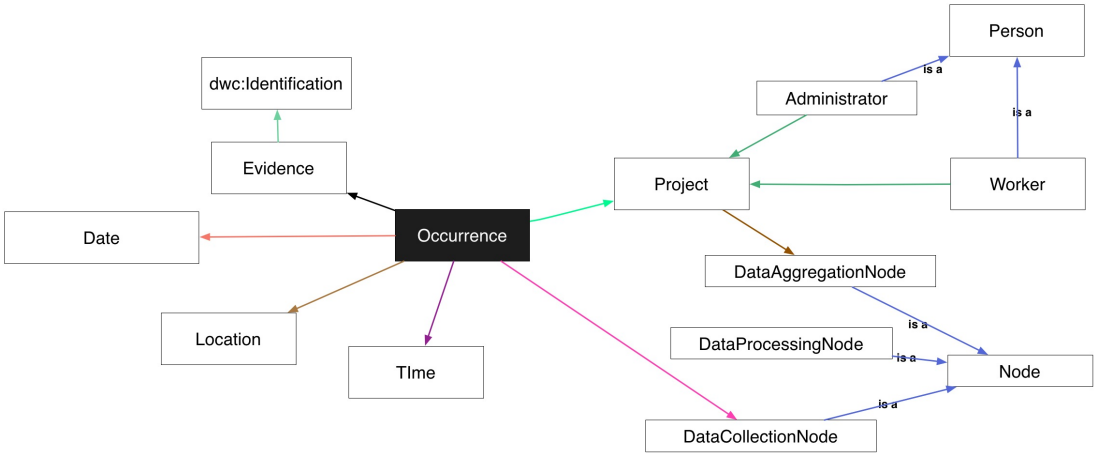


Figure 5.5: K-HAS Base Ontology

5.3.3 Build the Ontology - Code

Once the concepts had been identified (and agreed upon), the ontology must be coded. We used Protege 4.2.2 [**?**] and implemented K-HAS in the Web Ontology Language

Concept	Subclass Of	Definition
Occurrence	-	
Identification	-	A text-based recording of the content of an Occurrence
Evidence	-	Media to support the Identification, such as a photo or recording.
Location	-	The location of the Occurrence.
Date	-	The date of the Occurrence.
Time	-	The time of the Occurrence.
Project	-	Project(s) that can contain many Occurrences
Node	-	A device with sensing capabilities.
Data Collection (DC) Node	Node	Node with limited knowledge-processing capabilities charged with sensing a feature (or features) of its environment.
Data Processing (DP) Node	Node	Node with knowledge-processing capabilities charged with serving a subset of DC Nodes and processing their sensed data.
Data Aggregation (DA) Node	Node	Node with knowledge-processing capabilities that stores all knowledge and sensed data for the whole network.
Person	-	
Administrator	Person	Person in charge of a project (or projects).
Worker	Person	Person involved with a project.

Table 5.2: K-HAS Concepts

(OWL), creating a core file that could be expanded; should we need to import existing ontologies.

5.3.4 Build the Ontology - Integrate Existing Ontologies

Our proposal for K-HAS is focused on scientific observations. Because of this, the focal point of our existing ontology research is on ontologies that are centred around scientific observations. This allows us to create a more generic ontology that is still able to capture all of the semantic details associated with a wildlife observation.

As explained in Section 5.1, our research of existing ontologies covered two categories: observation-centric and sensor-centric. We identified ontologies, within multiple domains, that satisfied many of our requirements for K-HAS, but not all. Using the core concepts outlined in Section 5.3.2, we created a minimal ontology and used the results of our research to map K-HAS concepts to those that had already been identified.

During this process we found that we had determined concepts that mapped to existing ontologies, but may not share identical structures, or even the same name. For example, the OBOE ontology describes the concept of an occurrence, which is very similar to our occurrence and the occurrence in the Darwin Core-SW ontology. When we found these mappings, we used *sameAs* relationship to form a link that allows data stored according to these existing ontologies to map directly to K-HAS. However, the structure of an OBOE observation is, as outlined in Section 5.1.1, more generic for all types of scientific observation and depicts the observation of an entity, containing a measurement of a particular characteristic. Whereas the structure of a Darwin Core observation is more limited to scientific observations of taxa which allows it to have more predefined terms, such as location, species name and the evidence for the recorded individual. Because of this, it was difficult to create a structure that encapsulated both OBOE and Darwin Core due to the generality of an OBOE observation compared to the more specific structure of DwC.

Research showed that Darwin Core maps to K-HAS' requirements more completely, as well as the fact that there has been work to represent Darwin Core Observations in OBOE [**?**], K-HAS occurrences map directly to Darwin Core and elements that are

similar to OBOE have been linked by the *sameAs* relationship. This does mean that full OBOE observations cannot map directly, but K-HAS, and Darwin Core, occurrences can be converted, if necessary. However, using the terms in our ontology we can create a subset of an observation that does not include all of the terms defined in OBOE but can still map to all three ontologies.

For the sensor hardware of K-HAS, we found that the SensorML ontology maps directly to our concepts and we also realised that we did not need to recreate concepts that may already exist in popular ontologies outside of the domain spaces we researched. For example, the Friend of a Friend (FOAF) ontology [22] is an ontology designed to create machine-readable pages that describe people, so it is more logical that K-HAS reuses existing terms from popular ontologies to allow pre-existing data to be mapped with ease.

5.3.5 Extend the Ontology

Whilst researching widely-used ontologies, we became aware that some classes identified for K-HAS were not satisfied by what is currently available, these are prefixed with khas and shown in Figure 5.6. The final step for creating the ontology was to add these concepts to the linked ontologies, we call these *extension concepts*, concepts unique to K-HAS which do not exist in any other ontology. These terms were linked to the unique layered architecture of K-HAS and we defined them within a new K-HAS namespace, changing our ontology from an alingment of existing ontologies to an extension of these.

The final ontology is shown in Figure 5.6 (with the complete code in Appendix **??**) and shows how each concept maps to existing ontologies. The figure shows that there were only five extension concepts unique to K-HAS that can be subclassed from the *person* concept in the FOAF ontology and *node* in the SensorML ontology.

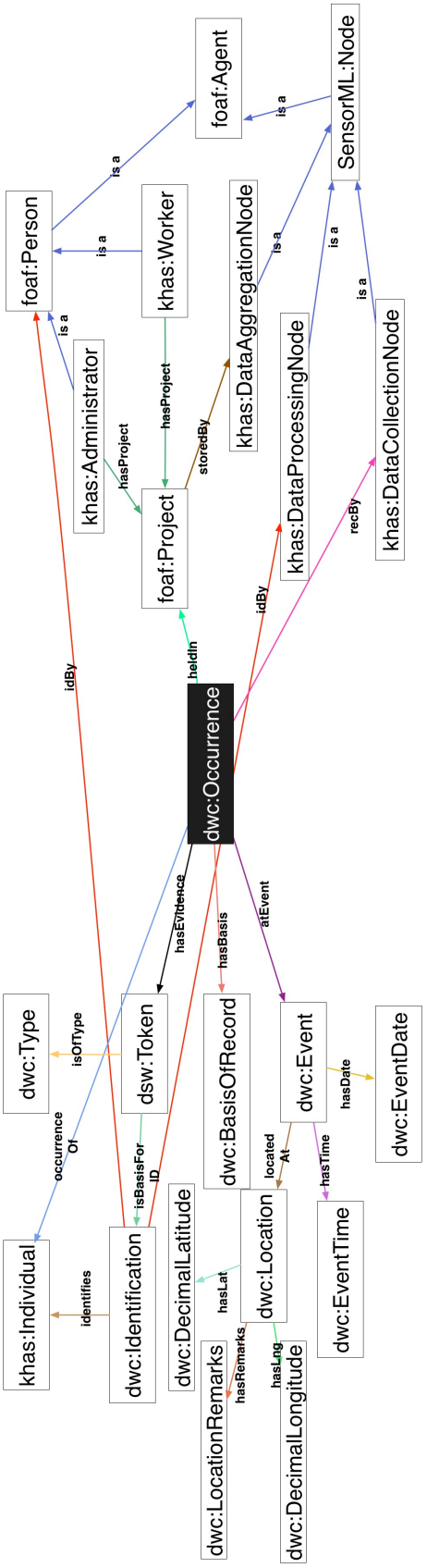


Figure 5.6: K-HAS Ontology

5.4 Evaluation

Uschold and King's ontology development method does not explain how to evaluate the created ontology and much of the literature describes a number of methods that can be employed, [21] outlines a number of different evaluation techniques and separates them into categories. These categories are listed below:

1. Comparing the ontology to a 'golden standard'.
2. Using the ontology in an application and evaluating the results.
3. Comparing the ontology with a collection of documents from the domain to be covered.
4. Manual evaluation by humans to test if the ontology meets a set of predefined criteria.

The last category mentioned is similar to the method in Section 5.2.3 that uses the competency questions to determine the effectiveness of the ontology. Because there is no agreed method,we have evaluated K-HAS by testing it in the application it was developed in (Protege), mapping it to existing documents within the intended domain and ensuring that it can satisfy all of the competency questions we outlined in Table 5.1.

5.4.1 Using the Ontology in an Application

We developed the K-HAS ontology in Protege, a Java based ontology editor, that also provides the functionality to reason over the ontology and check for inconsistencies. Using the Hermit reasoner that is built into Protege, the ontology was determined to be logically consistent.

To confirm the validation provided by Protege, we also used a web-based ontology validation tool, called the OntOlogy Pitfall Scanner (OOPS), that scans an ontology for common errors, such as defining incorrect inverse relationships or using recursive definitions, that can occur during the development phase [**?**]. The results of this tool showed that no pitfalls had been detected.

We have a customised build of K-HAS running that receives data from a variety of sources and stores it in a MySQL database that allows users to classify through a web site. The schema of the database maps to our ontology, scientific observations are received, unzipped and stored in the relevant fields in the database. Using real sensed data, we have successfully stored over two hundred occurrences and mapped existing DwC occurrence to K-HAS.

5.4.2 Competency Questions

In Table 5.1 we have identified some basic competency questions that we expected K-HAS to be able to satisfy with the concepts we had identified, Table 5.3 shows how the original questions are satisfied by K-HAS. We have mapped the terms we originally used to the terms used in K-HAS (and the linked ontologies) and shown the concepts involved with each question, as well as the relationships linking them.

