Enabling Context Awareness in Ambient Environments using Cloud Infrastructures

Umar Zia
Abstract

This thesis was prepared in collaboration with Acreo (Research and Development Company) and OpenCare (IT Company). Its purpose was to design and develop a research testbed to enable context awareness in pervasive environments by modifying the MediaSense framework (EU Funded project) of Mid Sweden University. In ubiquitous environments, the proliferation of devices such as sensors, active badges, mobile phones, RFID and NfC tags enables the development of intelligent services towards new forms of pervasive applications. These intelligent services seamlessly gather context information and the benefits offered are in the provision of better services. The inspiration given by these intelligent services has meant that the focus of thesis has been on using these services in a healthcare application. The challenge is that the proposed testbed should address the intelligent delivery of health information to any host, anywhere, based on the user’s context. Further, context reasoning requires substantial computing power and smart devices have limited resources in terms of processing and memory, so, the testbed should enable smart communication to take place between these devices. The proposed solution satisfies the stated requirements by using a cloud infrastructure and a Distributed Context eXchange Protocol (DCXP). Any device on the internet that is DCXP capable may register with the architecture and share context information in an efficient way. In order to view context information, TV, smartphones, internet tablets and web interfaces have been provided for both the user and the health centre. By successfully addressing the requirements of the testbed, this thesis has enabled the creation of a pervasive healthcare application. Hence, this thesis concludes with the observation that the proposed approach for context awareness in a healthcare system has the ability to deal with the stated challenges.

Keywords: Mobile, Context awareness, DCXP, Sensors, Cloud infrastructure, Active badges, RFID, NFC.
Acknowledgements

First of all I am thankful to Allah the Almighty for His countless blessings and for giving me the knowledge to complete this thesis. During the preparation of thesis, I have received funding from Acreo and guidance from several people, who deserve to be acknowledged here.

I would like to thank my supervisor Stefan Forsström and my examiner Prof. Theo Kanter for their ideas, motivation and guidance. Special thanks go to Dr. Patrik Österberg, Dr. Ulf Jennehag, Jamie Walters, Victor Kardeby and Roger Norling from the MediaSense research group who shared valuable knowledge with me.

Working in Acreo has given me the opportunity to work with many talented people. Many thanks goes to my tutor Olle Johansson (Project Manager) for his endless support during the time period of the thesis preparation. I would also like to thank Hemamali Pathirana (Technician), Anders Berntson (Head of Department), Roland Elverljung (Research Assistant) and of whom have provided many interesting discussions during lunch and coffee breaks.

In OpenCare, I would like to express my sincere gratitude to Eva Lena Zackrisson (CEO), Lena Furubacke (Project Manager) and Johan Sundin (System Architect) for their valuable guidance about the testbed.

Lastly I would like to thank all my family and friends for their love and encouragement.
Table of Contents

Abstract .................................................................................................................. ii

Acknowledgements ................................................................................................. iii

Terminology .............................................................................................................. vi

1 Introduction ........................................................................................................... 1
  1.1 Background and Problem Motivation ................................................................. 2
  1.2 Overall Aim ......................................................................................................... 2
  1.3 Scope .................................................................................................................. 2
  1.4 Concrete and Verifiable Goals ......................................................................... 3
  1.5 Ambient Life Scenario ..................................................................................... 5
  1.6 Outline ............................................................................................................... 5
  1.7 Contributions ..................................................................................................... 6

2 Theory ................................................................................................................... 7
  2.1 Defining Context ............................................................................................... 7
  2.2 Context Awareness ........................................................................................... 7
  2.3 Context Modeling ............................................................................................. 8
  2.4 Wireless Sensor Network ............................................................................... 9
  2.5 Ubiquitous Computing .................................................................................... 10
  2.6 Ubiquitous Computing Technologies in Healthcare ....................................... 11
  2.6.1 Biosensors .................................................................................................. 11
  2.6.2 Radio Frequency Identifier ....................................................................... 11
  2.6.3 Near Field Communication ....................................................................... 12
  2.6.4 Android ..................................................................................................... 13
  2.7 Overview of Context-Aware Frameworks ....................................................... 14
  2.7.1 Context Managing Framework ................................................................... 14
  2.7.2 Context Toolkit .......................................................................................... 15
  2.7.3 CoBrA ......................................................................................................... 16
  2.7.4 Gaia ............................................................................................................ 17
  2.8 Related Networks Technologies ...................................................................... 18
  2.9 Cloud Computing ............................................................................................ 18
  2.9.1 Google App Engine ................................................................................... 20
  2.9.2 Bigtable ..................................................................................................... 21
  2.10 Related Technologies ..................................................................................... 22
# Table of Contents

