Wireless Sensor Network Setup

Master's Thesis in Computer Systems Engineering

Javed Iqbal & Farhan Moughal

School of Information Science, Computer and Electrical Engineering
Halmstad University

Wireless Sensor Network Setup

Master Thesis in Computer Systems Engineering
Description of cover page picture/figure: typical wireless sensor network architecture with host base-station and sensors
Wireless Sensor Network Setup

Preface

First of all, we would like to thank Almighty Allah, the most Beneficial and the most Merciful.

We dedicate our work to our parents who are always with us whenever we need them and by their prayers we became able to do this work efficiently and timely.

We would like to express our sincere gratitude to our supervisors Edison Pignaton De Freitas and Tony Larsson, their feedback and valuable suggestions always let us think on a right direction. Their guidance and experience matter a lots in achievement of our thesis goal.

Javed Iqbal & Farhan Moughal
Halmstad University, June 2010
Wireless Sensor Network Setup
Abstract

Exploitation of wireless sensor networks (WSNs) in ubiquitous computing environments is continuously increasing for gathering data. Contemporary distributed software systems on WSNs for pragmatic business applications have become extremely adaptive, dynamic, heterogeneous and large scaled. Management of such system is not trivial to fulfil these features, leading to more and more complex management and configuration. Along with encompassing state of art and novel techniques for such diversely dynamic system, in this thesis two alternative techniques namely “task initiation by command” and “run-time task deployment and processing” are compared, for such system’s setup and configuration. Both techniques have their own pros and cons which makes them suitable according to the requirements and contextual situations. A lot of effort has been put to make WSNs more and more efficient in terms of computations and power consumption. Hence comparative analysis of both techniques used in this report to setup and configure WSN can be a benchmark to lead towards most appropriate solution to compensate the need of efficient energy and resource consumption.

Both alternative schemes are implemented to setup WSN on Sun Microsystems sunSPOT (Small Programmable Object Technology) sensor nodes which are embedded microcontrollers and programmed them in java (j2me). It performs radio communication between wireless sensors and host via sink node also called base station, along with over the air run-time management of sensors. SunSPOTs built in libraries and KSN libraries are used to implement these alternatives and compare the memory footprint, communication pattern and energy consumption.
Wireless Sensor Network Setup

Contents

PREFACE .................................................................................................................. 1
ABSTRACT ................................................................................................................ IV
CONTENTS ............................................................................................................... III
LIST OF FIGURES ................................................................................................... VI
LIST OF ABBREVIATIONS AND ACRONYMS ..................................................... VI

1 INTRODUCTION ................................................................................................ 1
  1.1 APPLICATION AREA .......................................................................................... 2
    1.1.1 Motivational Application Scenario ................................................................. 4
  1.2 PROBLEM STATEMENT ................................................................................... 7
  1.3 STUDIED PROBLEM AND PROPOSED SOLUTION ........................................ 8
  1.4 THESIS GOALS AND EXPECTED RESULTS ............................................... 9

2 BACKGROUND ................................................................................................... 12
  2.1 MIDDLEWARE ................................................................................................ 12
  2.2 MIDDLEWARE SERVICES ............................................................................. 13
  2.3 MIDDLEWARES/TOOLS APPROACHES ...................................................... 15
    2.3.1 Virtual Machine Approach .......................................................................... 15
    2.3.2 Event-, Message-, Publish/subscribe based approaches ............................ 16
    2.3.3 Modular Programming / Service Oriented approach .................................. 17
    2.3.4 Application Driven .................................................................................... 18
    2.3.5 Component based Approach ................................................................. 18
    2.3.6 Macro Programming approach ................................................................. 20
    2.3.7 Database Approach .................................................................................. 20
    2.3.8 Mobile Agents Approach ........................................................................ 22
    2.3.9 Data Compression approach ................................................................. 23
  2.4 CLOSELY RELATED WORK ........................................................................... 24
    2.4.1 Generic Multi–Packet Communication through Object Serialization ....... 24
    2.4.3 DARMA: Adaptable Service and Resource Management for Wireless Sensor Networks [34] ...... 28
    2.4.4 Flexible Integration of Data Qualities in Wireless Sensor Networks [35] ................... 30
    2.4.5 Enabling Mobility in Heterogeneous wireless sensor networks cooperating with UAVs for Mission Critical Management [37] .. 31
  2.5 MIDDLEWARE CHALLENGES ..................................................................... 32

3 INVESTIGATION OF SOLUTION ....................................................................... 36
  3.1 RPC (REMOTE PROCEDURAL CALL) .............................................................. 36
  3.2 JAVA RMI (REMOTE METHOD INVOCATION) ............................................. 36
  3.3 J2ME POLISH Serialization ......................................................................... 38
  3.4 KSN Serialization .......................................................................................... 38

4 DETAILED DESCRIPTION OF THE INVESTIGATED SOLUTION ................. 39
Wireless Sensor Network Setup

4.1 SUN'S SMALL PROGRAMMABLE OBJECT TECHNOLOGY (SUN SPOT) ............................................. 39
  4.1.1 Hardware Specs .................................................................................................................. 42
4.2 KARLSRUHE SENSOR NETWORKING (KSN) ........................................................................ 43
  4.2.1 KSN Radio Stack ................................................................................................................. 43
  4.2.2 KSN OTA Management ..................................................................................................... 43
4.3 ALTERNATIVE 1 (COMMAND BASED WITH DATAGRAM) ..................................................... 43
4.4 ALTERNATIVE 2 (LARGE OBJECTS DEPLOYMENT BASED WITH DATASTREAM) .............. 44
4.5 FUNCTIONAL DESCRIPTION OF APPLICATION ....................................................................... 44
  4.5.1 Demon mimicking .............................................................................................................. 44
  4.5.2 Receiver ............................................................................................................................ 45
  4.5.3 Transmitter ....................................................................................................................... 45
  4.5.4 Compression ...................................................................................................................... 45
  4.5.5 Serialization ....................................................................................................................... 45

5 TEST WITH ANALYSIS ................................................................................................................. 46
  5.1 TEST OBJECTIVES .................................................................................................................. 46
  5.2 TEST SCENARIO AND BASIC CONFIGURATIONS ................................................................ 46
    5.2.1 Alternative 1: Initiation (of functionality) through short commands ................................. 46
    5.2.2 Alternative 2: Run-Time deployment (of functionality) through large object .................. 46
  5.3 APPLICATION FOR ALTERNATIVE 1 ..................................................................................... 49
    5.3.1 Host Application Process flow .......................................................................................... 49
    5.3.2 SunSPOTs (Sensors) Application Process flow ............................................................... 50
  5.4 APPLICATION FOR ALTERNATIVE 2 ..................................................................................... 51
    5.4.1 Host Application Process .................................................................................................. 51
    5.4.2 SunSPOTs (Sensors) Application Process ....................................................................... 52

6 RESULTS ....................................................................................................................................... 53
  6.1 TEST METRICS ....................................................................................................................... 53
    6.1.1 Transmission time ............................................................................................................ 53
    6.1.2 Memory Utilization ......................................................................................................... 56
    6.1.3 Power Consumption ....................................................................................................... 58

7 CONCLUSIONS AND SUGGESTED FUTURE WORK ..................................................................... 60
  7.1 COMPARISONS OF OBTAINED RESULTS BETWEEN COMMAND BASED INITIATION APPROACH AND LARGE OBJECT BASED DEPLOYMENT ........................................... 60
    7.1.1 Single/Multi hop transmission cost .................................................................................... 60
    7.1.2 Memory Utilization ......................................................................................................... 61
    7.1.3 Power Consumption ........................................................................................................ 61
  7.2 CONCLUSIONS ......................................................................................................................... 62
  7.3 FUTURE WORK ...................................................................................................................... 62

8 REFERENCES .................................................................................................................................. 63
List of Figures