Competency Questions	Concepts	Relationships
Find all **Occurrences** of an **Individual**	Occurrence; Individual	Occurrence occurrenceOf Individual
Find all **Occurrences** of an **Individual** at a **Location**	Occurrence, Token, Individual, Location, Identification	Occurrence hasEvidence Token Token isBasisForID Identification Identification identifies Individual
Find all **Occurrence** within a specified **Date** and **Time** range	Occurrence; Event; Date; Time	Occurrence atEvent Event Event hasTime Time Event hasDate Date
Find all **Data-Collection Nodes** that have recorded an **Occurrence** of an **Individual**	DCNode; Individual; Occurrence	Occurrence recordedBy DCNode Occurrence occurrenceOf Individual
Find all **Locations** of an **Individual**	Occurrence; Individual; Event; Location	Occurrence occurrenceOf Individual Occurrence hasEvent Event Event locatedAt Location
Find the storage location of a **Project**	Project, Data Aggregation Node	Project storedBy DataAggregationNode
Find all **Projects** containing an **Individual**	Individual, Project; Occurrence	Individual occurredIn Occurrence Occurrence heldIn Project
Find all **Persons** involved in a **Project**	Person; Administrator; Worker; Project	Administrator hasProject Project Worker hasProject Project
Find all the **Evidence** that supports an **Occurrence**	Occurrence; Token	Occurrence hasEvidence Evidence
Find all the **Types** of evidence that supports an **Occurrence**	Occurrence; Token; Type	Occurrence hasEvidence Token Token isOfType Type

Table 5.3: Competency Questions

The table shows that all of the competency questions outlined in Section 5.3.1 have been satisfied by the final ontology and each concept used maps quite easily with little to no change.

5.4.3 Comparing the Ontology with a Collection of Documents

Within the domain of scientific observations, Darwin Core is a popular choice for observations of wildlife and plants, it was because of this that we chose to use many of the pre-existing concepts from Darwin Core in K-HAS.

In the data driven approach suggested by [27], a corpus of documents, related to the domain that the ontology covers, and the keyword content is matched with the terms used in the ontology. Because Darwin Core observations are structured into archives of files, explained in Section 5.1.1, we evaluated the K-HAS ontology by combining the approach of comparing a corpus of documents related to the domain with human evaluation of ensuring an ontology met a set of requirements to create a method that ensured existing scientific sensed observations could be mapped to K-HAS with little to no modification.

As explained in Section 4.3.1, an archive of Darwin Core files consists of a minimum of 3 files: a metadata file that contains information about the creator of the archive and the project it relates to (eml.xml), a metadata file that describes all of the files that contain the occurrence and the fields within them (meta.xml) and a csv file that contains the data relating to the occurrence itself. Within the archive, the core files that need to be mapped to K-HAS are the meta.xml and eml.xml files as this allows us to store what and who.

We used files from a DwC archive made available online to extract the terms associated with an occurrence and we mapped these terms to K-HAS concepts, the results can be seen in Table 5.4.

Darwin Core Term	K-HAS Concept
occurrenceID	Occurrence
basisOfRecord	Basis of Record
recordedBy	Person/Node
associatedMedia	Token
eventDate	Event Date
eventTime	Event Time
locationId	Location
scientificName	Individual
identifiedBy	Person
dateIdentified	Identification

Table 5.4: Evaluation of K-HAS against Darwin Core Occurrence Terms

Each term within a Darwin Core observation maps to K-HAS with only a few changes to the terms, while the definitions do not change. As K-HAS is an alignment ontology, there are extra concepts that do not map to Darwin Core, but can encapsulate each occurrence. For example, a project in K-HAS can contain many occurrences. As previously mentioned, [**?**] shows how a DwC archive can be represented in OBOE, which means that it would not be too complex to map an OBOE observation in K-HAS.

5.5 Conclusion

In this chapter we have presented an ontology for the K-HAS architecture and described our methodology for development, as well as showing that existing ontologies are not complete enough to cover our requirements for K-HAS. The K-HAS ontology we present is an alignment of ontologies that are spread across multiple domains and provides a complete solution for a sensor network that deals with scientific observations and we believe this is the first extension ontology that combines sensor-centric and observation-centric ontologies. We have also extended this alignment to include concepts that model the unique features of K-HAS, but this ontology should be suitable for any sensor network that deals with observations.

While the main benefits of this ontology is that it provides a high-level overview of our network architecture and allows those that want to implement K-HAS to re-use all, or some, parts of the ontology, it is also used in the network itself to allow sensed data to be mapped to data stores, define how classifications should be made and determine if certain users have permissions to edit an observation. Using the ontology, we can ensure that all nodes use the same data standard and use a pre-defined user model that can be easily implemented in variations of K-HAS. On top of this, the competency questions, outlined in Table 5.3 also form the basis of queries that can be carried out on sensed data, nodes, users and projects within K-HAS, linking them all through a common model.

Our evaluation has shown that the ontology is logically consistent and that existing scientific observations, in other formats, can be mapped across to K-HAS with few changes. The competency questions we identified during the design phase of the ontology can all be satisfied by the resulting ontology and our K-HAS network currently stores data with a schema that follows the design of this ontology.

In the future, work could be done to make the Darwin Core terms more modular and users will then be able to 'plug in' their occurrence structure of choice. Currently, this ontology has been developed with the scientific observations of our motivating scenario in mind. The ~~modular design allows it to be adapted to suit observations with different structures. The~~ K-HAS, FOAF and SensorML modules can all be reused in almost any WSN but not all networks will be tasked with recording individuals. In these cases, DwC (and some K-HAS) terms would need to be replaced with terms suited to the task of the network, such as flood monitoring.

The tiered architecture of K-HAS means that observations are classified at each step, with DC nodes applying their limited knowledge at the time of capture, DP nodes processing the data in-depth and applying knowledge from previously sensed data and humans either confirming or modifying these classifications. Currently, this only updates the observation as it passes through the network but, using this ontology, we

would like to implement a provenance model that supports the classification of an observation at each stage of the network and edits made by humans once it has reached a DA node. In [**?**], provenance is defined as referring to "the sources of information, such as entities and procesess, involved in producing or delivering an artifact" and the W3C Open Provenance Model (OPM) [**?**] has been developed to "support a digital representation of provenance of any "thing", whether produced by computer systems or not." In future versions of this ontology, and our architecture, we would like to implement this model to support a complete history of edits made to an observation and show a detailed flow of knowledge as it passes through the network.

Chapter 6

Deployment in the Field

In this chapter, we detail the design and deployment of a modified K-HAS in the Malaysian rainforest. We also highlight the issues we experienced deploying the architecture in a humid, dense rainforest for use by those without domain knowledge of WSNs.

Experiments carried out in the rainforest had already shown that the range of wireless communications could be reduced by up to 80% and we expected that the lifetime of nodes in such conditions would be affected, based on experiments explained in Section 3.3.

Yearly visits were made to the Danau Girang Field Centre (DGFC) to test different nodes, gather knowledge and trial iterations of K-HAS. The first visit showed us just how much the rainforest affected communications, but also allowed us to gather knowledge from researchers and cameras that had previously been deployed. Subsequent visits were then used to test our own nodes and software, based on the knowledge we had gained from the first visit, explained in Section 3.3. We developed K-HAS with a view to deploying it in Malaysia, however, it soon became clear that our design was beyond what was currently available, as well as the time constraints.

Developing a camera with both wireless and processing capabilities, as well as being waterproof, proved to a be an extremely difficult task, while wireless wildlife cameras were already commercially available with range much higher than what our experiments had yielded. This could be because of their test environments being places such

as American forest with less dense foliage, lower canopies and much lower humidity and using wireless technologies that are not licensed for worldwide use. For our own node designs, we attempted to use various node types connected to a camera, such as the Raspberry Pi and Waspmote nodes; both yielded problems with mounting an SD card that was readable by both camera and node. At the time, we were unable to create a camera combined with a node that could be left untouched for months at a time in a rainforest.

However, we still needed to implement a network to evaluate our approach, thus we developed a modification to the K-HAS that utilised the latest commercially available hardware to provide an architecture that provides similar capabilities. Because these changes were made due to issues with deployment in Danau Girang, we chose to name it LORIS, after a famous animal in Malaysia. A Slow Loris is a small, nocturnal primate commonly found in South and ~~Southeast~~ South-east Asia but in our context it is an acronym that stands for Local-knowledge Ontology-based Remote-sensing Informatics System.

LORIS has been developed specifically for the Malaysian rainforest, our motivating scenario and, as such, much of the hardware and software has been implemented to reflect this.

The rest of this chapter is structured as follows. Section 6.1 highlights the changes we had to make from K-HAS and explains the hardware used at each tier. Section 6.2 explains how we planned and deployed the network. Section 6.3 details the results from the deployment in Malaysia and Section 6.5 concludes the chapter with a summary of our results.

6.1 Deployment Architecture

The aim of LORIS was to keep as much of the architecture of K-HAS as possible and reuse the ontology without the need for modification; serving as a form of validation.

In order to do this, we identified the tiers of the network that could be deployed in the rainforest, given the hardware and wireless constraints, and the areas that needed to be addressed. Regular power and cheap computers with good knowledge-processing capabilities meant that DA nodes were simple to deploy and, with the growth of powerful microcomputers like the Raspberry Pi, DP nodes were also in abundance. However, finding a reliable camera that could integrate with a node capable of processing observations that was also reliable enough to withstand the humid rainforest proved to be difficult.

6.1.1 Data Collection

Experiments were run yearly in Malaysia to test the performance of variations of hardware. As covered in Section 3.3.1, the first year involved testing the range of Wi-Fi with an IGEP board. The limited range of 30m meant that, despite the high transfer rate, it was not possible. Wi-Fi was not designed for use in resource-constrained sensors, so the power consumption of the radio meant that it limited the lifetime of the node.

The following year, we used the Digimesh protocol, explained in Section 3.3.2, which provided a longer range and was developed for use in sensor networks. While the range was suitable for the rainforest, it was not as much as we had anticipated. The other difficulty proved in mounting an SD card on two devices at the same time, the existing wildlife cameras and our Waspmote nodes, without causing corruption from read/write clashes.

From these experiments, we began to look into commercial alternatives that combined both the node and the camera. The most usable solution we found was the Raspberry Pi coupled with a camera attachment [**?**], but it was not ready for use in the Malaysian rainforest or to be used for an extended unsupervised period. Existing wildlife manufacturers were then looked into and we found that wireless cameras had been manu-

factured, but they had no local processing capabilities and had not been used much in research. Reconyx had created a cellular camera that, in Malaysia, could send images via SMS [**?**], but cellular coverage on the rainforest floor is not common and the cost of the camera was high. Based on these findings, we used Buckeye X7R cameras [**?**][24], shown in figure 6.1. Using the same Digimesh protocol as the Waspmote, they had a tested range of 1 mile and their casing had been developed to withstand harsh environments. The 1 mile range was tested in a more open woodland in the US but the higher quality components used, as well as their modifiable antenna supported our theory that we would achieve better range with their cameras.

Figure 6.1: Buckeye X7R and Battery Pack

We expected to achieve around 800m of range and these proved to be suitable DC node replacements, despite the fact that we had to forgo local processing or the storage of any local knowledge.

6.1.2 Data Processing and Aggregation

Because our DC nodes were wildlife cameras that did not have any knowledge processing capabilities, and because our DC node sites were near enough to the field centre, we combined the DP and DA nodes into one ~~machine~~ desktop stored at DGFC, that contained both a Digimesh radio and an Internet connection. The benefit of using commercial grade hardware was the software that accompanied it; remote management and configuration software allowed users to modify the settings on each camera, as well as handling the retrieval of images from every node deployed. EXIF tags written to images could be modified, as well as how many images were captured for each observation. The software had been created to be used by those without any specialist knowledge and was easily used by researchers in the field centre.