3 **Methodology** ........................................................................................................... 24  
3.1.1 Requirements ........................................................................................................... 24  
3.1.2 Considered Solutions ............................................................................................... 25  
3.1.3 Test Model ............................................................................................................... 26  

4 **Design** ....................................................................................................................... 27  
4.1 Scenarios ....................................................................................................................... 27  
4.1.1 Applying Indoor Context Scenarios ......................................................................... 27  
4.1.2 Applying Outdoor Context Scenario ....................................................................... 30  
4.2 Sensor Integration using MediaSense Framework ....................................................... 32  
4.2.1 Design of Context Information Integration Ontology Model ................................... 32  
4.2.2 Persistent Storage for MediaSense Framework ......................................................... 33  
4.2.3 Estimating Sensor Ranking ....................................................................................... 35  
4.2.4 Associating User with Overlay .................................................................................. 36  
4.2.5 Connecting Devices to Overlay by using Smartphone ............................................. 37  

5 **Results** ....................................................................................................................... 39  
5.1.1 Acreo Lab Results .................................................................................................... 39  
5.1.2 Smartphone Interface ............................................................................................... 40  
5.1.3 Web Interface .......................................................................................................... 46  
5.1.4 Internet Tablet Interface ............................................................................................ 47  
5.2 Measurements and Comparisons .................................................................................. 48  

6 **Conclusions** ............................................................................................................... 52  
6.1 Future Work ................................................................................................................ 53  
6.2 Concluding Remarks ................................................................................................... 53  

References ......................................................................................................................... 54  
Appendix A: Contents of CD .............................................................................................. 59
Terminology

Acronyms and Abbreviations

3G            Third generation cellular wireless standards
4G            Fourth generation cellular wireless standards
AJAX          Asynchronous JavaScript and XML
API            Application Programming Interface
CoBrA         Context Broker Architecture
DAML          DARPA Agent Markup Language
DCXP          Distributed Context eXchange Protocol
DOM           Document Object Model
GFS           Google File System
GNU           General Public License
HDTV          High Definition Television
IaaS          Infrastructure as a Service
JDO           Java Data Objects
JPA           Java Persistence API
JSON          JavaScript Object Notation
JSP           JavaServer Pages
NDEF          NFC Data Exchange Format
NFC           Near Field Communication
ODBMS         Object Database Management System
ODF           Open Document Format
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OIL</td>
<td>Ontology Inference Layer or Language</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PARC</td>
<td>Palo Alto Research Center</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>SPARQL</td>
<td>Simple Protocol and RDF Query Language</td>
</tr>
<tr>
<td>UCI</td>
<td>Universal Context Identifier</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

This thesis communicates a new general approach to context aware services for ambient environments. Context aware computing enables us to use the features of mobile computing and pervasive applications in new scenarios. Today, it is possible to use Android based mobile phones to create context aware mobile applications that can utilize the sensors and mobility of the phone to create intelligence. Users can now share their locations, preferences, health profiles and temperature while it is possible for subscribers to be able to access this context. This information can then be used to acquire intelligence about each other in order to create better services.

In this research a cloud infrastructure has been introduced for the MediaSense Framework\(^1\) which is an EU funded project for enabling innovative services. It uses DCXP, which is an application level context exchange protocol in order to provide real time distribution of context and reliable communication among the nodes. Mobile clients that are DCXP capable have ability of connecting to a home location by short range mediums such as 802.11 or via long range wireless mediums such as 3G or 4G mobile networks. Using the MediaSense framework in the healthcare sector is carried out in collaboration with the following research partners.

**Acreo** is one of Europe’s top research companies providing ground breaking results within the fields of electronics, optics and communication technologies. They turn academic research into commercial applications and offer value added technology solutions for growth and competitiveness in industry and society.

**OpenCare** provides IT solutions for healthcare and are actively involved in both national and international development. They provide services in GPS tracking systems for Alzheimer’s patients, Vivago security watches and secure web portals.

---

1.1 **Background and Problem Motivation**

The motivation behind the thesis is to introduce cloud infrastructure as a backbone for the MediaSense framework. Current system is not capable of using cloud infrastructure and has scalability issues and thus this thesis will attempt to overcome such deficiencies for our research partners. After modifying, the primary focus is to use an updated framework in order to provide ubiquitous healthcare services. Furthermore, this thesis includes a study of the storage of context data and the problems which arise when storing continuously changing information, in relation to both the technical and privacy issues. For example, the database scaling based on the number of users, the battery performance of the mobile device based on the update frequency and privacy issues associated with storing personal context information.