Figure 1 : Role of Middleware in WSN .................................................................................................................. 1
Figure 2 : Wireless sensor network system with middleware .................................................................................... 12
Figure 3 : (a) Feature trees of WSN middleware Legends [3] .................................................................................... 14
Figure 4 : (b) Feature tree of middleware services [3] ............................................................................................. 14
Figure 5 : (c) Feature tree of middleware architecture. [3] ..................................................................................... 15
Figure 6 : Middleware Features Table 1(a) ............................................................................................................... 34
Figure 7 : Middleware Features Table 1(b) ............................................................................................................... 34
Figure 8 : Middleware Features Table 1(c) ............................................................................................................... 35
Figure 9 : Middleware Features Table 1(d) ............................................................................................................... 35
Figure 10: A typical implementation of java RMI ................................................................................................... 36
Figure 11 : RMI classes [54] .................................................................................................................................. 37
Figure 12 : sunSPOT software stack [59] ................................................................................................................. 40
Figure 13 : Host application software stack ........................................................................................................... 41
Figure 14 : Sending commands to sunSPOTs ........................................................................................................... 43
Figure 15 : Sending large objects to sunSPOTs ......................................................................................................... 44
Figure 16 : Host application process flow chart for alternative 1 ........................................................................... 49
Figure 17: sunSPOT (sensor) application flow chart for Alternative 1 ................................................................... 50
Figure 18 : Host application Process flow for alternative 2 .................................................................................... 51
Figure 19 : SunSPOT (sensor) application process flow for alternative 2 ............................................................... 52
Figure 20 : Single hop transmission time of alternative 1 ........................................................................................ 53
Figure 21 : Multi hop transmission time of alternative 1 ........................................................................................ 54
Figure 22 : Single hop, multi hop transmission time comparison of alternative 1 ..................................................... 54
Figure 23 : Single hop transmission time of alternative 2 ....................................................................................... 55
Figure 24 : Multi hop transmission time of alternative 2 ........................................................................................ 55
Figure 25: Transmission time comparison between single hop and multi hop of alternative 2 .............................. 56
Figure 26 : Memory utilization of alternative 1 ......................................................................................................... 57
Figure 27 : Memory utilization of alternative 2 ......................................................................................................... 57
Figure 28 : Power drawn in mili Ampere (mA) of alternative 1 .............................................................................. 58
Figure 29 : Power drawn in mili Ampere (mA) of alternative 2 .............................................................................. 59
Figure 30 : Comparing single and multi-hop transmission time ............................................................................. 60
Figure 31 : Comparing memory utilization ............................................................................................................ 61
Figure 32 : Comparing power consumption .......................................................................................................... 61
# List of Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
<tr>
<td>SunSPOT</td>
<td>Sun Small Programmable Object Technology</td>
</tr>
<tr>
<td>KSN</td>
<td>Karlsruhe Sensor Networking</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>QOS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>ALERT</td>
<td>Automated Local Evaluation in Real Time</td>
</tr>
<tr>
<td>FWI</td>
<td>Fire Weather Index</td>
</tr>
<tr>
<td>NFDRS</td>
<td>National Fire Danger Rating System</td>
</tr>
<tr>
<td>EMG</td>
<td>Electromyography</td>
</tr>
<tr>
<td>ECG</td>
<td>Electrocardiography</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>AMS</td>
<td>Application Management Service</td>
</tr>
<tr>
<td>MIDP</td>
<td>Mobile Information Device Profile</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented approach</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>MBD</td>
<td>Model Based Diagnosis</td>
</tr>
<tr>
<td>Java EE</td>
<td>Java Enterprise Edition</td>
</tr>
<tr>
<td>Java ME</td>
<td>Java Micro Edition</td>
</tr>
<tr>
<td>J2SE</td>
<td>Java Standard Edition</td>
</tr>
<tr>
<td>MCU</td>
<td>Memory Control Unit</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>ACQP</td>
<td>Acquisitional Query Processing</td>
</tr>
<tr>
<td>SINA</td>
<td>Sensor Information Networking Architecture</td>
</tr>
<tr>
<td>DSWARE</td>
<td>Data Service Middleware</td>
</tr>
<tr>
<td>SObjects</td>
<td>Serialized Objects</td>
</tr>
<tr>
<td>DARMA</td>
<td>Distrinet Adaptive Resource Management Architecture</td>
</tr>
<tr>
<td>LooCI</td>
<td>Loosely Coupled Component Infrastructure</td>
</tr>
<tr>
<td>SRC</td>
<td>Service Resources Components</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreements</td>
</tr>
<tr>
<td>CLDC</td>
<td>Connected Limited Device Configuration</td>
</tr>
<tr>
<td>ECA</td>
<td>Event Condition Action</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedural Call</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>RMI OP</td>
<td>RMI optional package</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>CDC</td>
<td>Connected Device Configuration</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark-up Language</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/ Output</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>SHP</td>
<td>Single Hop Protocol</td>
</tr>
<tr>
<td>OTA</td>
<td>Over The Air</td>
</tr>
<tr>
<td>ASM</td>
<td>Application Service Manager</td>
</tr>
<tr>
<td>MB</td>
<td>Mega Byte</td>
</tr>
<tr>
<td>mA</td>
<td>Mili Ampere</td>
</tr>
</tbody>
</table>
Wireless Sensor Network Setup

1 Introduction

This is a modern age of technology where size of the electronic devices are available in scale of nanometre and performance is improving day by day with millions of computing components fit in small integrated chip. Recent integration of sensors with microcontroller and embedded microprocessors make this dream come true where integration of physical world with the digital technology is possible. This leads us with massive production of sophisticated systems and advancements in wireless technology promises to revolutionize the way of living, work and interact with the environment in new daily prospect.

The subject of sensor networks has been steadily more studied and researched especially in the computer science and electronics engineering. Wireless sensor network is the field most influence by a military application e.g. battle field surveillance but with their functional growth they used in many civilian areas and industrial application with the focus of efficient utilization and processing power.

Wireless sensor networks gained popularity quickly because the fact that they are very low cost solutions to a variety of real-world challenges. Low cost of sensor provides a freedom to deploy large sensor arrays in several conditions capable of performing many tasks. The way WSN work is significantly different from traditional computer networks, because WSN have very tight integration with the physical world around. Along with this sensor networks have some unique characteristics that make the development of applications non-trivial

There are many framework and approaches which make them efficient in terms of energy and processing power. The most common such system is called WSN Middleware which actually bridges the gap between low level WSN system and high level user applications as shown below in the figure. WSN Middlewares are discussed in details in chapter 2 of the report.

Figure 1: Role of Middleware in WSN
Wireless Sensor Network Setup

Sensor networks usually work in such a fashion that sensed data are disseminated towards host or destination node through adjacent/neighbour nodes. This process of dissemination can be static or dynamic. Our implementations cover both static and dynamic aspects of querying and data dissemination along with their comparative analysis to show the actual tradeoffs between them.

1.1 Application Area

Recent development in wireless sensor networks has shown their way into a wide range of application and system with vastly varying requirements and characteristics, as a result it has become difficult to discuss typical requirements regarding hardware and software. This causes a problem in WSN research area, where close interaction between users, hardware design, software developers and domain experts is needed to implement efficient system.

In the early days of Wireless Sensor Networks (WSN) programming software involved expertise in both hardware and networking. After that middleware architectures have been proposed to achieve a suitable abstraction from distribution and management tasks of sensor devices. This allows users to focus on the application development. A categorization of some applications according to the design space is elaborated below. Applications considered are adoptability, energy efficient, scalability, mobility, heterogeneity, application knowledge, usability, QOS, real time support, security, context awareness.

Sensor network are used to monitor physical or environment conditions around us, such as vibration, temperature, motion, pollution, sound, light, pressure etc. The Wireless sensor network consist of hundred, even thousand of nodes, the sensor node collect information from environment and send it to main node called sink node, there can be one or more sink nodes.

Each node in WSN is equipped with one or more sensor, a wireless communication device, processor and energy source usually a battery. Wireless topology for such networks can be dynamic or static because node may drop or move for various reasons and can be static where sensor are at fix location, however dynamic and static topologies have different protocols.
Wireless Sensor Network Setup

WSN works in limited computational power, storage capacity, communication range and limited energy source. Energy Constraints is a main limitation in WSN where sensor can have limited dedicated power and they are not rechargeable, it is known that nodes life depends on energy, more energy means longer sensor life, which can be obtained by reducing power consumption in different WSN operations.

Following are some WSN applications.

- **Applications**
  
  1) **Habitat monitoring**
  
  Great Duck Island [19] is a complete solution that provides habitat monitoring. In it whole network consists of 32 sensor nodes on a small island. This application provides mote based tiered sensor network for “Great Duck Island” on August 2002 by UCB/Intel research laboratory to monitor real world habitat monitoring. System design covers hardware design of the nodes and remote data access and management. This system architecture was designed for habitat monitoring of seabird nesting environments. This application provides important further work in data sampling, re-tasking, communication and health monitoring.

  2) **Animal life monitoring**
  
  ZebraNet [45] was developed to monitor animal’s life. The goal of this project was to implement, tests, and evaluate systems that integrate computing, wireless communication and non volatile storage with global positioning system (GPS). This is an effort in which biology and computer system disciplinary work together for studying power-aware, position-aware computing or communication systems. From the biology side, this project demands studies of animal migration and inter species interactions.

  From the computer science point of view it demands biological researchers to stringent enough to require real milestone in wireless protocols and low power computer systems design. It is a power-aware ad-hoc wireless sensor network with more bandwidth and computational needs.
Wireless Sensor Network Setup

3) Water Level Monitoring

As the name describes this system was developed to monitor water level. Example of its utilization can be flood detection, the ALERT [46] system deployed in the US. Rainfalls, water level, weather sensors are used in this system to detect, predict and hence prevent floods.

4) Fire Detection[47]

Sensor networks can be deliberately deployed to detect the origin of fire in the forest, resident area and coal mine. E.g. in Canada the Fire Weather Index (FWI) system being developed and the National Fire Danger Rating System (NFDRS) introduced by the National Oceanic and Atmospheric Administration.

5) Health Monitoring System[24]

WSN can be used in many health applications such as sensors in/on the body provide patients symptoms e.g. pulse sensor, EMG sensor, ECG Sensor, inertial sensor, blood pressure sensor to aggregator, and then transfer this information to the hospital database which is store for long time and can be used for medical exploration.

1.1.1 Motivational Application Scenario

In this part of report an application scenario is presented in the area of battle field surveillance to further motivate the work

- Battle Field Surveillance

Sensors are used to detect events and objects in the environment within a certain area; the common type of application is military operation, where the situation is not deterministic which means that the system has to deal with dynamic scenario where changes may occur in a short time. Frequent changes in weather conditions might require adaptation of the network in order to continue accomplishing user requirements. Similarly Changes in the user requirements may require the system to change the way it operates in order to continuously provide a good level of services, e.g. a user may need more information about specific phenomenon, data sampling and its period may increase or decrease. Changes in the network
Wireless Sensor Network Setup

topology represent another reason which makes changes in the network operations, e.g. node may fail, nodes may become out of range or interferences may occur etc.

- **Basic Design Concept**
  
  Sensor nodes are fixed on the ground and provide information. Mobile node can be used to acquire information of network from static nodes. Some groups of nodes have different task to do from other group of nodes. For instance, some nodes will measure temperature; others will measure the vibration, velocity, or electromagnetic field in the region. Task will be allocated according to the capabilities of nodes.