Each observation was saved into a directory that matched the ID of the camera it originated from and this meant that the existing software used in K-HAS could be used without modification. Sensor middleware, such as GSN, is running on the same machine and virtual sensors listen for changes in each directory; where each virtual sensor represents a deployed camera.

When new observations are detected, Drools runs on the images to start metadata and EXIF processing. Some rules had already been extracted from the processing of 120,000 images collected from our yearly visits to DGFC; other rules had to be added by users of the WSN. When an observation had been processed, rules were run once again to determine if a match could be made to existing projects and, if so, who should be notified.

The web interface for classifying and viewing all observations was accessible by those within DGFC, as well as uploading rule files and new locations. Using this interface proved to be easier for those without a WSN background and the GSN interface had a steeper learning curve. Figure 6.2 shows the basic metadata of an observation, outlining where it was captured and when; as well as the images themselves.

Figure 6.2: LORIS Web Interface

6.2 Deployment

In June of 2013, a three week visit was made to Danau Gurang, with three Buckeye cameras and the software required to deploy LORIS. The network was first tested within the field centre and one of the Buckeye cameras had been broken during transport, giving us only two cameras. Due to the protected nature of the forest, we were unable to nail any cables to trees to use the high gain antenna and it was not possible to use the cables without first securing them, because animals tampering with cameras was a common occurrence.

For the initial week of the deployment, we wanted to test the robustness of the network before focussing on the data. This meant placing the cameras in locations where they would be triggered often and in an area where they would be affected by rainfall and humidity. Figure 6.3 shows the locations of the cameras during both of the weeks that it was deployed. Buckeye 1 was deployed for the first week along the main path leading to the field centre from the river, this experienced the most traffic due to the number of

researchers present at the time.

The cameras were initially configured to send a single image for each trigger, as opposed to the three that is used by the current Reconyx cameras at DGFC, because we wanted to test range and data rate for a single transmission. This was fine for images of humans but we found that triggers caused by animals could miss and have no content, we then changed the number of images to 2 so that network traffic was still minimised, especially as images could only be sent was power was available at the field centre.

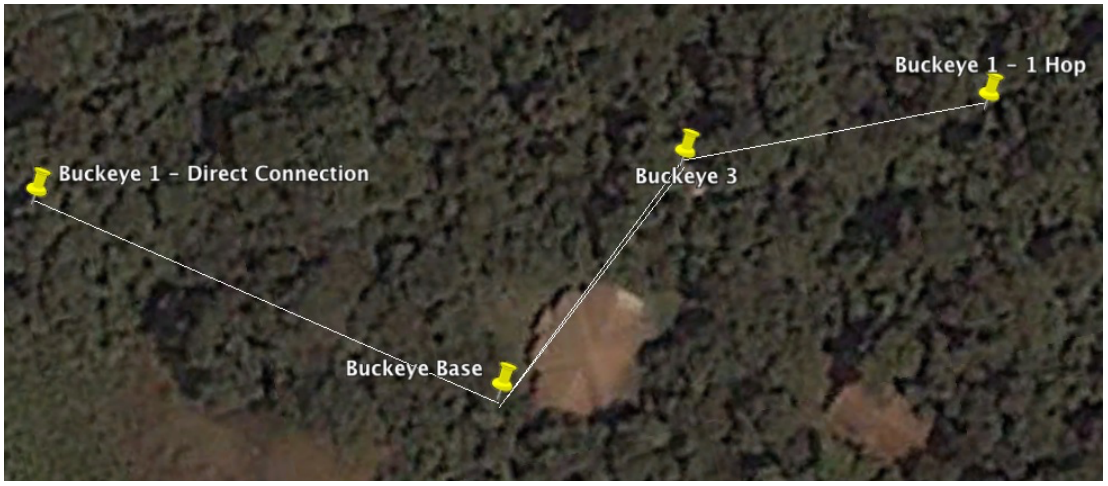


Figure 6.3: Camera Locations Around Danau Girang

6.2.1 Direct Connection

For six days, we tested two Buckeye cameras that were near enough to the field centre to maintain an active connection to the base station. This meant that no hopping was involved and cameras were not reliant on each other to transmit images. Buckeye 1 was placed on a main path that guaranteed human foot traffic and Buckeye 3 was placed on a trail in the forest that, while experiencing minimal foot traffic, was expected to yield small mammals and birds.

For the duration of the six days, the cameras did not report a drop in battery levels and a total of 1076 images were sent. Buckeye 1 was in a more open environment, despite

being further away, and average speeds of 4KB/s were achieved.

Transmissions from Buckeye 3 were significantly hampered by the dense forest that it was surrounded by, as well as the fact that a low-gain antenna was used. We also speculate that the field centre itself acted as a barrier to the signal, as the base station was located on the opposite end. Because of this, we experienced an average speed of 1.2KB/s, taking around three minutes to receive an image. This is consistent with experiments conducted in previous years where dense, humid forest has led to a decreased range of, up to, 78%. Lower frequencies do experience a longer range, but the data rate is significantly impacted.

6.2.2 1-Hop Network

For a further five days, Buckeye 1 was moved further into the forest and Buckeye 3 was set up as a routing camera that forwarded images onto the base station, 63 images were taken during this period.

The location of the moved camera is also shown in Figure 6.3. As was expected, the traffic of the network reduced significantly when the cameras were moved from the main path, with almost all 63 pictures being of animals. Due to animals moving a lot faster than humans, the number of pictures taken per trigger was increased to two, this would ensure network traffic was not overwhelming while increasing the chance of capturing the subject.

Surprisingly, the speed of transmission from Buckeye 1 was faster than Buckeye 3, despite being routed through it, with an average speeds of 1.8KB/s.

6.3 Results

Deploying a network for two weeks is not a robust experiment and does not show that LORIS can withstand months of running without human intervention. However, it

does serve as a proof of concept that, using commercial hardware, a modified K-HAS network can be implemented and utilise the knowledge of its environment.

The pictures of humans from the first six days are not of use for the current motivating scenario, focussed on animals in the rainforest corridor, but one could see this network also being tasked to warn the authorities of hunters in restricted areas of the rainforest. However, the images we have of animals, while few are animals of interest due to the proximity of the cameras to humans, were processed and used to create templates for future observations.

Figure 6.4: First Image of Lizard by Buckeye 3

More interestingly, we were able to infer further rules from the short deployment that helped us to narrow down the choice of animals to match to based on the time of day and location. For example, Figure 6.4 shows a lizard crossing the path of Buckeye 3 in the middle of the day and Figure 6.5 shows the lizard walking past in early afternoon

on a different day. This allows us to encode a window of time in which the lizard is more likely to appear onto the static knowledge base of DC nodes or, in the case of LORIS, onto the DA/DP node. Images an hour later were captured by the camera, showing the lizard walking in the opposite direction. While this was not every day, and the deployment time was too short to establish a pattern, it does help us to infer that a lizard passing by in early afternoon is likely to walk the same path approximately an hour later.

Figure 6.5: Second Image of Lizard by Buckeye 3

6.4 Rules

This section explains the rules used in LORIS and K-HAS in more detail, to highlight how classifications are made as well as how nodes in the network are monitored. Any

examples made here relate to our motivating scenario. The rules on each node are a vital way of encoding local knowledge and a practical use to show how local knowledge, from users or previously sensed data, can be used classify newly sensed data.

As highlighted in Section 4.3.4, K-HAS uses the Drools rule engine, a Java based rule management system that uses forward chaining based inference. Forward chaining starts with some data, in our case a set of images, and uses inference rules to extract more data. We use this method to make meaningful inferences from properties of the sensed data. For example, an image taken at 8pm could be run through Drools and an inference rule will tell the system to start by looking for nocturnal animals.

6.4.1 Data Processing and Data Aggregation

The increased knowledge-processing capabilities allow DP nodes to run a complete implementation of Drools, this means that forward chaining can be used and the knowledge base is dynamic, i.e. it can be updated whilst the network is deployed.

Rules are written in *drl* files that are loaded into a knowledge session. Sessions, as well as the firing of rules, are handled by the Java code. The benefit of using Drools with a Java based middleware, such as GSN, allows the simple integration of the two technologies, allowing a sensor class in GSN to insert an archive into the session and run the rules. When sensed data is received from a DC node, it is unarchived and Drools is run on the DwC files describing the original observation, the metadata, the originating DC node and the knowledge base of the DP node.

Listing 6.1 shows a psuedocode example of some more simple rules that could be contained within the knowledge base, a full listing is in Appendix **??**. In a rules file, existing local knowledge is used with the properties of a new observation to infer the contents and then update the DwC archive, or return suggestions as to what the contents may be. For example, the sample rule uses the location ID of the camera to check if it is near a plantation and uses local knowledge of the area and global knowledge of

the time to infer that the image may contain a sun bear or Malay civet, based on the knowledge that they have been known to forage in the plantation at night when there are no humans. This rule maps to the terms defined in the ontology (Chapter 5) where the individual is a sun bear or a Malay civet, the evidence is an image and the occurrence is the DwC archive created by the triggering of a DC node.

```
rule "Plantation night"
  when New_Sensed_Data
    if location of node is Plantation and time of day is night
      write to file set.CSV
        dateIdentified = \today
        \DIFdelbegin \DIFdel{scientificName = sun bear
        }\DIFdelend identifiedBy = node ID
      Print "Likely a sun bear or Malay civet"
```

Listing 6.1: Pseudocode of a Drools Rules File

These rules are very simple and only scrape the surface of the local knowledge that can be utilised, as well as how DwC archives can be effectively used to carry the data of these inferences, but it does show that Drools is a powerful rule engine and that these rules can be extended further to make full use of the knowledge base.

DA nodes use the same rule engine as DP nodes but rules are used for the post-processing of Darwin Core archives. For example, when a classified archive is received from a DP node, Drools is run to check if the classification matches an existing project. If it does, then it searches for those that are subscribed to the project and notifies them via their selected medium. Drools can also be used to monitor the nodes deployed in the network and check their status. Rules are run to check for archives from DC nodes and, if a node has not sent an archive in a set number of days, then administrators are informed that it may have run out of battery.

More importantly, rules are vital for the exchange of knowledge between nodes in the network. If a user classifies an observation on a DA node, then rules are fired that identify the DP node that sent it and update the knowledge base, on the original DP, so that the image template it uses in future classification matches the one identified by the user.

6.4.2 Existing Data

Throughout our visits to Danau Girang, we collected local knowledge through field work and interviews with the researchers based at Danau Girang. As well as this, we collected classifications made by the researchers for images that had been taken during camera deployment. Using these data, we have extracted patterns in animal movements in order to create some basic rules for the DC and DP nodes. In this section, we will describe the data we have and the methods used to extract rules.

Semi Structured Interviews

On our final visit to Danau Girang, we designed a set of 14 semi structured interview questions that we asked to ten researchers based at the field centre, with the majority working on different projects. We used these questions to gain insight into common trends between projects and patterns recorded about the animal subjects. A subset of the questions have been listed below:

- What species are you looking at?
- Do you have specific sites that you look into?
- What specifics do you know about the target species?
- Do others use your research? If so, how?

These questions have been designed to be as open as possible to allow the interviewees to provide as much detail as they wish, or even transition to a different topic. The interviews were recorded and later transcribed to text. We then used a software package called Dedoose [1], which is an online tool that ~~is used for performing~~ assists with performing manual qualitative analysis on text based documents. Uploading our documents to the service allowed us to manually review the responses to each question and highlight, or 'tag', excerpts of interest and link to other excerpts that had either

been mentioned by another interviewee; or were related to other comments. Each excerpt was tagged with the subject it was discussing, for example, a comment on the accessibility of an area of the forest during certain times of the year were tagged as *local knowledge*. The excerpts were combined and exported into a CSV file that detailed who the interviewee was, the content of the excerpt and the tag. Appendix **??** shows ~~an interview transcript~~ a complete interview transcript with a researcher about the Bornean clouded leopard and Appendix **??** provides an extract of ~~a file~~ an interview with a researcher, about the sun bear, once it has been annotated within Dedoose. The extract included shows an example of questions that were spawned from asking about specific sites and exploring further based on the answers given and show highlighted texts that have been tagged and coloured, by Dedoose, based on the tag.

~~This method allowed us to gain knowledge we would not have learnt from our short time at the field centre, these~~ These face to face sessions gave us insight into patterns and observations that ~~had been made by researchers who have been based~~ researchers had learnt during their months, or years, at Danau Girang~~for many years~~; something we would have otherwise been unable to learn in our short periods at the field centre. An example of this is ~~the global knowledge of the sleeping patterns of clouded leopards. It is widely accepted that they are nocturnal however, the camera trapping project has shown that their sleeping patterns around Danau Girang do not support this and~~ learning that the clouded leopard is not nocturnal around Danau Girang, while it is in the rest of the world. The global knowledge is that clouded leopars are nocturnal animals but they have been seen ~~throughout the day and night. This~~ at all hours of the day around DG. Researchers believe that this could be due to the ~~human impact, availability of prey or even the climate of~~ fact that the rainforest is secondary, growing back after heavy logging 40 years ago, or human impact from palm oil plantations that border the edges of the rainforest or the ~~area, but, if researchers had just used this global knowledge then they would have seen fewer clouded leopards . Similarly, if we had used a rule to only look for clouded leopards at night, then a lot of images would have been misclassified. While it~~ availability of prey. Whilst the local knowledge that

clouded leopards can be seen throughout the day cannot be directly encoded as a ~~local knowledge ruleto constantly look out for clouded leopards, this is an example of local knowledge overriding global knowledge.~~ rule, it does mean that the animal can be removed from rules that may filter out nocturnal animals as possible classifications for images captured during the day.

While we were not able to construct a full set of rules from these findings, the results have provided support for patterns extracted from existing data that has been classified by researchers, which we discuss further in the next section.

Existing Data

When images are manually collected at Danau Girang, researchers process each image set and they have recently been recording the classifications. However, these classifications are only being recorded for currently ongoing projects and other images are ignored. During one of our visits, we collected a CSV file of 2650 confirmed classifications made by researchers. The data had not been cleaned and multiple users had used different names for the same species, we cleaned the data manually and matched it with the images in our database. We then used the classifications for each species to determine any patterns by analysing the details recorded for each observation, which are listed below:

- Time
- Date
- Location
- Temperature
- Moonphase
- Classification

SQL queries provided these details for each species and we recorded patterns that could be identified. For example, if a Samba Deer was only spotted between the hours of 7pm and 4am then we could write a rule that the species is nocturnal. More detailed rules can be created if there are patterns across multiple details. A species that is only spotted in two of the twenty sites between the hours of 7pm and 4am means that nodes with local knowledge can make accurate classifications without the need for image processing. However, these rules are based on large amounts of data collected from an active, or previous, deployment. DC nodes in their early deployment stages would not have the data to make such detailed rules.

Stored procedures were created to join the results from multiple tables and manually explore patterns in the data for the individual species. However, many species either did not have enough observations or the observations they did have had so much variation that there was no discernible pattern. If this work were taken further then we would like to have our entire dataset classified by professionals and Danau Girang is currently undergoing this process for all images taken since the camera trapping project began.

```
if TIME > 1500 and TEMP <= 26:
    if TIME < 0000:
        if MOONPHASE <= 2:
            CLASSIFICATION = GOAT (25\% chance)
        if MOONPHASE > 2:
            if DATE in JAN:
                CLASSIFICATION = HUNTER (2.6\% chance)
```

Listing 6.2: Example Rule created from Existing Data

Using the Weka package ~~, we~~ [**?**], we initially constructed a J48 decision tree ~~and~~ (an open source Java implementation of the C4.5 algorithm [**?**]) but found that the accuracy was only 27% and the rules extracted from the model related to individual times that had only seen a single observation. From this, we then use the Decision Table [**?**] within Weka the model yielded an accuracy of 53%. We used the resulting model to ~~create~~ extract a collection of 281 simple rules that could be run on a DC node. Figure 6.2 shows a rule that was created from the output. This rule checks the time of capture for the observation, the temperature and the moonphase; which has been converted into a numeric value. If the temperature is ~~less~~ fewer than 26 degrees and the time is between 3pm and midnight, then there is a 25% chance of the classification

being a goat. The if-statements are executed in order and the classification that matches the properties of the observation, and has the highest percentage chance, is forwarded to a DP node. The resulting model yielded 281 rules.

With Weka, we experienced the same problem as with SQL, we only had 2700 classifications made by people with domain knowledge, so any models generated from Weka were limited not just to those species in those classifications, but to those species with a sufficient number of observations. Of those 2700 classifications, there were only 45 different species and some had fewer than 10 sightings. For example, the set we used primarily contains goats and the resulting model provided 84% accuracy when given an observation of a goat. However, the Malay badger had very few sightings in the set and the model only yielded 17% accuracy. With future work, we would like to add new classifications to this model and test it extensively with new, unclassified data, as well as add more variables on top of moonphase, temperature, date and time but ensuring they are variables that would be available to data collection nodes.

Because the features used to generate the rules are available in every observation, and do not require any external information, DC nodes are able to process the series of if-statements quickly. This method of knowledge-processing comes at the cost of accuracy, when compared to using existing data, image processing and/or a dynamic knowledge base, but the speed and simplicity of these rules mean that they can be used by almost any node, regardless of computational capability.

6.5 Conclusion

K-HAS was developed, with the motivating scenario in mind, as an ideal, general-purpose network architecture that is able to utilise the knowledge of its environment. However, experiments to support the design and deployment of such a network within the timeframe of this PhD showed that the current technology was not ready to support a network that would run without human intervention for extended periods. LORIS is

our more specialised, cut down approach that can be implemented using commercial hardware that is readily available, as well as software that we have open-sourced.

Our deployment in Malaysia has shown that local knowledge can assist with classifications and can be gained from even a few images collected within the first few days of deployment. Ideally, we would like to leave this network running for an extended period to infer rules from the sensed data and test the reliability of the network in harsh conditions. From both the visit and the deployment, we have offered evidence for the existence of local knowledge and provided a method that can be used to extract it from field experts.

The rules extracted from the interviews and classified data we collected while deploying LORIS can be used as a static knowledge base for DC nodes in the K-HAS architecture, or as the basic starting knowledge base for the DA node in LORIS. As more classifications are made to new sensed data, this same process can be used to create new rules for previously unrecorded classifications, as well as refine existing rules.

The testing of these rules have been drawn from a small set of existing data collected from Danau Girang, although we believe the results are promising enough to show that, in principle, local knowledge could be used to make accurate classifications on sensed data and a larger dataset could generate more specific, more accurate rules in order to do so.

LORIS has shown that local knowledge can be used to enrich sensed data, and automate classification, when it has been received at the field centre, but we maintain that K-HAS' aim of pushing local knowledge right out to the edge of the network makes a network more effective and allows for a more timely delivery of important sensed data. The main drawback of LORIS is that it relies on the chronological delivery mechanism used by the commercial cameras, whereas K-HAS can improve this mechanism by sending the data that it infers to be of a higher importance, rather than what was simply captured first.

Chapter 7

Simulation Experiments and Evaluation of the Architecture

In this chapter, we present the results of simulation experiments developed to evaluate and explore choices offered by our K-HAS architecture. Using nodes with knowledge-processing capabilities to deliver interesting data quicker than a standard WSN, we have developed a simulation for our proposed architecture, K-HAS, as well as variations on the knowledge processing capabilities of the nodes at each tier.

The simulations were developed to determine whether K-HAS is the best mix of processing and collection nodes that maximises network lifetime while minimising the transmission time of interesting data. This was done by using a network structure, that matched our motivating scenario, and changing the knowledge-processing capabilities on each node at every tier; ranging from no knowledge on every node to the maximum knowledge-processing capabilities across the network. We aim to show that the more knowledge-processing capabilities that are pushed out towards the edge of the network, the more effective the network becomes at prioritising data that it believes to be interesting and delaying what it believes to be empty.

This chapter is structured as follows. Section ~~?? describes the implementation of~~ 7.1 describes the tool used to develop the network. Section ~~?? outlines the results and Section 7.4 compares these with LORIS and the current solution in our motivating scenario. Section~~ 7.2 outlines the scenarios used in our experiments and Section 7.3

describes the parameters used to configure each run and what was observed. Finally, Section 7.4 explains the results from multiple runs and Section 7.5 concludes our findings and highlights areas that require further experimentation.

7.1 Simulation Environment

Using RePast Simphony [28], ~~a~~ an agent-based network simulation tool developed in Java, we created a network to emulate K-HAS. RePast is an agent-based modelling system that allows for agents to be created and placed on a grid. Ticks denote a period of time and simulations can run for a fixed number of ticks, or until stopped. Ticks can also be used to schedule events, such as searching for neighbours, by calling methods that last for a set number of ticks, or begin at a particular tick. For example, a camera sensor node may be tasked with taking a picture every three hundred seconds. When the simulation reaches three hundred ticks (or six hundred, nine hundred, twelve hundred and so on), a scheduled event is run to simulate the camera's capture of an image and transmitting it to an endpoint.

RePast was chosen because we did not require the low level network configuration provided by other tools, such as NS2 [**?**], but we did need to modify and record the behaviour between nodes as they capture and process sensed data. RePast's event scheduling allows for nodes to be modelled as agents and the dynamic configuration allowed us to modify the simulations during run time. The aim of these simulations was to visualise how different knowledge-processing capabilities can affect the prioritisation and transmission time of observations, as well as the accuracy of their classifications. RePast allowed us to utilise existing Java code we were using in our K-HAS middleware and develop a base simulation that could be configured easily with XML files.