  The selection of sensor nodes for the right job is important aspect and can also improve the network performance. In a battlefield environment, static sensor nodes are densely deployed due to its mission critical nature. Since sensor networks are densely deployed, destruction of some nodes by hostile actions does not affect the system functionality as much as the destruction of a traditional sensor, i.e. a single sensor station, such as traditional radar, which makes sensor networks concept a better approach for battlefields.

- **Mobility and Tracking**

  In real world signals get attenuated and corrupted, which limits the system performance. To reduce the communication overhead, it is possible to carry mobile nodes by vehicles e.g. unmanned aerial vehicle (UAV) so that surveillance of large area can be achieved. Network deployment depends upon how much area each node can cover, i.e. range of the network node and how much time the mobiles node will take to collect information from the static nodes. WSN can either be range based or range free. Range based WSN, first determine the distance between two nodes and then decide the location of the node. Extra hardware is used to determine the distance by time difference of arrival sound and radio waves. In range free scheme, distance is determined using hop count technique.

  The main objective of this scenario is to alert military control unit upon the occurrence of events in the area of interest. Events can be presence of persons, soldiers of the opponent force, and unauthorized vehicle. For surveillance purpose it includes tracking service and classification components. Velocity calculation provides the target speed and next position of the object.
Wireless Sensor Network Setup

Time synchronization and node localization are important for a surveillance application because detection and tracking process relies on multiple calculations by multiple nodes. In environment each node should be aware of its own location. Some nodes are responsible for tracking and some are for detection and while other will be in sleep mode until an event occurs in their range. If events occur in their region the node awakes others nodes in the region.

Sensors can be deployed randomly in inaccessible regions and critical areas for monitoring the presence of opposing forces. Furthermore they are also useful to discover approach routes and paths in scenarios without human interruption.

In surveillance area phenomena of interest may disturb the environment in several ways, such as thermally, seismically, acoustically and chemically. For instance, human bodies emit infrared (IR) energy in Omni-directions from the source. Human foot print can also cause ringing at the natural frequencies of the ground and also cause acoustic signal that travel in the air at different speed then the seismic effects of footsteps. Human bodies are also considered dielectric that may cause a change in electric field and also emits chemical smell.

Besides, a soldier usually carry gun and other equipment that contain steel or other equipments or it is also possible to detect other metal so soldier equipment create more magnetic field then unarmed person or it is also possible to detect soldiers with better reflection and scattered electromagnetic signals using radars.

Vehicles can disturb environment thermally, seismically, acoustically, electrically, magnetically, chemically and optically like humans. Vehicles contain more metallic mass that affects the electric and magnetic field in a larger area then a soldier. Moreover, vehicles emit carbon dioxide and also reflect, scatter and absorb electromagnetic signal, acoustic and ultrasound.
Wireless Sensor Network Setup

1.2 Problem Statement

The data-aggregation model for sensors has many limitations and problems. These limitations are mentioned below.

1) Adoptability need to be focused in wireless sensor network where frequent changes require the change in task perform by sensor nodes, e.g. change in weather condition, or user may need to increase sampling rate of data.

2) Selection of sensor for a specific task is also very important where sensor node should be capable to perform the new task in heterogeneous environment.

3) Communication overhead in existing communication protocols for relatively small sensor data. The data sensed by the sensor is very small, but for sending this small data over network, the header overhead of the protocol make this data large enough.

4) Wireless sensor network usually use single frequency channel for receiving and sending, so the bandwidth is shared among all the neighbour nodes, efficient sharing of bandwidth while keeping low power consumption is a big task. [38]

5) Limited energy and resources
   Nodes are subject to failures due to exhausted batteries. Limited, also typically means restricted resources (CPU performance, memory, wireless communication bandwidth and range.

6) Environmental influences
   Due to environmental influences and interference node failure rate increases

7) Heterogeneity
   Another issue is heterogeneity. WSN may consist of a large number of different nodes in terms of sensors, computing power, and memory.
8) Scalability

The large number of sensor nodes raises scalability issues on the one hand, but provides a high level of redundancy on the other hand. Also, nodes have to operate unattended, since it is impossible to service a large number of nodes in remote, possibly inaccessible locations.

1.3 Studied Problem and Proposed Solution

Wireless sensor networks are proposed to deliver observations at low cost for longer period of time. Recent work in this field has devoted lots of effort towards different challenges e.g. energy efficiency, mobility of nodes and target objects, context aware processing, routing, heterogeneity of nodes etc. These challenges are satisfied by different researches to some extents but still there is need of a sort of comparison between different possible techniques that can be used to overcome these challenges.

For example very broadly speaking of the functionality required from a wireless sensor network system is gathering of valuable sensor values for different number of times in different intervals. Now depending on the particular requirement which can be as simple as, to get temperature value at a certain place, to identification and location of a moving object, there can be different alternative possible implementations. Among different implementation for these simple or complex requirements on one extreme there can be just triggering of a waiting application through a very small piece of alert and on the other extreme it can be transmission and deployment of a totally new application to perform the task. From the implementation and different comparisons of these extreme cases which cover all the possible functionality that case be required from a WSN, decision about the selection of efficient scheme can be easily made.
1.4 Thesis Goals and Expected Results

Recently a lot of effort has been put to make wireless sensor networks more and more energy efficient and dynamic. This is due to the enhancing application area that is covered by wireless sensor networks now a day. In such diverse and dynamic wireless sensor network systems in broader sense there are two alternatives that need to be done to perform certain activities or tasks.

**Alternative 1**
Initiation of process on target node(s) by host, through short commands.

**Alternative 2**
Run-time deployment of a new functionality of target node(s) through some large object that includes the functionality to be deployed.

In first case target node already has the required functionality ready to be executed but it is waiting for an event or trigger to activate it. So a very short notice, a command is need to be sent to the node to activate its functionality. This short notice or command can off course contain configuration parameters required to perform that particular task.

In 2nd case target node(s) do not have the required functionality already there which is a more dynamic and robust situation and can be though in this way; target node(s) are required to perform some operations that has been discovered lately after the prior deployment of functionality over target nodes, so in this case new required functionality need to be deployed.

Following are the detailed thesis goals:

- **Implementation of Alternative 1 on Wireless Sensor Network**
  This case requires communication of very short data (commands) towards target sensors and gathers results back towards host. So datagram communication technique is used, since data needs to be sent is short so no reliability overhead is required.
Wireless Sensor Network Setup

- **Implementation of Alternative 2** *(run-time software deployment to target node(s) through large object)*
  
  This case requires pretty much attention than that of previous one because of its dynamic configuration and reliable communication requirement. New software to be deployed at run time will be very large in size as compared to the other alternative and thus require reliable communication. Loss of any bit can cause untoward deployment of failure of overall process.

To cope all these factors following techniques need to be adopted:

1. **Serialization**
   
The large object that includes the software of functionality to be deployed should be serialized for efficient transmission. To do so, both host and sensor applications need to make serialization mechanism aware.

2. **Compression**
   
   To reduce communication cost of large object it will be compressed and transferred and then decompress on the receiver side. Although compression and decompression is overhead but due to the large size of object it is compensative.

3. **Run-Time Deployment**
   
   On the receiver or target nodes side, application must be efficient enough to receive image of the application to be installed and then install it successfully. So this is a kind of management services which are need to be implemented in our second case. To do so application management service (AMS) on the MIDP based systems are controlled to perform these management operations.

- **Comparison of these both alternatives in terms of:**

  1. Radio Transmission Time
  2. Memory utilization
  3. Power consumption
Expected results and analysis of proposed goals

Thinking about the both alternatives one can easily say that off course second implementation or alternative will need more transmission time, processing power and memory. On the other had dynamicity to fulfil the new demanding application need is obtained in second alternative which is indispensable. So still there are some questions to be answered e.g.

- In what relation actually the transmission cost, energy consumption and memory utilization vary with other alternative?
- On which situations one alternative can be chose instead of other one.
- On what terms overhead of large object based deployment technique is compensated.

To answer these and such related questions these alternatives needs to be implemented and compare them to find out their appropriate utilizations according to the requirements and context.
2 Background

This chapter describes the technologies and background information that has been used to overcome different challenges faced by WSN systems. Most of these are the frameworks proposed for WSNs named middleware, and their comparison with their approaches and principle features. Section 2.1 provides some important wireless sensor network related terminologies and definitions. Section 2.2 gives challenges of WSN middleware and framework. Section 2.3 gives a comparison between these techniques and framework.

2.1 Middleware

Middleware is used to bridge the gap between operating system (low level component) and the application to ease the development of required applications which are usually distributed. Wireless sensor networks are also distributed systems, so there is a need for distributed computing middleware in such networks. Following figure below illustrates detailed architectural overview of WSN system with middleware.

![Wireless sensor network system with middleware](image)

Figure 2: Wireless sensor network system with middleware
2.2 Middleware Services

✓ Code management
   a. Allocation of code nodes
   b. Migration of code among nodes

✓ Data management
   a. Acquisition of data
   b. Processing of data can be centralized, at node level or at network level.
   c. Storage of data can be local to node, external or both.

✓ Resources and information discovery
   a) Resource discovery service returns the data type that a discovered node can provide, the modes in which it can operate, and the transmission power level or residual energy level of a sensor node.

   b) Information discovery service returns the information about the network topology, the network protocols, the neighbours and the locations of the discovered nodes.

✓ Resource management
   Resource configuration at setup time and resource adaptation at runtime.
   Sleep scheduling protocols in the MAC layer and CPU voltage scaling in the physical layer for energy efficiency along with clustering or grouping mechanisms.

✓ Integration to external world
   Integrated into other exiting network infrastructures

✓ Run-Time Support
   Runtime support extends the functions of operating system for processing, communication and storage management to provide well defined execution environment for applications and system programs. These services include inter-process communication, memory control and power control in terms of voltage scaling and component activation and deactivation, along with task coordination (cooperation of sensors to complete a distributed task).
Wireless Sensor Network Setup

✓ **Quality of Service**
Real-time support for jitter (delay, latency), security, reliability, delay control, and loss control, throughput effectiveness.

Figure below expresses features of middleware for both service and architectural point of view in a tree form. For legends (a), for middleware service features (b), for architecture features (c).

![Feature trees of WSN middleware Legends [3]](image)

**Figure 3 : (a)** Feature trees of WSN middleware Legends [3]

![Feature tree of middleware services [3]](image)

**Figure 4 : (b)** Feature tree of middleware services [3]
2.3 Middlewares/Tools Approaches

2.3.1 Virtual Machine Approach

VM provides high level abstraction to application programmers for implementing low level distributed tasks. Such middlewares are built on VM, Interpreters and mobile agents [1].

a) Mate

Mate is a byte code interpreter built on TinyOS [9]. It provides better interaction and adaptation to the changing nature of network using its active message approach to update network parameters and protocol by injecting new capsules. For energy and context awareness, Mate is only suitable for sleepy applications that are in low-duty cycle most of the time. For complex applications, it is not suitable because of its instruction interpretation overhead.

b) Magnet

Built to work on top of Magnet OS, a power-aware, adaptive operating system specially designed for sensor networks and ad hoc networks. It provides an abstraction for the heterogeneity by constitute a layer defined Single System Image [8]. Using it programmers can adjust object placement and migration, which reduces network communication by moving objects closer to data source. Application programmer writes a single java program, which is
Wireless Sensor Network Setup

being partitioned, placed and migrated by the runtime system. It offers application adaptation and network scalability [2]

2.3.2 Event-, Message-, Publish/subscribe based approaches

It is based on event identification and notification mechanism. Applications specify interests in state changes and upon detection of such event; the middleware will send notification to interested applications.

a) Mires

It is based on publish/subscribe communication in which information supplier publishes messages that are forwarded to one or more subscribers. In Mires, only the messages referring to the subscribed topics are sent, hence reducing the number of transmissions and energy consumption. It makes use of a Pub-Sub mechanism to support the routing management. Owing to the loosely coupled interactions between the nodes in the Pub/Sub paradigm, it is very flexible to provide new kind of data routing implementation [3].

b) Enviro Track

It is object-based programming model that provides a convenient and powerful way to program sensor network applications that track activities in their physical environment. It adopts a data centric programming paradigm called attributed-based naming through “context labels” [2], where the routing and addressing are based on the content of the requested data rather than the identity of the target sensor node.

c) A Rule-based Acceleration Data Processing Engine for Small Sensor Node [27]

A System built on rule based engine; consisting of various types of sensors with computing power to recognize events in the real world which enable the deployment of new services dynamically in real time, based on large amounts of information such as events, contexts, and situations. Among sensors the acceleration sensor is one of the most significant sensors for obtaining data as dynamic motions in real time. Data processing is available on sensor node and reduces communication over head and sends data when event is detected by engine.
Wireless Sensor Network Setup

2.3.3 Modular Programming / Service Oriented approach

The concept of this approach is to modularize applications as much as possible to facilitate their injection and distribution through the overall network by mobile code, which will reduce transmission overhead.

a) Impala
It was designed as part of ZebraNet project (a wildlife watch project). It is based on asynchronous event based mechanism that uses mobile agents (program modules) which are compiled into binary instructions. It ensures application adaptation and can automatically discern needed parameters settings or software usages. New protocols can be plugged in at anytime and switches between protocols can be performed [10]. It supports dynamic applications by providing network updates. It also increases fault tolerance and network self organization by Passive replication (code is updates by a primary replica, backup replicas are then synchronized). But the fact is: it is not suitable for those devices with limited resources.

This is a frame work to support the application designer to make easy to use security setup for WSN. For security evaluation of this setup, it used a formal and structural approach. Each attack path is evaluated with respect to its possibility. It provides feasible security setups, and security evaluation, the application designer first has to specify the desired abstract security service for each data type, e.g. confidentiality for location data. This defines how this data type will be handled security-wise when communicated in the network during operation.
Wireless Sensor Network Setup

2.3.4 Application Driven

In this approach developers are able to control network based on the application requirements through an architecture that allows access to network protocol stack.

a) Milan

Its focus was to provide high level interface based on the application requirement, affecting the entire network. Milan allows the increasing of applications lifetime by specifying application specific quality needs. These goals are accomplished by the representation of requirements from applications to Milan through specialized graphs which incorporate state-based changes in application requirements [2]. It is well suited for application adaptation and QOS requirements. However its architecture lacks innovation in new programming models suitable for WSN and heterogeneity because Milan should know a lot of information about each sensor to work.

2.3.5 Component based Approach

Such architectures provide good support for dynamic configuration and standard interface for integration of components. To save energy components can switch between on or off modes.

a) Runes [5]

It is architecture for WSNs which provides heterogeneity in hardware, link layer and networking protocols. Besides it offers simple programming language independent APIs. Reconfigurability support allows great programming flexibility and low implementation overhead. Dynamic reconfiguration and wirelessly uploading and deployment of new components can be performed at runtime which offers great flexibility.

b) MoMi - Model-Based Diagnosis Middleware for Sensor Networks [32]

MoMi is a middleware component for identifying causes of failures in sensor network deployments. System health is checked in a decentralized manner based on a middleware abstraction.
Wireless Sensor Network Setup

Model-Based Diagnosis (MBD) is a framework, independent of hardware and OS. Short model of normal behaviour of a system is provided by a user instead of failure model describing each one of them. MoMi has been implemented for TinyOS 2, consists of a local component on every sensor node and a diagnosis host receiving information via a sink node. The sink node forwards them upon detection of new symptoms over the serial port to a host, so that host performs the final diagnosis.

c) Towards policy-based management of sensor networks [29]
Is a way to tackle heavy weight process of traditional policy based management into smaller building blocks and achieve the maximum heterogeneity and distribution of the environment for hosting them and deployed according to host capabilities. Due to various change in environment and application they can never be static in nature.

Policy based middleware approach offers management of distributed sensor applications and provides abstraction to large scale enterprise system. Policies can be implemented on managed resources which communicate with each other using a communication bus and enables configure them equally and implement the behaviour according to business requirements.

Architecture for this middleware is formed as an independent abstraction layer which builds on component based middleware. (i.e. where services are implemented in terms of software components). Examples of such middleware’s are: Java EE on back-end systems OSGi or GridKit on gateway devices, or the nesC component model on sensor nodes.

Idea behind Lorien OS for WSN is to build a component-oriented OS that allows dynamic component-based changes to itself entirely i.e. on every component of it even the kernel. Its model provides general primitive of changes through it can load, unload and interconnect components at runtime without restarting system by reprogramming. Lorien is built upon the well-established OpenCom component model. A component in Lorien is a strongly separated, self-contained unit of functionality. In Lorien currently running system or components resides in the MCU’s on-board cache memory and components can be individually loaded from external memory as desired. There is only very a small set of components that only deal with boot-strap the system which is a static part, all other components are dynamic.
Wireless Sensor Network Setup

2.3.6 Macro Programming approach

Provide higher-level abstractions for expressing local node behaviour.

a) *Kairos*

Kairos is a control driven programming paradigm providing a distributed shared memory abstraction to node level programming. It allows programmer to program the global behaviour of distributed computation on entire network in a centralized way. Kairos compile-time and runtime subsystems provides some programming primitives through which hides the details of distributed code generation and instantiation, remote data access and its management, internode coordination of program flow, from the programmers.[6]

2.3.7 Database Approach

The database approaches treats the whole sensor network as a large “virtual” data base. The interaction with sensors is done by querying using SQL-like languages. It is easy to use and suitable for a number of applications.

a) *Tiny DB*

TinyDB is a distributed middleware and is designed and implemented as Acquisitional Query Processing (ACQP) system for collecting data in a sensor network, it runs on the top of TinyOS operating system, with SQL like interface to execute data from sensor nodes .It has many of the features of a traditional query processor (e.g., the ability to select, join, project, and aggregate data)

b) *Cougar*

Cougar is a middleware which adopts a database approach in which sensor data are considered a virtual relational database. It implements WSN management operations in the form of queries, using an SQL-like language. A system comprises of sensor database and sensor queries. Cougar provides a distributed database interface to the information from a sensor network.
Wireless Sensor Network Setup

c) SINA

SINA is a middleware that stands for “Sensor Information Networking Architecture”. It has a cluster based middleware design which focuses on the cooperation among sensors to conduct sensing tasks. Its extensive SQL like primitives can be used to issue queries into the sensor network. However, it does not provide schemes to hide the faulty nature of both sensor operations and wireless communication. In SINA it is the application layer that must provide robustness and reliability for data services. SINA incorporates low level mechanisms for hierarchical clustering of sensors for efficient data aggregation as well as protocols that limit the re-transmission of similar information from geographically proximate sensor nodes.

d) DSWARE

DSWare (Data Service Middleware) is middleware which has a specialized layer that integrates various real-time data services for sensor networks and provides a database-like abstraction to applications. Unlike SINA, which relies on the application layer to provide robustness, DSWare contains modules which supply features to ensure robustness. Data storage and data caching features provide the core foundation for the robustness.

e) Senceive

The middleware is designed as database system like TinyDB. The basic three tiered architecture is inspired by Crossbows MoteWorks. The implemented routing protocols are improved versions of the sdlib protocols; and the query processing is similar to the Cougar approach. Along with existing state of art low weight and efficient query processing techniques senceive also provides a highly flexible network configuration (management) interface to enable network administration.
2.3.8 Mobile Agents Approach

Mobile agent is a middleware, used for applications modules for injection and distribution through the network using mobile codes. The sensor network can implement tasks by transmitting application modules. The use of modules provides a more energy efficient code migration then transmitting a complete application.