~~Agents were created, using the RePast SDK, and Java classes were used to manipulate their behaviour. Simple networks may only contain basic agents with only a few~~

~~variations from those provided by RePast. However, for more complex networks, a hierarchy of agents is required and Java's inheritance can then be used to create subclasses of an agent.~~

~~A 2D space is used to display the grid and the simulation is run within RePast's own GUI. This GUI provides functionality such as editing the properties of classes, integrating with Matlab, taking screenshots and saving different configurations of the same network~~Simulations are shown in a GUI that allows the network to be visualised and configured, with results exported to other applications, such as MATLAB and R.

7.2 ~~Experiment Design~~Simulation Scenarios

While the aim of these simulations was to show the effectiveness of K-HAS over the current solution, we also wanted to determine if it was the optimal solution, in terms of delivery of interesting data and network lifetime. We believe that the ideal solution would be to attach nodes with high knowledge-processing capabilities to all cameras in the network, however the short battery life means that replacements would be made as often as the current manual solution, detailed in Section 3.1.

Throughout this chapter, we will be referring to nodes tasked with different purposes as the three definitions listed here:

- Sensing Node: A node that has been tasked with the captuing, and routing, of sensed data.
- Routing Node: A gateway node that is tasked with collecting, processing and forwarding sensed data.
- Central Node: A node with similar functionality to a typical base station, tasked with storing all sensed data and providing an interface to users.

At the routing and sensing tier, the degree of knowledge processing capabilities can range from the levels outlined below:

- No Knowledge (NK): The node possesses no knowledge processing capabilities.
- Minimal knowledge (MK): The node possesses basic knowledge processing capabilities and contains a static rule base (Section 4.2.1).
- High Knowledge (HK): The node possesses high knowledge processing capabilities and is able to process data, metadata and use a dynamic rule base (Section 4.2.2).

The scenarios we have tested show various combinations of knowledge within the network, as well as the combination of knowledge processing at the centre, as a baseline. We can list all the scenarios in as triples, with the knowledge levels described above depicting the knowledge level for each node. The list below shows this:

- NK-NK-NK: Sensing and routing nodes possess no knowledge processing capabilities.
- MK-MK-NK: Sensing and routing nodes possess minimal knowledge processing capabilities.
- NK-MK-NK: Sensing nodes possess no knowledge processing and routing nodes have minimal knowledge.
- MK-HK-NK (K-HAS): Sensing nodes have minimal knowledge and routing nodes possess high knowledge. This scenario matches the K-HAS architecture we proposed in Chapter 4
- HK-HK-NK: Sensing and routing nodes have high processing capabilities.
- NK-NK-HK: Sensing and routing nodes have no knowledge processing capabilities with minimal processing capabilities on the central node.

- NK-NK-HK: Sensing and routing nodes have no knowledge processing capabilities with high processing capabilities on the central node.

The higher knowledge-processing capabilities of HK nodes allow them to classify observations with a greater accuracy, but their battery life is much shorter than MK nodes due to their increased power needs (Section 3.2). In contrast, MK nodes can run for a longer period without requiring battery replacement. MK nodes, however, are unable to classify observations to the same level as HK nodes. While HK nodes can classify an observation as interesting or empty~~and, in our scenario, match to a species,~~ , MK nodes can only assume an observation is interesting using the image's metadata as they lack the capabilities to reliably determine whether an observation is empty or not. The scenarios we have implemented cover combinations of HK, MK and NK, in a twenty five node network. We use twenty five nodes because DGFC had a between twenty and twenty two active cameras during our first and second visits and we knew that the first implementation of the network would require a single endpoint. From this, we chose to use four routing nodes so that each could handle an equal number of sensing nodes, if they were all within range, and a single central node. Our experiments in Danau Girang were restricted to a smaller number of nodes, due to cost, but we expected to deploy a sensing node onto all of the active cameras. The hierarchical nature of the network is shown in Figure 7.1~~and explained later in this section. These scenarios were developed to determine which combination of MK, NK and HK nodes allowed for the greatest network lifetime, as well as the greatest accuracy when delivering interesting sensed data, and they have been outlined below:~~ , with yellow nodes representing sensing nodes, green nodes representing routing nodes and the red node representing the central node.

~~NK-ALL: Sensing and routing nodes possess no knowledge processing capabilities. MK-ALL: Sensing and routing nodes possess minimal knowledge processing capabilities. NK-MK: Sensing nodes possess no knowledge processing and routing nodes have minimal knowledge. MK-HK (K-HAS): Sensing nodes have minimal knowledge and~~

~~routing nodes possess high knowledge. This scenario matches the K-HAS architecture we proposed in Chapter 4 HK-ALL: Sensing and routing nodes have high processing capabilities.~~

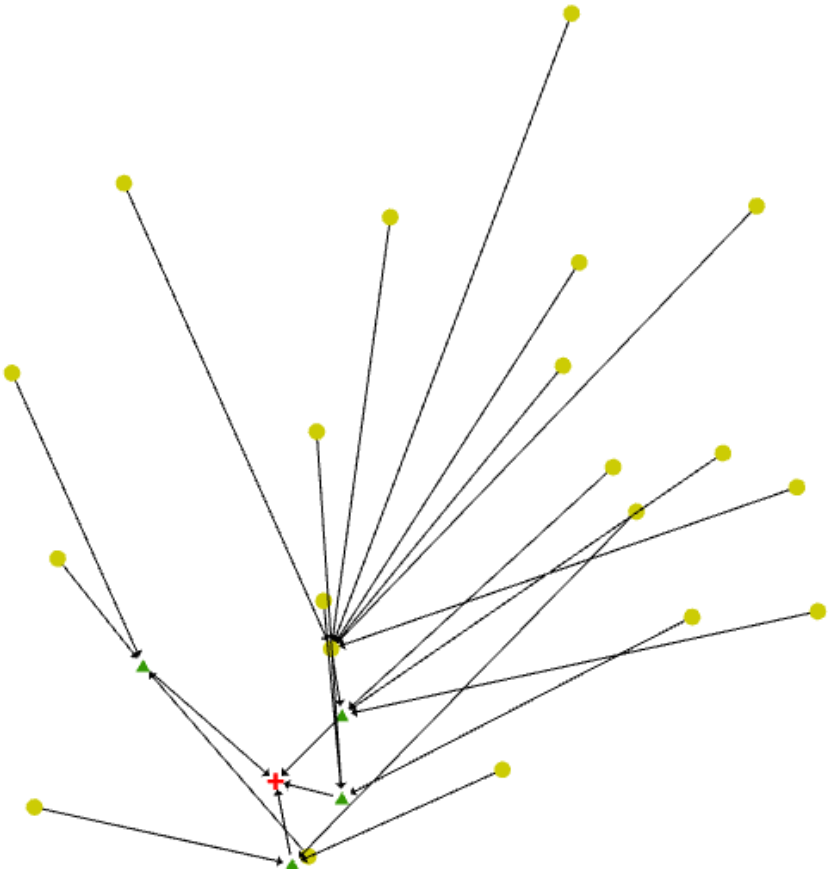


Figure 7.1: Simulation Example in RePast Simulator

Before implementing, we designed the agents required based on the nodes described in the ontology we proposed in Chapter 5. Using that, we created a hierarchy of nodes inheriting common properties from a node object. As previously mentioned, we had metrics on range and transmission times from previous experiments and the deployment of LORIS. We used these to create properties for each transmission medium that could be used by each node object. Table 7.1 shows how the K-HAS terms, introduced in Chapter 4, and LORIS terms, detailed in Chapter 6, map to the node types described above. The nodes used in our simulations map directly to the ontology.

Network Type	Original Term	Maps To
K-HAS	DC Node	Sensing Node + MK
	DP Node	Routing Node + HK
	DA Node	Central Node + HK
LORIS	Buckeye	Sensing Node + NK
	DA and DP	Routing/Central Node + HK

Table 7.1: Mapping of K-HAS to Simulation Terminology

Using a Java library we developed for DwC archives during the implementation of LORIS (Section 6.1), we were able to implement DwC archives as the data standard in our simulations, this allowed us to model the prioritisation of sensed data, as well as the classification of observations, in the same way that ~~an actual WSN~~ a K-HAS deployment would.

~~The structure of the simulation is as follows: The *network builder* instantiates all the nodes, places them randomly on the grid and schedules events once~~

7.3 Network Setup

7.3.1 Parameters

The parameters used in these simulations were either gained through experiments performed at Danau Girang, gleaned from the data or extracted from technical specifications. Repast allows these parameters to be set in a configuration file and can be manipulated once loaded into the simulation. An example of this file can be seen in Appendix **??**. The table below outlines the parameters we have used and some that require further explanation are covered in this section.

Parameter	Description	Value(s)	Constant?	Observed?
Sensing Node Count	The number of sensing nodes in the network	20	Yes	No
Routing Node Count	The number of routing nodes in the network	4	Yes	No
Central Node Count	The number of central nodes in the network	1	Yes	No
Height	The height of the available space	1200	Yes	No
Width	The width of the available space	1200	Yes	No
Capture Chance	The chance of an observation being captured each second in normal circumstances	0.000857703189	Yes	No
Interesting Chance	The chance of an observation containing an item of interest	0.207	Yes	No
Transmission Rate	The transmission rate of sensing nodes in the ~~simulation has started. The nodes then use the properties of their transmission medium to find nodes in range and create a connection; depicted by a line between the node. The simulation uses metrics, such as the size of the image, extracted from the images taken at Danau Girang and the chance of an image being captured by a camera is based on the average capture~~ network	Ideal or Variable	Yes	No
Sensing Node Transmission Rate (Zigbee)	When *Transmission Rate* is set to *ideal*, this is set as the maximum. *Variable* sets this randomly between 20 and the maximum	20-250 kbps	No	No
Sensing Node Transmission Range (Zigbee)	The transmission range of Zigbee	30	Yes	No
Routing Node Transmission Rate	Always set as the maximum due to the proximity to the central node	54000 kbps	Yes	No

Transmission Rate

Each scenario, listed in Section 7.2, can be run with either a *variable* or *ideal* transmission rate. This means that the transmission rate of a ~~camera. The fire rate has been calculated by the average number of pictures captured in a day taken by each camera. Figure 7.1 shows an example topology of the MK-HK scenario , sensing nodes are marked in yellow, routing nodes in green and the central node is red. Black edges are used to link the nodes. Each node is placed randmonly on the space and links are established between neighbouring nodes with the fewest hops to a central node. Network parameters are set in a configuration file, before the simulation runs, such as: the number of each node type and the size of~~ sensing node can be fixed at the maximum rate at the start of each run or a randomly generated number (between 20 and the maximum) is set at the start of each run. The maximum value was taken from the Zigbee technical specifications.

Bandwidth, Capture Chance and Interesting Chance

As with transmission rate, each scenario can be run with the bandwidth under *normal* or *saturated* load. When the bandwidth is normal, the ~~space.~~ chance of capture is the value shown in Table 7.3.1, but saturated increases that chance ten fold. A random number is generated on the sensing node every second and compared with the capture chance, generating an observation if the value is less than the chance.