a) Agilla

Agilla is a pure agent based middleware platform built on top of TinyOS sensor mote operating system. It allows the rapid deployment of applications in wireless sensor networks (WSNs) for adaptability. Special programs called mobile agents can be created and injected by users that coordinate through local tuple spaces of nodes. They migrates tuple space across the WSN to perform application-specific tasks. Agilla is based on MATE, but unlike MATE which divides application and flooded, Agilla injects application on mobile agents. Mobile agents can move or clone themselves towards a desired location. Each node has a local tuple space. The aim of Agilla is to extend the functionality of TinyOS and provide on the fly code updates to allow for dynamic adaptation due to environmental changes. This allows flexible application deployment, resulting in a sensor network platform that can run different applications in multiple sessions.

b) Sensorware

Sensorware defines and supports lightweight and mobile control scripts that allow the computation, communication, and sensing resources at the sensor nodes to be efficiently harnessed in an application specific fashion, through the use of abstraction services. A key feature is that the run-time abstraction can change by dynamically defining new services.

c) Tiny Lime

Tiny lime is a middleware based on tuple space model for wireless sensor where sensor data is not collected by a central monitoring station rather it enables multiple mobile stations to access the sensor nodes in their proximity and sharing of collected data through wireless links. Applications where sensors are sparse/isolated and on-site, location dependent data gathering is required, demands this basic context-aware setting.
Wireless Sensor Network Setup

d) Teeny Lime

TeenyLIME is a middleware for sensor networks based on the tuple space model LIME. TeenyLIME operates by distributing the tuple space among the devices, transiently sharing the tuple spaces contents and introducing reactive operations that fire when data matching a template appears in the tuple space.

e) TS-MID

TS-Mid provides communication based on tuple space (shared memory). Hence, its threats provide an asynchronous and decoupled communication style on both time and space. Together the use of tuple spaces, TS-Mid has another basic characteristic that refers to the ability of creating logical regions inside WSNs. Collected data are aggregated within these regions, which results in a considerable economy of resources, and consequently an increase in lifetime of the WSN.

2.3.9 Data Compression approach

a) Lightweight Tracing for Wireless Sensor Networks Debugging [33]

It is a tool for WSN, which uses control flow tracing and encoding scheme to detect and manage complex runtime problem and fault, e.g. race conditions, and generates high compress control flow trace. Tracing tools support trace to store them in non volatile memory and interface enable base station to get back this trace when required.

For long running program, the trace tool incurs very low overhead. On run time trace is recorded in memory and compressed before written in external non volatile flash memory. Upon detection of problem, a trace is send to the base station for analysis and debugging.
Wireless Sensor Network Setup

2.4 Closely Related Work

2.4.1 Generic Multi–Packet Communication through Object Serialization

It is an object serialization mechanism (ObjectStreams) to enable multi-packet stream communication, while minimizing memory requirements. This mechanism results up to 3.5 times reduction in communication and more reliability in communication compared to state of the art methods to achieve same functionality.

ObjectStreams transfers collections of linked lists (data structure) objects in a sequence of packets, which makes it feasible for any payload size. It can be used with any existing MAC, routing and transport protocols for WSNs and provides reliability through acknowledgments and retransmissions.

- **ObjectStreams**

ObjectStreams is a library which provides operations to allocate access and de-allocate SObjects (its generic datatype) enabling multi-packet streaming communication of SObjects (serialized objects). SObjects can contain four different kinds of data (integer, float, pair, Null) and the library provides a number of conversion routines to make it easier the use of SObjects in WSN application code written in NesC.

- **Serialization**

For communication ObjectStreams serializes (method to transport language level objects on network) SObjects as a sequence of bytes in packet(s) (filled at max. payload size) to be transported and then reconstructed on the receiver side as SObjects. Encoding method uses pattern matching on the type of SObjects to define the operation to be performed in encoding routine. Tokens encode which one of four possible types of data is encoded next, after which the actual data follows. The SObject is traversed and tokens emitted until the packet's payload section is filled, or the entire SObject structure is encoded.

Similarly, receiving nodes decode the contents of a packet sequence on a per-packet basis. During decoding, tokens are read from the packet payload, and the encoded SObject structure is created, allocating SObjects as needed.
Wireless Sensor Network Setup

- **Packet Sequencing**

Packet header defined by the ObjectStream protocol takes care of the ordered transmission of packets through sequence numbers without missing packets and avoiding duplications. At the same time it is compact enough to induce only marginal overhead so size can be calculated only by traversing the entire structure. Payload is carried in a stream of packets, each consisting of a small header, and a payload field to carry the stream content. The tuple (sequence number, source address), called a packet ID, uniquely defines every individual packet. Therefore, no two packets with the same packet ID should be alive in the network at any moment.

- **Evaluation**

To evaluate this protocol it was used a simulated network of nodes to compare with two alternative implementations which use existing WSN communication protocols. The first (single-packet) is an implementation of protocol that transports single measurement per packet. The second (multi-packet) operates similar to ObjectStreams implementation (sending a single multi-packet in each time interval) but uses fixed static buffers for data storage from neighbours.

All three implementations send data items containing three 16-bit sensor values and a 16 bit network address, and use packets with 28 bytes of payload. It is concluded that ObjectStreams implementation is independent of network size. Three implementations behave quite differently, single packet implementation uses least memory and achieves best performance and multi-packet implementation is the most energy-efficient as far as communication is considered at the cost of considerable memory requirements. In between these two, ObjectStreams operates energy-efficiently, while keeping memory consumption low.
2.4.2 A Self-Adaptive Context Processing Framework for Wireless Sensor Networks [28]

In WSNs context aware computing sensor nodes are usually considered as the data collectors, regardless of considering context elements. This work firstly proposes an approach to incorporate context information in modelling sensor network and then a middleware framework to map the context model to software components.

The model is based on the Context Node which is actually virtual sensor nodes specifying some context information and is distributed over physical sensor nodes in the network according to its role in context model and type of information. During the development of context nodes they are mapped to context data processing software components, which make it dynamically reconfigurable according to the context changes.

In general context management system for WSN can be seen as composed of three layers.

a) Context Collector: It collects information and data from the environment along with the user preferences, in these proposal sensors do this responsibility.

b) Context processor: Raw data coming from context collectors are filtered, aggregated, computed and then fed into the adaptation layer, processing is usually done by high energy nodes normally sink or cluster.

c) Context adaptation: it decides whether to perform adaptation or not on the based on the collected data from processor.

- **Context node Architecture and context model**

A context node is implemented as a composite component that contains at least five primitive components: the context processor, the context reasoner, the context configurator, the activity manager, and the message manager. Context Node interacts by exchanging messages and can be either active or passive. An active node periodically gathers messages through activity manager while a passive node obtains messages on demand. This context information is processed by context processor into context information so that it can be used by acting upon some functional or non-functional requirements. These actions are planned and executed by a context reasoner and a context configurator, respectively. The context model supports sharing
Wireless Sensor Network Setup

of context reasoned, configurator, and activity and message managers across adjacent context nodes to reduce memory usage and resource consumption. Context nodes can be placed in any layer of context processing system based on its functionality. In context model context nodes can be shared between sensor nodes hence sharing of context management policies. In addition to the propagation of context information, the context nodes can embed some autonomic behaviour to react to context changes.

- Implementation of a context middleware

The middleware run-time system is responsible for managing context nodes and their interactions according to the context model descriptions. Observation and notification mechanisms are implemented by a functional interface and reconfiguration is provided by the control interface. Context model is mapped to context components according to the logic behind the context component (a portion of context model) and then deployed over sensor nodes.
Wireless Sensor Network Setup

2.4.3 DARMA: Adaptable Service and Resource Management for Wireless Sensor Networks [34]

DARMA (Distrinet Adaptive Resource Management Architecture) is light weight, simple, flexible, SLAs (Service Level Agreements) based service platform to meet the requirements of sensor networks. SLAs provide abstraction through which it is possible delegate decision making to the point of action (physical location where action occurred), which help in decreasing process execution and transactional cost, providing quicker response time and hence improving service quality. It has been evaluated using LooCI (A Loosely Coupled Component Infrastructure for Embedded Network Eccentric Systems) on SunSPOT platform.

- Component Architecture
LooCI [38] is designed to support embedded Java ME platforms such as the SunSPOT. LooCI components which are extendable are indirectly bound over the event bus and they define their interfaces as the set of events to publish. The receptacles of a LooCI component are similarly defined as the events to which they subscribe. Each LooCI event has a globally unique identifier for its classification. Traditional Web Service approaches for services used in this paper are too heavy-weight to be feasible in embedded WSN environment. Component-based reconfiguration offers adaptation mechanisms to manage this dynamism and allow for system evolution through the deployment of new components.

- ADAPTABLE RESOURCE MANAGEMENT ARCHITECTURE
The core DARMA architecture has four key conceptual elements; service managers, service selection, service resources (SRC) and context monitors. Each system element is modelled as a LooCI component.

- Service Managers
Each service manager maintains references to collections of service resources (remote components) and takes a number of SLAs as input and based upon SLA’s specification it will select most appropriate service (using service selection component) informed by context monitors running on each node.
Wireless Sensor Network Setup

- Service selection component

It maintains a directory of all services offered within the managed group, which is used to match service requests to available services.

- Service Resource Components

Service Resource Components (SRCs) extends basic LooCI components by allowing multiple applications to use the same service concurrently.

- Context Monitors

Context monitors are standard LooCI components and may be deployed on any sensor node. Context monitors provide contextual data to service managers and SRCs

• DARMA OPERATION

Operation of DARMA can be seen as composed of four phases

1. Service Request: is specified using a light-weight, high-level and human-readable specification language, identified by unique ID.

2. Service Initiation: When a service request is received, the service selection component inspects all nodes in the target specification by comparing context data and attempt to parameterize the optimal service as requested and initialize it.

3. SLA enforcement: After service initialization the service selection components periodically polls all nodes within target specification, gathering context data and where non optimal SRC service serving request is found it is swapped with the most optimal available.

4. Service Termination: Each service request also contains service termination specification which lets service manager to terminate it based upon time-out or reception of a unique event.
2.4.4 Flexible Integration of Data Qualities in Wireless Sensor Networks [35]

It focuses on the non-functional requirements of WSNs to enable the integration WSNs with enterprise environments which demand certain challenges to be addressed like Quality of Data and Security.

In WSN context these features are not easy to be implemented so to make them appropriate and easily implementable it was designed a high-level policy-based approach. It provides a lightweight framework to integrate those enterprise non-functional level requirements on the underlying run-time reconfigurable component model. A prototype of this framework for the LooCi component model was realized and evaluated. This approach is to combine the benefits of reconfigurable component model with the efficient high-level non-functional policy based rule set.

Implementation of this framework on ME CLDC 1.1 compliant SunSPOT nodes representing a single policy is very efficient as it only consumes 376 bytes of memory in average, as compared to implementing LooCI component which require 26 KB in average.

- **Policy Framework**

The framework uses policy rules which follows Event-Condition-Action (ECA) approach and is similar to an event-driven approach usually used in WSN platforms. ECA consists of Triggering events, optional logical expression based condition and a list of response actions to be enforced. This policy framework can be deployed on each sensor node, gateway, or backend system according to the requirements and mainly consist of three key components

1) **Policy Engine**

   Policy Engine is main component of framework which is responsible for detecting events and evaluating them based on the policy rules in each node. When a match occurs between the event and policy rule, it enforces the action to be performed in the action part of the policy.

2) **Policy Distribution**

   PD distributes policy files from back-end to the sensor network in compact binary form so that they can be efficiently deployed by rule manager.
3) Rule Manager
The Rule Manager on each individual sensor node is responsible for storing and managing the set of policy rules on the node. After reception of a binary policy from the distribution component, the rule manager converts the policy into a data structure, suitable for more efficient evaluation, which is then passed to the policy engine on a per triggering-event base.

2.4.5 Enabling Mobility in Heterogeneous wireless sensor networks cooperating with UAVs for Mission Critical Management [37]

This is a platform, which proposes processing and communication of mission critical applications with low latency, reliability and success ratio of delivery and dynamics, and self-adapting. Sensor nodes (including sink node) for environment monitoring can be static or dynamic, and mobility of nodes has challenges with them. They can be used in target detection or disaster management. For instance sensors deliver their data to specific sink nodes and multiple mobile sinks in WSNs can provide fast energy-efficient data collection, with well designed networking protocols.

It has a two-layer architecture designed for heterogeneous mobile WSN, dynamic networking under mobility conditions and distributed services. Instead of each sensor sending its own data report directly to the data sink; it has a global-local gradient paradigm to only send aggregated data from centre of the event to the data sink.
2.5 Middleware challenges

✓ Adaptability
Reactivity or pro-activity of the network according to the changes in environment.

✓ Energy Efficiency
Power saving due to the limited resources like energy, computing power, memory, and communication bandwidth.

✓ Scalability
Ability to maintain performance characteristics irrespective of the size of the network.

✓ Heterogeneity
It considers sensor nodes that execute different tasks and or have different hardware and sense device platforms.

✓ Mobility
- Node mobility: considers that the nodes that compose the sensor network can be mobile.
- Observed phenomena (object) mobility: considers mobility of objects that compose the observed phenomenon, for example, mobile object in movement tracking applications.

✓ Application knowledge
Ability to handle application specificities related to a given application domain.

✓ Usability
Ease of use for the WSN application developer.
Wireless Sensor Network Setup

✓ Quality of Service
Support for information accuracy, coverage and reliability etc.

✓ Real-time Support
Support for accuracy in assuring-temporal properties of operations.

✓ Security
Security assurance from intruders. Attackers may disturb or control a sensor network by different ways, for example injecting false data into the network, or taking the data gathered by the sensor nodes, among others.

✓ Context Aware
Whether middleware keeps track of the currently executing operation’s context or not.

1) Levels of support to these challenges

a) Full
If the middleware provides full support to the specific challenge.

b) Partial
If the middleware provides support to a challenge to a given extent. For example in the context of energy efficiency and context awareness, Mate is only suitable for sleepy applications that are in low-duty cycle most of the time. For complex applications, it is not suitable because of its instruction interpretation overhead.

c) None
Middleware does not support that challenge at any level or may be very nominal.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Scalability</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
</tr>
<tr>
<td>Mobility</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>Partial</td>
<td>Full</td>
</tr>
<tr>
<td>Application knowledge</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
</tr>
<tr>
<td>Usability</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
</tr>
<tr>
<td>QoS</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Real-time Support</td>
<td>None</td>
<td>Partial</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>security</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Context Aware</td>
<td>Partial</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
</tr>
</tbody>
</table>

Figure 6: Middleware Features Table 1(a)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Scalability</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>Partial</td>
<td>None</td>
<td>Partial</td>
<td>None</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Mobility</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>full partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Application knowledge</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
<td>None</td>
</tr>
<tr>
<td>Usability</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>QoS</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>Partial</td>
<td>None</td>
</tr>
<tr>
<td>Real-time Support</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>partial</td>
<td>Partial</td>
</tr>
<tr>
<td>security</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Context Aware</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
</tbody>
</table>

Figure 7: Middleware Features Table 1(b)
### Wireless Sensor Network Setup

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>Partial</td>
<td>Partial</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Scalability</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>None</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Mobility</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Full</td>
</tr>
<tr>
<td>Application knowledge</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Usability</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
<td>None</td>
<td>Partial</td>
</tr>
<tr>
<td>QOS</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Real-time support</td>
<td>None</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Security</td>
<td>None</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Context Aware</td>
<td>None</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
</tr>
</tbody>
</table>

**Figure 8**: Middleware Features Table 1(c)

<table>
<thead>
<tr>
<th>Features</th>
<th>DARMA [34]</th>
<th>Flexible Integration of Data qualities[35]</th>
<th>Multi packet Streamedserialized communication [31]</th>
<th>Self Adoptive context processing[28]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>Partial</td>
<td>Partial</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Scalability</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>Full</td>
<td>Full</td>
<td>Partial</td>
<td>Full</td>
</tr>
<tr>
<td>Mobility</td>
<td>Full</td>
<td>None</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Application knowledge</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>Partial</td>
</tr>
<tr>
<td>Usability</td>
<td>partial</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
</tr>
<tr>
<td>QOS</td>
<td>partial</td>
<td>Full</td>
<td>Partial</td>
<td>Partial</td>
</tr>
<tr>
<td>Real-time support</td>
<td>Full</td>
<td>Full</td>
<td>None</td>
<td>Full</td>
</tr>
<tr>
<td>Security</td>
<td>None</td>
<td>Full</td>
<td>None</td>
<td>none</td>
</tr>
<tr>
<td>Context Aware</td>
<td>Full</td>
<td>Partial</td>
<td>None</td>
<td>Full</td>
</tr>
</tbody>
</table>

**Figure 9**: Middleware Features Table 1(d)
Wireless Sensor Network Setup

3 Investigation of Solution

As described before wireless sensor network built on very low power battery, low cost in term of hardware, and low processing capabilities, so WSN require solution which best fit to overcome these shortcomings. Below are few solutions which are related to the possible available solutions to the problems for WSN.

3.1 RPC (Remote Procedural Call)

This is a inter process communication between two programs. The Remote Procedural call works in a way that the remote server executes the procedure and end user (caller) get its result back. This type of implementation is called remote procedure call. In object oriented programmings RPC may be referred as “remote invocation and remote method invocation”.

This mechanism works with message passing, the client sends a message to known remote server to initiate a RPC, as a result server executes a procedure with parameters supplied by the client and then result returns back to the client. To transfer a parameter it require conversion of data into a stream of bytes is called “marshalling” at client end and conversion of stream of bytes into data is called “unmarshalling” at server ends.

3.2 Java RMI (Remote Method Invocation)

This is a java programming interface that works like RPC in object oriented environment where it enables programmers to build a distributed java base application. Where method of remote class is invoke by another java virtual machine. Java RMI uses serialization mechanism to marshal and unmarshal parameters. It works with client stub and server skeleton to perform the task.

![Figure 10: A typical implementation of java RMI](image-url)
Wireless Sensor Network Setup

The primary goal of java RMI is to create a distributed application with the same syntax used in non distributed application and how classes and objects works with multiple java virtual machines. However RMI is java dependent, to make this for non java CORBA (Common Object Request Broker Architecture) was introduced. [39]

![Figure 11: RMI classes](Image)

The architecture of RMI is based upon the behavior and implementation which runs on separate JVM from each other’s and allow the code that define behavior and implementation of that code. The implementation define the definition of a remote service in a class where and interface defines the behavior.

Java RMI is supported for both java standard edition and java micro edition, the RMI optional package (RMI OP) enables to write an application for embedded systems and devices. It is possible to write an application for small hand held and smart communicators. Java RMI OP is the subset of J2SE RMI and can be used with the devices which have support connected Device Configuration (CDC)
3.3 J2ME Polish Serialization

J2ME polish serialization is a framework which used to serialize the object easily by just implementing the interface of the framework. It uses either slim binary protocol or XML-RPC protocol (to achieve interoperability) to access the server functionality from a mobile application. It generates all the codes which need to read and write into the data stream and vice versa and uses key modifier “transient” to mark the fields which should not be serialized where as static fields cannot be serialized.