~~MK-HK Simulation Example in RePast Simulator~~ Using the existing data collected from Danau Girang, we calculated how often a camera triggers in a six month deployment, as well as how often the observation contained interesting content.

To calculate the count of interesting images, we processed every directory of images to extract the largest object in the foreground, using our Triton program. Once processed, we iterated through every directory, counted the total number of images and the total number of extracted images; an extracted image is a black and white image containing

the largest object that has been found in the observation and are only created when the processing believes that the observation is interesting. This gave us a 20.7% chance of an image being interesting, across every camera.

7.3.2 ~~Darwin Core~~

~~The Darwin Core class represents a DwC archive,encapsulating *Identification*,*Location*,*Occurrence*, *Image* and *Species* (Section 5.3.2). The images we have collected~~ The chance of a camera being triggered each second was calculated by the total number of observations (13,399) divided by the number of seconds in six months (15,552,000). This gives a chance of 0.000861561 of a camera trigger in any given second.

Transmission Range

From our experiments at Danau Girang, see Section 3.3, we tested the range of Wi-Fi and Zigbee in a humid rainforest, using those values we set trhe transmission range as the maximum that we achieved during our experiments.

Observation Size

Using the observations we extracted from Danau Girang~~were processed to find details such as the average size when captured at night and day,how often an average camera triggers and the percentage of images with animal content. These data were then used to specify how often a randomly placed node should capture an observation per tick.~~

~~Upon each capture, images are created and given a random size ,~~ , we listed the size of all 120,000 files and extracted the minimum and maximum size from the set, when an observation is generated in RePast, it consists of three images and each image is a randomly generated size between the maximum and minimum ~~size found in the 120,000 images collected from DG. The sum of the image sizes is used to calculate the size~~

~~of the archive. Using this size, a sensing node calculates how long the archive takes to send based on the size and the transmission rate. We assume that the rate stays constant for the duration of transmission. When an archive is sent to the routing node, we used the average time for our image processing tool and Drools engine to run and attempt a classification, which is 43 seconds (ticks), explained in detail in Section 3.4.1. To keep the classifications as general as possible, so that the simulation applies to any WSN for scientific observations, archives are not classified down to the species level, they are marked as~~ bounds outlined in the table.

Processing Times

While the processing times are fixed, central nodes can perform higher knowledge processing slightly faster and they are able to process up to 4 observations at once, whereas routing and sensing nodes can only process one at a time.

Run Count and Duration

Each scenario is configured to run for a period of three months, this is solely due to time constraints due the run time of each simulation. Tests showed that three months was a long enough period to model a saturated network and the capture chance is kept at the six month chance because the network would be deployed for six month periods and these runs serve to show a snippet of their deployment, rather than cover the whole period.

Twenty five runs were completed of each scenario, as well as for the possible combinations of transmission rate and bandwidth. For example, a network simulating NK-NK-NK would have 25 runs for *ideal* transmission rate and *normal* saturation, 25 for ~~*interesting*~~*ideal* ~~or *empty* and then forwarded to the central node~~transmission rate and *saturated* bandwidth and so on.

7.3.2 Routing Protocol

The routing protocol used needs to be dynamic in order to adapt to nodes being added and removed during deployment, while minimising traffic in a resource constrained network. In our approach, we use a modification of the Minimum Cost Forwarding Algorithm (MCFA), described in Section 4.3.5. A cost is assigned to each node, based on how far they are from the central node, with neighbouring nodes choosing to connect to the node with the lowest cost. However, in normal implementations of MCFA, all nodes are of the same type and simply need to connect to a ~~base station~~central node. This protocol is used in all scenarios.

In our K-HAS architecture, sensing nodes cannot connect directly to a central node because processing would not take place. Because of this, we used the same routing method across all scenarios. Our implementation of MCFA works with a discovery phase and a transmission phase. The discovery phase is a scheduled event, taking place at the start of deployment but it can be run throughout deployment to react to nodes being added or removed.

Discovery

Discovery begins at each central node, scanning nodes in range for routing nodes and sending a broadcast packet, with a cost of 0, to inform them that they are within range of a central node. Links between Central and Routing nodes use W-Fi in all of our scenarios. Once received, routing nodes increment the count and forward the packet to any routing nodes within range of them, where we use the range of Zigbee. We found that this method overloaded the routing nodes and all sensing nodes within range would connect to the first routing node they receive the broadcast from. We then implemented a method, called *load balancing* [**?**], which uses the sensing nodes connected to a routing node to calculate whether it should offload new nodes to a neighbouring routing node.

The maximum connections a routing node can have is determined by the total number of sensing nodes in the network divided by the total number of routing nodes, which is held in the knowledge base of the central node. Once a routing node has the maximum number of connections allowed, it starts to offload to a neighbouring routing node that is also in range of the sensing node requesting a connection. If there are no neighbouring nodes then the routing node exceeds the maximum number of connections allowed, to save sensing nodes being left with nowhere to send their data.

If the sensing node that receives the broadcast does not have an existing route to a central node, or the cost of the current route is higher than the received route, it adds an edge to the routing node, increments the count and forwards it to all nodes in range. This process continues until the broadcast reaches the edge of the network. Nodes do not have global knowledge of the route to the central node, only of their neighbour with the lowest cost.

This phase can be repeated throughout the course of the deployment, simply by scheduling it as an event to occur every n ticks. However, the simulation currently only uses the discovery phase at the beginning of the deployment.

Transmission

Once the discovery phase has been completed, providing nodes are within range of the central node, the transmission phase begins where only DwC archives are then sent across the network. Observations are captured based on the mode of the simulation and sent to the lowest cost neighbour.

In order to manage transmissions, ~~sensing~~ nodes have a *SendState* object that contains the next archive to send, the time it will take to send it and whether it is currently sending. This is used to determine what operations to perform~~, once~~. Once an archive has been sent, it is deleted from the SendState and the sending flag is set to false. A new archive is then added and sent when the opportunity arises.

When a routing node receives the archive, it begins processing. Routing nodes use the SendState as well, but they only add an archive once it has been processed and they then select the oldest archive that has been classified as interesting, providing an archive is not already waiting to be sent. The archive stores information about the route it takes, recording every hop, as well as the time it took from capture to central node.

Scheduled sending events run every thousand ticks, which is configurable, to check the sending state of the node and send any archives in the SendState. The node then waits for the number of ticks that it will take in order to transmit the archive.

Once the simulation is completed, either manually or through a defined number of ticks, the archives in each ~~central~~ node are iterated over and written to a CSV file, with details such as the path it took, total transmission time and time of capture.

7.3.3 ~~Capture~~Observation Transmission

~~Using the existing data collected from Danau Girang, we calculated how often a camera triggers in a six month deployment, as well as how often the observation contained interesting content.~~

~~To calculate the count of interesting images, we processed every directory of images to extract the largest object in the foreground, using our Triton program. Once processed, we iterated through every directory, counted the total number of images and the total number of extracted images. This gave us a 20.7% chance of an image being interesting, across every camera.~~

~~The chance of a camera being triggered each second was calculated by the total number of observations (13,399) divided by the number of seconds in six months (15,552,000). This gives a chance of 0.000861561 of a camera trigger in any given second.~~

7.3.4 ~~Processing~~

The types of knowledge processing capabilities that we outlined in Section ?? are used in the simulation to determine which type of processing to perform on observations. The result of processing is that an observation is marked as interesting or empty. The limitation of our image processing tool is that only the largest region of interest (ROI) is extracted, even if there are multiple objects in the image. The outcome can be any of the following: True positive (TP): An ROI is extracted that contains the animal in the set. False positive (FN): An ROI is extracted that contains nothing of interest. True negative (TN): A camera is triggered with nothing of interest in the image and no ROI is extracted. False negative (FN): An image containing an animal has no ROI extracted.

Using the results of our image processing application (explained in Section 3.4.1) for the properties of HK processing, we encoded that an 82% accuracy at detecting TP images, with a 98% accuracy for finding TN images. Nodes with MK do not have the ability to mark an image as empty, but they can mark an image as interesting. However, the results we have from our rule base are not as extensive as the results we have for Triton (due to limited local knowledge and existing rules approved by domain experts), so instead we use a predefined 10% accuracy for detecting TPs. This 10% is used because we needed to show the difference between a node with MK and a node with HK. HK nodes have the ability to process the contents and metadata of sensed data, with access to libraries and a dynamic knowledge base. MK nodes are able to look at the basic metadata and only have a static knowledge base, limited in size. In future work, we could test with greater accuracy or, with more rules, we could run experiments to find the exact accuracy of an MK node. An interesting image is an image that contains an animal and can be either a TP or FN. Empty images are those that do not contain an animal and can be either FP or TN.

7.4 Results

~~In this section, we explains the results from simulating each scenario in a randomly generated network graph.~~

~~Each scenario, outlined in Section **??**, runs to simulate a 6 month deployment. Using our motivating scenario, we modelled each scenario on a fixed number of nodes: 1 central node , 4 routing nodes and 20 sensing nodes. The implementation of our architecture has been limited to using Zigbee as the transmission medium between all sensing nodes, with a Wi-Fi connection between routing nodes and a central node. Sensing nodes then~~ When a node has no knowledge, it simply sends the observation that was first added to its queue. With MK, it finds the first *interesting* observation and sends that, if one is not in the queue then an observation marked as *unknown*. HK performs much the same as MK except it instead looks for *empty* observation if an *interesting* cannot be found. If a node has any form of knowledge processing power, it will not send an observation ~~when it is captured or, if it has not captured anything then it checks for a backlog every ten minutes. Sensing nodes check for new observations to process every five minutes. We ran each scenario a hundred times, with each run simulating a 6 month duration. The time to process an observation using MK has been simulated to take 5 seconds compared to the 90 seconds when a node has HK capabilities; these values have been chosen based on the average processing time we derived using existing data.~~

~~Percentage of Interesting Observations for All Scenarios.~~

~~The simulations were run in two modes: ideal transmission rate and variable transmission. Ideal used a fixed transmission rate (250kbps) for Zigbee links between all sensing nodes, with all observations being sent at the highest possible rate. Variable generates a random number for the transmission rate (between 20~~ unless it has first been processed. Nodes can only send an observation one at a time and ~~250kbps) for each sensing node link, at the time of the network's initialisation. This models the large fluctuations in both speed and range that we observed in the Malaysian rainforest when we ran tests with Zigbee equipment. The Wi-Fi range remains constant because the distance~~

of routing nodes from central nodes, in our motivating scenario, can be made small enough that a consistently good connection is achievablemust also wait for the receiving node to be free.