Implantation of this interface can be done in two ways

- Serializable
- Externalizable

They can be implement with just include the `de.enough.polish.io.Serializable` and `de.enough.polish.io.Externalizable`. Serialization can handle necessary conversion automatically where externalizable take the full control of the process.

3.4 KSN Serialization

Since Squawk VM on SunSPOTs does not support serialization, KSN serialization can be used in case where some complex or large object needed to be transferred over the network. SPOTs SDK does only support transmission of simple data types but not complex data types like double linked lists or trees. KSN serialization is a Java like serialization where it can be achieved by implementing required available interfaces. Along with serialization support of complex data types it takes care of following things that need to be considered for complex data structures.

References: references to other objects having complex data structures like graphs or trees.
List contents: lists that hold objects
Maintainable code: change in the implementation of structure
KSN serialization resolves all these issues.
Wireless Sensor Network Setup

4 Detailed Description of the Investigated Solution

4.1 Sun’s Small Programmable Object Technology (SunSPOT)

Sun Microsystems developed wireless sensor network (WSN) mote called SunSPOTs. SunSPOTs are built upon the IEEE 802.15.4 standard [41] these are MIDP [40] compliant devices which run Squawk Java Virtual Machine [42]

The SunSPOT Device is a small, wireless, battery powered experimental platform. It is programmed almost entirely in Java, to allow regular programmers to create projects that used to require specialized embedded system development skills. The hardware platform includes a range of on-board built-in sensors as well as the ability to interface to external devices. This device comes with three sensors namely: accelerometer, light and temperature sensor. MIDLets [43] are installed on these devices because of their MIDP compliant nature. There is a specific base station which is used to connect with the host computer and then through host applications remote SunSPOTs are used.

The physical arrangement of the basic connection arrangement between host computer and sensor nodes is showed in the figure below

The host application can be J2SE program but target sensor nodes application need to be J2ME MIDLet for the Squawk VM. There are two modes for base station to run, shared mode or dedicated mode. In shared mode, two JVMs are launched on the host computer: one manages the base-station and another runs the host application. In this model, the host application has its own system-generated address, distinct from that of the base-station. Base-station accepts the multicast request in this mode. The main advantage of shared mode is that more than one host application can use the same base-station simultaneously. On the other hand in dedicated mode, it runs within the same JVM as the host application and only host application can use it. In this model, the host’s address is that of the base-station.
Wireless Sensor Network Setup

Figure below illustrates the sunSPOT software stack that comes with the built in SDK.

![](image)

**Figure 12 : sunSPOT software stack [59]**

- **User MIDlets**
  An application framework built on java for MIDP and typically implemented on java enabled mobile or embedded device.

- **Transducerlib**
  A library provides access to spot hardware on Edemo sensor board, e.g. accelerometer, switches, LEDs, digital I/O, inputs.

- **Multihoplib**
  A library provides high level of protocols, e.g. Radiogram, Radiostream and manages routing of packets.

- **SPOTlib**
  Includes various SPOT libraries and provide access to low level MAC radio protocol and basic I/O.

- **Squawk VM**
  Is a java micro edition virtual machine runs on java enable device without support of operating system e.g. sunSPOTs
Wireless Sensor Network Setup

Figure below illustrates the software stack of the host for the sunSPOT SDK.

Figure 13: Host application software stack

- **User Application**
  Also called application software running on host PC design to perform specific task/ function for user or for any other application.

- **Spotclientlib**
  The library gives access to a number of OTA (over-the-air) commands that can be sent to a free-range SPOT. E.g. Hello command used to discover free range sunSPOTs.

- **Multihoplib**
  A library provides high level of protocols, e.g. Radiogram, Radiostream and manages routing of packets.
Wireless Sensor Network Setup

- **SPOTlib Host**
  This library provides basic I/O operations. These I/O operations are carried out by accessing Low-Level MAC radio protocol, which can be accessed either through USB connection or socket connection to communication with other host applications.

- **Rxtx library**
  Is a library is used to perform serial I/O function to the base station over the USB connection

- **Java SE JVM**
  It is a standard edition of java virtual machine running on a host PC.

- **Host OS**
  At the foundation the layer of host operating system is resides it can be Windows, MAC OS, Linux, Solaris.

### 4.1.1 Hardware Specs

- 180 Mhz 32-bit ARM920T (ARM9)
- 512K RAM
- 4M ROM
- 2.4 GHz IEEE 802.15.4 radio (TI CC2420)
- USB interface
- SPI/I2C interfaces
4.2 Karlsruhe Sensor Networking (KSN)

4.2.1 KSN Radio Stack
KSN radio stack provide protocol suite for communication. It’s a layered architecture written in java for sunSPOTs, each layer handles different aspect of wireless communication in sensor network. Similar to TCP/IP protocol suits implementation and other well known protocols. KSN radio stack provides SHP (Single Hop Protocol) layer, Compression layer, Routing Layer.

4.2.2 KSN OTA Management
KSN OTA (over the Air) management give ease in managing wireless sensor network, e.g.

- Resetting the node(s)
- Configure system properties on a specific node or to the complete network
- Deploying a new MIDlet on targeted node(s).

4.3 Alternative 1 (Command based with Datagram)

Figure 14 below depicts Alternative 1 where short commands are sent using datagram technique, because due to the very short nature of theses commands overhead of reliable communication cannot be compromised.

![Image: Sending commands to sunSPOTs Diagram]
4.4 Alternative 2 (Large objects deployment based with Datastream)

In Alternative 2 as shown in figure 15 below, a continuous stream of comparatively large object is transmitted through reliable stream based communication, because this large object is usually an application that needs to be installed over the target nodes. Any transmission error in this alternative cannot be compromised. To improve the transmission process here serialization and compression of data is used.

![Diagram of sending large objects to sunSPOTs]

Figure 15 : Sending large objects to sunSPOTs

4.5 Functional Description of Application

Following are the functional descriptions of the test scenario that has been used to implement and evaluate our both alternatives.

4.5.1 Demon mimicking

To satisfy the robustness and dynamicity in functionality, sensor nodes are put in mode of waiting for some action from the host. So whenever something comes from the host which can be a command with parameters or application to be installed in this case, relevant functionality is performed.
Wireless Sensor Network Setup

4.5.2 Receiver

Receiver thread after its proper initialization is used to receiver data which can be in the form of data gram or data stream.

4.5.3 Transmitter

Transmitter thread after its proper initialization is used to send data to the receivers in either data gram or data stream form.

4.5.4 Compression

In case of large object transmission alternative, data to be transferred in the form of stream is compressed using the KSN radio stack for efficient transmission.

4.5.5 Serialization

For efficient transmission of large object over the network serialization of data is always suitable, although its overhead is not compromising when the data to be transmitted is comparatively lesser. So in case of large object transmission KSN serialization is used because serialization is not yet supported in J2ME. The KSN serialization entirely written in java, provide interface to use serialization on sunSPOT without doing any modification to the VM.
5 Test with Analysis

5.1 Test Objectives

In dynamic environment wireless sensor network have to adapt different functionalities to cope with the versatile current situation. Quite often a task to be performed by a node or cluster of nodes is already there but it needs to be initiated with certain parameters according to the contextual situation, but there is also possibility of that those nodes are missing that particular functionality which is required to perform that required task. Very broadly speaking there are two possibilities that need to be performed to tackle these dynamic situations.

- Initiate (functionality) with certain parameters if it is already there (Alternative 1)
- Send (functionality) it if it is missing there (Alternative 2)

In certain cases there is option to choose explicitly among these alternatives, which one is more favourable in terms of time duration, complexity, processing power, dynamicity. The purpose of these tests and their analysis is to make this decision easy through the found results. In each case there will be two applications one running on host that controls sensor operations and the other one called MIDLet that is running on the sensor nodes.

5.2 Test Scenario and Basic Configurations

Following are the two proposed approaches that satisfy the above described objective.

5.2.1 Alternative 1: Initiation (of functionality) through short commands

In alternative 1 very short commands are sent to initiate an already deployed functionality on the target nodes, where a kind of demon is waiting for the command from a host to activate its functionality according to the obtained configuration parameters from host.

5.2.2 Alternative 2: Run-Time deployment (of functionality) through large object

For the second more dynamic and robust alternative large data is sent in the form of a serialized large object that includes the functionality to be executed on demand.
Wireless Sensor Network Setup

To compare both alternatives 4 tests has been performed. Two for the initiation (one for single hope and other for multi-hope) and two for deployment (one for single hope and other for muti-hope).

Note: (All tests perform same operation i.e. sampling of LIGHT sensor for a specific interval and for the specified number of samples. All the sunspots are configured to -30db radio power to build our required test-bed on table)

Following is the detailed description of these test scenarios:

1. Alternative 1 (Command based single hope initiation)

In this test short command(s) are sent to target sensor nodes to initiate the process which is already deployed on target nodes. Host application sends commands to target sensors through a base station. In this test base station directly communication with the target sensor nodes. This scheme employees short command for communication so datagram technique can be efficiently used in it, because overhead of reliability cannot be compromised due to very small size of commands. Host application sends a command with the required configuration which in our case is the number of samples of light sensed values and sample period. On the sensor side a demon like thread is working which triggers the activation of the process as soon as it gets command from host. It processes its functionality and returns back the results towards host.

2. Alternative 1 (Command based multi-hope initiation)

This test does have the same process as mentioned in single hope initiation except that now the communication is not directly from base station attached to host computer and sensor nodes. Instead all communication between them is through a middle sensor node which acts as a router.