Ideal Transmission Rates. Variable Transmission Rate. Mean Duration of All Observations

Figure **??** shows the percentage of interesting observations that arrive at the central node. Due to the short transmission time, almost all observations arrive in near real-time (fewer than 2 minutes) and, as such, the split between empty and interesting remains almost constant at approximately 80:20. This is important because the chance of an image being interesting is 20.7%, so all data is being received at the DA node, without any observations backed up on nodes in the network. However, the True Positive observations are the most important as they are the interesting observations that have been marked as such. False Positives are interesting observations that have been wrongly marked as empty. Increasing the knowledge processing capabilities of routing nodes, shown by the difference between MK-ALL and MK-HK, increases the number of TPs, as well as reducing the number of FPs. Another important point to note is that MK-HK and HK-ALL provide the same results, despite HK-ALL solely consisting of nodes with the highest knowledge processing capabilities. This is probably due to the fact that all observations, in MK-HK, go through an HK routing node and, thus, would receive the same level of processing as in an HK-ALL scenario.

These results support our hypothesis that MK-HK is the best scenario to choose in terms of timeliness of data delivery, quality of data delivered and network lifetime, as MK-HK has all sensing nodes with minimal knowledge processing capabilities, allowing them to run for approximately three months, achieved from tests detailed in Section 3.2.4. However, with HK-ALL, all sensing nodes run with higher capabilities and are limited to run for a maximum of three weeks before battery replacement is required. While this number is similar to the current, manual process described in Section 3.1, this does include processing and prioritisation of sensed data, which can take weeks or, sometimes, months when carried out by humans.

7.3.1 Duration

The ideal transmission rate runs with the transmission rates of all nodes at their theoretical maximum, giving the highest speed for both Zigbee and Wi-Fi. Figure ?? shows the mean duration of any observation, interesting or empty, in each scenario. With the exception of NK-ALL, it is clear scenarios that, where more nodes have higher knowledge processing capabilities, the mean duration is increased. An explanation for NK-ALL nodes showing a higher average duration is because the lack of knowledge on any of the nodes prevents it from prioritising observations, which means that some could be queued for longer before being sent on. One of the key points to note is that the difference in transmission time from capture to central node is typically no more than one hundred seconds. When compared with a standard, power efficient WSN with no knowledge processing capabilities, the reduced battery life may not be worth the trade off. However, these times are not solely for transmission, they also include in-network processing and prioritisation when they arrive at the central node.

Total Interesting % Interesting Empty % Empty NK-ALL 44205 9133 20.7 35071 79.3 NK-MK 44244 9136 20.6 35107 79.4 MK-ALL 44052 9132 20.7 34920 79.3 MK-HK 43754 10066 23.0 33688 77.0 HKALL 42511 9671 22.7 32840 18.9 Mean Number of Observations for All Scenarios under Ideal Transmission Rate

During our visits to Danau Girang, we performed experiments for many different transmission methods: Wi-Fi, Zigbee and RF. The results were always different to what we experienced in the UK and the variation in connectivity in just a few minutes, or metres, was extreme in some cases. Days where the humidity was one percent higher than the previous day could result in a 50% range drop and rain for a few minutes could drop all connections for an hour afterwards. Modelling this in our simulations was necessary to accurately plan for what we could expect in Malaysia. Our planned network topology (Section 4.4.1) in Danau Girang has routing and central nodes so close to each other that the Wi-Fi connection remained fairly stable and connectivity should remain at 100%. When simulating sensing nodes, we randomly

set their transmission rate between 20 and 250kbps at the start of each run. As each scenario is being run 100 times, this method is more efficient while still allowing for variation in the transmission rates.

The number of interesting observations received in each scenario remained similar to the ideal mode (see Tables **??** and **??**), as expected, but the fluctuation in transmission time (Figure **??**) is much larger, with some images taking 800 seconds, or 13 minutes. In our motivating scenario, this is not much of an issue. However, if the image was of a hunter or the scenario was building intrusion detection system, then thirteen minutes is a long delay that could cause considerable damage. Of course, using all Wi-Fi connections is a more feasible option, but situations where long range is required prevents this. Our simulations do not currently take into account broken links in the network, where changes to the node's environment (such as a fallen tree) could cause it to drop out of range for a few minutes, or even hours, but this can be addressed by implementing the discovery phase (described in Section 7.3.2) of the routing protocol to run at a set interval throughout the duration of the simulation.

The duration of an observation with both a variable and ideal transmission rate does not fluctuate largely and Figure **??** shows that the processing power of MK-HK vs. HK-ALL is identical. However, Figures **??** and **??** show that the primary benefit of pushing HK processing capabilities out to the edge of the network allows for better prioritisation of observations. Sensing nodes with HK are able to determine whether an observation is interesting or empty with much greater accuracy than MK and these capabilities, right at the edge of the network, allows for an interesting observation to be processed as soon as it is captured and sent without further processing directly to a central node. The only delays being the processing time itself and the sending queue of intermediate nodes. Figures **??** and **??** show that, when a network is not saturated, interesting and empty observations take a similar amount of time to pass through the network; with the exception of the HK-ALL scenario. WSNs with much worse transmission rates could experience a much larger time difference between interesting

observations, especially when using long range FM frequencies that can transfer only a few bytes a second. Interesting Observations.

Empty Observations. Mean Duration under Variable Transmission Rate

Total Interesting % Interesting Empty % Empty NK-ALL 43743 9012 20.6 34730 79.4 NK-MK 44258 9195 20.8 35063 79.2 MK-ALL 44152 9130 20.7 35021 79.3 MK-HK 44273 10183 23.0 34059 77.0 HKALL 43011 9935 23.1 33076 18.9 Mean Number of Observations for All Scenarios under Variable Transmission Rate

7.3.1 Network Saturation

With the transmission rates of Zigbee and Wi-Fi, an observation of three images, each under a megabyte in size, can be processed and sent in a matter of minutes; when there are only a few hops. However, with these network topologies, we are unable to see how each scenario handles prioritisation when it is unable to send every observation captured. This can be done by either reducing the transmission rate or increasing the chance of an observation being captured. We chose to increase the chance of an observation being captured and maintaining the transmission rate in order to keep the simulations in line with our motivating scenario. Existing data from Danau Girang has shown a seasonality change in the number of images captured and we know that some sites are more active than others, therefore making a higher chance of capture more fitting.

Imagine a scenario where flash floods in the rainforest prevent access to the nodes to change the battery and a mating season of sun bear has caused the capture rate to increase. The chance of an image capture has increased and users of the network have programmed the sensing nodes's sleep duration to a minute; saturating the network with more images than can be sent in a six month period.

Total Interesting % Interesting Empty % Empty NKALL 254352 127149 50 127149 50 NKMK 264580 144058 54.4 120522 45.6 MKALL 183727 126530 68.9 57196 31.1

KHAS 159446 115054 72.2 44392 27.8 HKALL 192010 155656 81 36353 18.9 Mean Number of Observations for All Scenarios when Saturated

We simulated this across all five scenarios and the results are detailed in this section. The duration is increased for all observations (Figure **??**) and there is an increase in the time it takes an interesting observations when compared to an empty observation. Table **??** shows the average number of observation captured across all scenarios, however, when you look at the number delivered, we can see that the number of interesting observations delivered is much greater. This means that the network is successfully prioritising observations it believes to be interesting and not sending those that are empty because of the time and bandwidth constraints.

Mean Duration of All Observations Across All Scenarios

Figure **??** shows that the mean transmission time of all observation does not differ much between scenarios (with all scenarios at approximately 400 seconds) when the network is saturated, but we can see that the network must decide what images to send or drop when there are more observations than bandwidth available. HK-ALL is the slowest of these because of the increased time it takes to process and image with HK. This is shown more clearly when we compare *interesting* and *empty* observations. The red line (showing false positives) only exists for MK-HK and HK-ALL scenarios because MK nodes are not able to classify an observation as empty and, as such, NK-ALLAn observation is marked as delivered once it has been received by the central node, NK-MK and MK-ALL do not generate any false positives.

Empty.

Interesting. Mean Duration of Observations across All Scenarios

The mean transmission time of empty images (Figure **??**) shows that the increase of knowledge processing capabilities improves the network's ability to filter out empty observations and a network with only HK nodes is able to defer empty observations from the time of capture so that bandwidth remains free for those that are interesting.

~~Figure ?? shows the mean transmission time of interesting observations and we can see that interesting observations misclassified as empty bring the average duration for interesting observations up to approximately 400 seconds but the ability to prioritise allows TP observations to be delivered at a much faster rate of between 300 and 400 seconds, compared to more than 800 seconds for empty observations. It is worth nothing that, while misclassified interesting observations (FP) take longer to reach~~ if there is only processing at the central node~~in higher knowledge scenarios, the number of them received is greatly reduced from between 30% and 45% in MK scenarios to less than 20% for HK scenarios; as shown in Figure ??.~~

~~Percentage of Interesting Observations across All Scenarios~~

~~Percentage of Interesting Compared to Empty Observations across All Scenarios~~

~~Saturating each scenario shows how different levels of knowledge processing across the the nodes can prioritise the delivery of interesting observations while filtering out observations that have been classified as empty. We can see that the HK-ALL scenario delivers the highest percentage of interesting observations (80%), with a slightly higher average transmission time. Having HK on all nodes causes a large reduction in the lifetime of the network to 3 weeks. L-KHK (or KHAS) is a good alternative to this as it provides a network lifetime of 3 months for the majority of the nodes but still around 70% of all images (same as MK-ALL) are interesting, but the majority of these are *true positives*. In a network with limited access to a power source, MK-HK is the best implementation but, for networks with access to a continuous power source, HK-ALL can be used to increase the number of interesting observations delivered to the central node in a shorter time period~~, then delivery is delayed until the central node has processed it.

7.4 Results

7.4.1 In-Network Processing

7.4.2 Central Processing

7.5 Conclusion

In this chapter, we have detailed the development of simulations to show the different scenarios for pushing knowledge out to the edge of a network. Using different levels of knowledge processing capabilities on nodes, we have shown that a network with HK processing capabilities can detect and prioritise interesting images, while simultaneously delaying empty images for a time when the network is not busy. However, using a network that solely comprises of HK nodes results in a battery life that lasts for 3 weeks on each node. The MK-HK network architecture we have proposed provides a combination of HK and MK nodes, distributed based on their role in the network. For example, nodes tasked with sensing and forwarding images only have MK processing capabilities. These simulations have shown that the HK-MK scenario results in a delay of interesting image delivery, when compared to MK-ALL, but the percentage of interesting images delivered is significantly increased.

Using a model of real world transmission rates, we have seen that the variation in transmission times can be quite large, but the difference between MK-HK and HK-ALL is not that great. This suggests that our MK-HK proposal (K-HAS) is the best solution for most scenarios as it provides a longer network lifetime with same ratio of TP:FN images as a network where every node is equipped with HK.

While the simulation is not feature complete, it is accurate enough to show how MK-HK utilises the knowledge-processing capabilities at each tier to process, and prioritise, sensed data based on knowledge gained from the environment, previously sensed data

and from humans using the network. Our results also show that the difference between MK-HK and HK-ALL is not significant enough to warrant the loss in network lifetime. However, not every scenario would be suited to this. NK-MK shows a slightly faster transmission time and a much longer network lifetime, as most nodes would not have any processing power. While the processing of sensed data would not be enough to rely on, it could act as pre-processing that could be processed further once power is not an issue. This would be well suited for networks where nodes have limited power and are not easily accessible, such as bird nest monitoring networks.