3. Alternative 2 (Run-time deployment through single hope)

In this test a new functionality is transferred and installed on the target sensor nodes to perform a task. Since such transferred code is in the form of large objects so some measurements to handle its reliable transmission is required with some efficiency techniques
Wireless Sensor Network Setup

to make it adequate. This large object need to be serialized and compressed if possible and then transferred through reliable stream connection. Reliability in this case is not compromisable because what is being transferred is large and is the code which will be deployed on target nodes. In this case KSN radio stack is used instead of built-in radio stack of SunSPOT SDK. KSN radio stack provides compression and serialization of data being transferred through stream base reliable connection. First the image suite of the MIDLet to be deployed is transferred to the target nodes from the host after compressing and serializing this image object and then application service manager (ASM) residing on MIDP compliant sensor nodes, deploys it to the sensor nodes and resets it for new MIDLets execution. After the target nodes are ready same results are obtained that has been obtained in command based approach but in this case all communication is stream based with serialization and compression of data being transferred.

4. Alternative 2 (Run-time deployment through multi-hope)

This test is same as that of single hope deployment except that a middle node is used for communication between base station and target nodes. In this way in-between node(s) act as router for target nodes and host.
5.3 Application for Alternative 1

To implement both proposed alternatives different application for host and sensor nodes is required.

5.3.1 Host Application Process flow

Following flow chart depicts the flow of operations by the host application which is actually using base station to communication with the target sensor nodes.

![Flow Chart](image)

Figure 16: Host application process flow chart for alternative 1
5.3.2 SunSPOTs (Sensors) Application Process flow
Below is the flow chart of the application running on target sunSPOTS.

Figure 17: sunSPOT (sensor) application flow chart for Alternative 1
5.4 Application for Alternative 2
Similar to command based approach large object deployment based approach also have two applications one for host and other for sensor nodes.

5.4.1 Host Application Process

Figure 18: Host application Process flow for alternative 2
5.4.2 SunSPOTs (Sensors) Application Process

Figure 19: SunSPOT (sensor) application process flow for alternative 2
6 Results
This chapter describes the results obtained from the performed tests in terms of required test metrics.

6.1 Test Metrics
- Transmission time
- Memory
- Power (Battery)

6.1.1 Transmission time
It is a time consumed by a message to its destination after the first bit transmitted by the sender and the last bit received by the receiver. For this test test-bed has been made on the table with reduced transmitter power to -30db, so that functionality though router is achieved. So the distance in this experiment is not of much interest.

a) Alternative 1
Sending commands to sunSPOTs to start up the functionalities and acquired light sensor values with a sampling rate of 5 sec for each target node.

- Alternative 1 with single hop (initiation through command)
Wireless Sensor Network Setup

- Alternative 1: Multi hop (initiation through command)

![Graph showing Multi hop transmission time of alternative 1](image)

**Figure 21**: Multi hop transmission time of alternative 1

- Comparison between single hop and multi hop in Alternative 1

In this case multi-hop is extended with only one extra node between host and the target node, so it is a 2-hop communication. So from the graph below it is easily scene that extension of one hop does not affect much but of course its affect increases with the increase of number of hops. Here of course the distance also matters for the transmission time but since transmitter output power has been reduced to -30db to achieve out required test goals, so distance is set constant at this power.

![Graph showing Single hop, multi hop transmission time comparison of alternative 1](image)

**Figure 22**: Single hop, multi hop transmission time comparison of alternative 1
b) Alternative 2

Sending a large object to sunSPOT with new functionalities and acquired light sensor values with sampling rate of 5 sec. In this case the transmission and deployment cost of the new application is calculated separately. So following results are solely describes the transmission time after the new functionality has been deployed and here this communication is stream based.

- Alternative 2 with single Hop (deployment of large object by datastream)

Figure 23: Single hop transmission time of alternative 2

- Alternative 2 with Multi Hop (deployment of large object by datastream)

Figure 24: Multi hop transmission time of alternative 2
Wireless Sensor Network Setup

- Comparison between single Hop and multi hop of alternative 2

Here again multi-hop means 2 hop i.e. there is another node between the target node then the host for communication and off course target node is not in direct communication with the host but through the middle node that act as a router between them.

![Figure 25: Transmission time comparison between single hop and multi hop of alternative 2](diagram)

6.1.2 Memory Utilization

a) Memory utilization of alternative 1

Memory consumed by sunSPOT while sending short commands. The sunSPOTs have flash ROM size 4 MB i.e. 4MB = 4194304 Bytes

Memory occupied = Total Flash Memory size – Flash free memory

= 4194304 Bytes – 307936 Bytes

= 3886368 Bytes

```java
- do - host run:
  [java] [radio] Adding: Server on port 8
  [java] -30
  [java] [radio] Adding: Server on port 67
  [java] 22:13:57.129 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=387936
  [java] 22:13:58.432 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=385856
  [java] 22:14:02.138 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=382272
  [java] 22:14:03.432 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=385972
  [java] 22:14:07.131 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=396568
  [java] 22:14:08.433 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=386748
  [java] 22:14:10.132 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=390452
  [java] 22:14:12.432 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=387532
  [java] 22:14:18.435 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=324832
  [java] 22:14:23.140 SPOT: 0014.4F01.0000.0000 7124 Free Flash Memory=343012
```
Wireless Sensor Network Setup

b) Memory utilization of alternative 2

It’s a memory consumed by sunSPOT while sending a large object through serialization. The sunSPOTs have memory size 4 MB below is the figure showing its memory foot prints.

![Figure 26: Memory utilization of alternative 1](image)

![Figure 27: Memory utilization of alternative 2](image)
6.1.3 Power Consumption

Current drawn by the sunSPOTs for command based and large object based approach.

The unit of current is in milliAmpere (mA).

- Power consumption of alternative 1

![Graph showing power consumption](image)

Figure 28: Power drawn in milliAmpere (mA) of alternative 1
Wireless Sensor Network Setup

- Power consumption of alternative 2

![Graph showing power consumption of alternative 2]

Figure 29: Power drawn in milli Ampere (mA) of alternative 2
7 Conclusions and Suggested Future Work

7.1 Comparisons of obtained results between command based initiation approach and large object based deployment

7.1.1 Single/Multi hop transmission cost

Figure 30: Comparing single and multi-hop transmission time

Figure above compares both alternatives of simple command based and complex large object based run-time deployment and processing. In second case average cost of almost 24 seconds is required first to transmit the image to be deployed and then its installation and resetting the node for its operation. Besides this initial cost in the process of sampling and getting results still large object based robust alternative costs quite much then that of command based simple operation. Following are the factors which cause this variation:
Wireless Sensor Network Setup

a. Compression
b. Serialization
c. Reliable stream connection

7.1.2 Memory Utilization
In term of memory utilization stream based approach is better and consistent as compared to datagram based.

![Comparing memory utilization](image)

**Figure 31**: Comparing memory utilization

7.1.3 Power Consumption
In case of power consumption initially stream based approach takes quite a much current but once communication established it gets streamlined and even better then command based datagram approach.

![Comparing power consumption](image)

**Figure 32**: Comparing power consumption
7.2 Conclusions

- Diversity in situation and required functionality
  In order to achieve adaptability in dynamic contexts, overhead of run-time deployment based approach has to be tolerated. Although something in the middle of both extremes can be chose to reduce the overhead but only if application requirements leverage for.

- Compromise on large object stream based approach
  For time critical applications it cannot be relied on deployment based approach instead either command based or somewhat related approach need to be chose. But if application’s requirements cannot be filled with simple approach like command based then the overhead of dynamic approach need to be compromised.

- What is already deployed and what not
  In some cases decision about these both extreme alternatives can be achieved by visualizing the fact of whether required functionality is always there or need to be transferred.

7.3 Future Work

- Other possibilities to improve run-time large object deployment based alternative, i.e. efficiency in:
  a) Data compression
  b) Serialization
  c) Partial functionality deployment

- Parallel dynamic deployment
  Run-time Deployment method should be parallel for more than one target nodes.

- Large Scale implementation
  Implementation of dynamic run-time deployment of functionality over large scale.

- Intelligent application development, aware of all possible alternatives to improve WSN setup process
Wireless Sensor Network Setup

8 References

[21] Mobile Agent Middleware for Sensor Networks: An Application Case Study Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu Washington University in Saint Louis, Missouri 63130
[22] Wendi B. Heinzelman_y, Amy L. Murphy_z, Hervaldo S. Carvalho_, and Mark A. Perillo_y Middleware to Support Sensor Network Applications Center for Future Health University of Rochester Rochester, NY 14627
[23] Shuoqi Li, Ying Lin, Sang H. Son, John A. Stankovic, and Yuan Wei Department of Computer Science, Event Detection Services Using Data Service Middleware in Distributed Sensor Networks University of Virginia, USA
Wireless Sensor Network Setup

[25] Carlo Curino a Matteo Giani a Marco Gioretta a Alessandro Giusti a Amy L. Murphy b Gian Pietro Picco aDip. di Elettronica e Informazione, Politecnico di Milano, Italy bDept. of Informatics. Mobile Data Collection in Sensor Networks: The TinyLIME Middleware, University of Lugano, Switzerland


[31] Leon Evers, Maria Eva Lijding, Jan Kuper. Generic Multi–Packet Communication through Object Serialization. MidSens’08, December 1-5, 2008, Leuven, Belgium


[34] Pedro Javier Del Cid, Danny Hughes, Jö Ueyama, Sam Michiels, Wouter Joosen DARMA: Adaptable Service and Resource Management for Wireless Sensor Networks. MidSens’09, November 30 - December 4, 2009 Urbana Champaign, Illinois, USA


[42] Doug Simon, Cristina Cifuentes. The Squawk Virtual Machine: Java(TM) on the Bare Metal.


[47] Majid Bahrepour, Nirvana Meratnia, Paul Havinga AUTOMATIC FIRE DETECTION: A SURVEY FROM WIRELESS SENSOR NETWORK PERSPECTIVE. Pervasive Systems Group, University of Twente