MK-HK allows sensed data to be processed, and delivered, in near real-time with approximately 87% of all interesting data being correctly classified. This number could increase the longer that the network is deployed, which is something we would like to investigate in the future.

Finally, we simulated the situation where there is more sensed data than the bandwidth available. The greater knowledge-processing capabilities of HK-ALL scenarios ensured that empty images were delayed and interesting observations were sent with a greater priority. While there were interesting observations misclassified as empty that were delayed, they made up less than 10% of all interesting observations. MK-HK did receive fewer interesting observations, most likely due to its limited ability to prioritise sensed data right from the edge of the network, but the majority of interesting observations were true positives and they were prioritised over empty observations. MK-ALL, with a longer network lifetime of MK-HK and HK-ALL, did not prioritise interesting over empty data effectively but still provided more true positives than false negatives.

We can conclude that, for a power efficient WSN, the results from these simulations suggest that MK-HK can deliver many of the same benefits that HK-ALL provides while not pushing HK to the edge of the network, reducing the network lifetime. If power was of no concern, HK-ALL is able to deliver more interesting observations and prioritise more accurately from the edge of the network, as well as filtering important data more effectively when the network is saturated.

Chapter 8

Conclusion and Future Work

In this thesis, we aimed to show that utilising the local knowledge of an environment in a WSN improves the efficiency of the network by giving it the ability to prioritise sensed data based on the results of in-network processing. We believe that pushing knowledge out farther towards the edges of the network improves the overall performance. To show this, we have developed a three-tier WSN architecture that uses knowledge-processing capabilities to process sensed data as it is forwarded through the network. Most current WSN implementations deliver data chronologically, or store it on the node to be retrieved by queries. We believe that this knowledge can be used to infer how valuable sensed data is and prioritise that through the network, delivering the most interesting data first. However, resources are still limited in WSNs and our architecture had to utilise these resources effectively, such as battery life and bandwidth, to maximise the network lifetime. Using Data Collection (DC) nodes at the edge of the network, they capture observations and use their limited knowledge-processing capabilities to enrich the sensed data before sending it on. Data Processing (DP) nodes use more powerful knowledge-processing features to attempt to classify observations and prioritise the sending of them to Data Aggregation (DA) nodes. DA node make the data available to users and use their classifications, and input, to dynamically update its knowledge base.

We used a collaboration with a research centre in Malaysia with the aim to implement K-HAS in the Malaysian rainforest, in an area that had experienced logging and contained a diverse range of rare wildlife. Using K-HAS, we wanted to deploy a network

that would prioritise images of rare wildlife and only send images of common wildlife when bandwidth was available. Over the course of 3 visits, we gathered knowledge from the area, researchers and locals to build a knowledge base and create rules that could be used to classify data. We also collected images taken from their current, manual solution to infer patterns and use existing classifications for new sensed data.

Current sensing technology means that K-HAS is not ready to be implemented as the architecture dictates. DC nodes are not ~~a type of sensor that is~~ yet readily available, and this is especially true for image capturing sensor nodes. Because of this, we modified the architecture to use commercial hardware with fewer capabilities that would allow us to actually deploy a sensor network in the Malaysian rainforest that does use local knowledge. We call this architecture LORIS, Local-knowledge Ontology-based Remote-Sensing Informatics System, and this solution combines the DA and DP node to hold all of the knowledge-processing capabilities.

LORIS has shown that even using local knowledge at the base station of a WSN means that sensed data is processed and organised within minutes of being received. This method also means that users are alerted to data that they have subscribed automatically.

Our simulations of K-HAS show that the bandwidth in a WSN is used more effectively when knowledge is pushed out towards the edge of the network, allowing nodes to perform knowledge-processing to make inferences about the contents of the data and prioritise, or delay, its delivery appropriately. We have no reason to think that this would not work in practice.

8.1 Summary of Contributions

In this section, we summarise the contributions detailed in this thesis, focussing on our deployment in Malaysia, the tiered architecture designed for general use and the align-

ment ontology created to formalise the architecture of K-HAS and the data standard that it uses.

8.1.1 K-HAS

In Chapter 4, we present a novel tiered architecture, K-HAS, for WSNs that uses local knowledge. We explored existing networks, those related to our motivating scenario in Malaysia and also those that use knowledge or context-awareness, and routing protocols that use the sensed data to determine how it is routed. We explain the purpose of each tier in the network and show how sensed data is enriched and routed as it progresses through the network. Using Darwin Core as a data standard, each node communicates in a common format and metadata is packaged with data in archives that can be read by any node in the network. We also show how rules can be used to infer the content of sensed data and the ability to run rule engines in the network allows sensed data to be prioritised based on its value; not just the time it was captured.

8.1.2 Ontology

In Chapter 5 we explain the aligning ontology created to formally represent the K-HAS architecture and the data standard used. We show how~~no ontology current joins~~ , while there are existing ontologies that join observation-centric and sensor-centric ontologies~~to completely represent the hardware used in a WSN as well as the sensed data~~, K-HAS allows for the representation of knowledge exchange in the network and show nodes performing tasks similar to humans. We also show how the ontology is extensible and ~~in no way~~ does not need to be specific to K-HAS; it can be used with any WSN that deals with scientific observations. ~~The modular nature of the ontology means that parts of the ontology can be extracted and re-used in a network that the entire ontology may not be suitable for.~~

8.1.3 LORIS

In Chapter 6, we present the Local-knowledge Ontology-based Remote-Sensing Informatics System (LORIS), a system developed for our motivating scenario when we discovered that current sensing technology means that K-HAS is not ready to be implemented in its current form. DC nodes are not a type of sensor that is readily available, and this is especially true for image capturing sensor nodes. Because of this, we modified the architecture to use commercial hardware with fewer capabilities that would allow us to deploy a sensor network in the Malaysian rainforest that does use local knowledge. This solution combines the DA and DP node to hold all of the knowledge-processing capabilities and then uses commercially available sensors to replace DC nodes.

This architecture is easier to implement but does not provide in-network processing, although it automates the delivery of sensed data and uses the increased knowledge-processing capabilities of modern PCs to process sensed data on arrival and inform users. We show how our deployment of LORIS was successful and highlighted how sensed data was delivered within minutes of being captured in some cases, and processed shortly after.

8.1.4 Simulations

In Chapter 7, we explain the implementation and results of our simulations to model an ideal deployment of K-HAS. We model every variable of the network on existing data collected from our motivating scenario and show that the delivery of observations can be reduced by more than four hundred hours, when compared to the current manual solution. We also outline how the network is able to prioritise data that it believes to be interesting, using a priority queue mechanism that delays data it believes to be empty. Comparing K-HAS to different network implementations, where nodes have different levels of knowledge processing capabilities, shows that our network implementation is

~~more power efficient than~~ similar in performance to a network where every node has higher knowledge processing capabilities~~, yet its performance is similar~~.

8.2 Future Work

The focus of this thesis is to show that local knowledge can improve the timeliness of interesting data by making inferences based on previously sensed data and knowledge of the environment. We have shown, using simulations, that this can be done with our tiered approach. However, another aim was to deploy such an architecture for our collaborators. We have explained that current technology means that this is not feasible, but sensing, and microcomputer technology has moved forward significantly since the beginning of this PhD and it would be possible with a longer time period. Our deployment in Malaysia was limited due to time constraints and we would haved liked to leave a functional deployment active in Danau Girang for a six month period.

With more time, we could create custom DC nodes using webcams and micro-computers, like the Raspberry Pi, encased in a watertight enclosure; allowing us to use higher levels of knowledge-processing at the edge of the network. A deployment of this length would also allow nodes to act on sensed data classified by users and update their knowledge base, responding to changes in the network dynamically.

K-HAS uses a combination of open source projects and some software created during the course of this work, but it does require specialist knowledge to be deployed. We would like to create an installation candidate that would be usable by those without any expertise that could provide basic information on the purpose of the network and the installation of all necessary packages would be automated.

On top of this, our software's user interfaces have been tested by researchers at Danau Girang but we have no metrics on the usability of the software. Testing the software on users to determine how they would score the different areas of the software, such as usability, response time and learning curve, would help us to improve the software and

ensure that it can be used by those without any technical knowledge of K-HAS or its underlying architecture.

Experiments have shown us that the range of wireless transmissions is often hard to predict and can be heavily influenced by weather and obstacles. If sensors were placed at the edge of range for a neighbouring sensor, then it is not guaranteed that transmissions could be made every time. We believe that using humans as an intermediate hop could be an interesting research opportunity. Using a mobile app to transfer data between nodes as they are within walking range could speed up data transfers and their knowledge could be used to prioritise data without the need to process it.

If a human views a recent observation when they are in range of a node, they could make an instant assessment on whether it is interesting or not. If they mark it as such, then it could be passed through the network with a higher priority and reach a DA node within a short period as it would not require processing, updating knowledge bases for future, similar observations as it is forwarded.

One important goal is to create a deployment of K-HAS for a different purpose than our motivating scenario in Malaysia. There is a need in Malaysia to track hunters in the forest and alert authorities, this would not require any change in how K-HAS is currently implemented but would show how it can dynamically adapt based to changes in its sensing requirements. We would also like to test K-HAS in a situation where power is no longer a limiting factor, but delivery time of sensed data would be. A building security network would be one such example, deploying K-HAS across a number of floors and processing video feeds to alert users within seconds about suspicious activity.

However, using K-HAS for a WSN that, for example, uses text based sensed data to monitor the temperature and lava level of a volcano is a completely different implementation, but the local knowledge could be used to prevent an emergency and predict eruptions. A deployment such as this would show that the benefit of using local knowledge in a WSN is not limited to our motivating scenario, but is versatile enough to

benefit almost any network.

Using heterogenous sensor nodes within a K-HAS network would also show how local knowledge can be used in different streams, as well as how they can be combined to make more detailed inferences. Using motion sensors with microphones could be used to determine what person/animal is near the sensor and this information can then be used to infer patterns as fallback sources when an image based classification cannot be made.

Our simulations in Chapter 7 show how K-HAS can use local knowledge, but it is by no means a complete implementation. We need to implement a more modular simulation that allows K-HAS to be applied to any form of WSN. We would like to simulate individual knowledge bases on every node and experiment with networks that contain multiple central nodes to visualise the flow of sensed data through the network and see if there is an effect, positive or negative, on the speed of interesting sensed data. A more immediate goal is to test K-HAS when all of the network uses a communication medium with lower range but a higher transfer rate, such as Wi-Fi, to determine how much Zigbee slows the delivery of sensed data when compared with processing.

We would also like to experiment with different ratios of knowledge processing capabilities on nodes to determine if there is an ideal ratio that maximises the flow of sensed data, delivering interesting data first and quickly but also delivering data that has been classified as not interesting within a shorter time period that would allow humans to act on the data if it had been misclassified.