1.2 **Overall Aim**

The aim of the project is to build a testbed for acquiring intelligence about the user’s health by using smart devices. Information gathered from a testbed will define the criteria for contextual updates, as users are always interested in having contextual knowledge which is specific to their health profile and preferences. When in a pharmacy or in a shopping mall, users are only interested in medicines or offers which are related to their particular health profile. Additionally, when in a hotel or when eating out they will only be interested in food based on the diet plans set by the healthcare system. Achievement of these scenarios is only possible when information gathered from different devices is context dependant and thus this thesis investigates context driven applications in ambient environments.

1.3 **Scope**

The scope of the thesis is quite wide as it has to deal with third party data, therefore a generic prototype will be presented. The programming language in the project will be Java for building an application on Android and on a Google app engine. Following units define the scope of project.

a) **Privacy**: Contextual information offers several types of benefits but gathering context data gives rise to many privacy issues. As users are always concerned with privacy issues, the prototype must provide
answers in relation to these issues by defining the privacy settings for the user.

d) **Scalability:** The prototype will be scalable in nature and will have the ability to deal with a dynamic number of users, services, storage and network growth.

b) **Responsiveness:** To measure the performance of any system, the most important factor is responsiveness. Users will be provided with information regarding services by means of a smartphone, web and TV connected with internet tablets via an HDMI port. The aim of project will be to use efficient parsing techniques in order to view healthcare services with the minimal delay.

c) **Mobility:** The system should be fault tolerant in nature i.e. in real time environments users should have the ability to move and become connected to the prototype even if there is a change to the network. It will not be necessary for users to restart the services, as our pervasive service will easily adapt to that network and start sending updates.

e) **Usability:** The prototype will be very easy to install, even for new users, and will have the ability to interact with different types of devices. There have been a number of case studies conducted for building the interface interactive for different devices.

1.4 **Concrete and Verifiable Goals**

The overall goals set for this research work are separated into “study work” and “implementation”.

**Study work:**

The study work involves the following steps:

A) Modifying the MediaSense Framework and using it to provide ubiquitous healthcare services.

B) Studying existing context-aware frameworks and different related technologies in order to provide a better understanding of a testbed to be built.

C) Evaluating smart devices in different indoor and outdoor scenarios.
D) Proposing a generic prototype for context aggregation and dissemination.

Implementation:

In order to build the prototype, the following requirements must be implemented:

A) Providing contextual information on the TV so that the users have the ability to view information on a large screen.

B) Building an Android application for gathering context information from sensors, tags and providing services with the following functionalities.

1. Menu screen for navigation.
2. Settings.
3. Auto recommendation system on the basis of sensors and tags.
4. Smart reservation system.
5. Health graphs.
6. Diet plans according to the health profile.
7. Availability of services on the basis of the health context.
8. Using tags for the healthcare services.
9. Privacy control.
10. Location awareness.
11. Emergency calling system, guide and feedback.

C) Building web interfaces for user and health centre to view and manage stored context information.

D) Performing an evaluation of the application based on the update frequency and available radio connectivity.
1.5 Ambient Life Scenario

The area of ambient intelligence is becoming more prevalent in everyday life by enabling the system to adapt according to the user’s context. This thesis must examine both indoor and outdoor scenarios for context awareness in ambient environments which should lead to interesting conclusions. Several places such as hospitals, offices, shopping malls and the home which have devices such as smartphones, NFC tags, passive RFID and biosensors have to be taken into consideration.

Consider a scenario where a person living in an urban environment has connected his/her sensors with the health centre and with other sources of information. When s/he wakes up in the morning his/her smartphone suggests a menu for breakfast. After breakfast s/he receives information concerning meetings at the office and the traffic information indicates that it is necessary to leave earlier than usual and to take a jacket because it is cold outside. While driving, a notification is given about the shortest possible route to reach his/her office. In the office s/he is alerted by a reminder about pending work and after some time, due to the workload, wearable health sensors indicate that s/he needs to see the doctor, depending on availability and time. After visiting the doctor his/her smartphone retrieves more information about the medications and diet plans.

When traveling back to home, s/he receives the location information regarding a pharmacy on the way with availability of the medication prescribed by the doctor. After leaving the pharmacy, s/he is informed about an offer regarding food matching his/her diet plans from the shop next to the pharmacy. On arrival at home the plans for the following day are available. This scenario provided the inspiration to determine a solution for the acquisition and provisioning of the context from these kinds of environments.

1.6 Outline

Chapter 1 describes the background and problem motivations behind the project. Chapter 2 provides the theory and related work. Chapter 3 describes the approach to the problem and how the results are to be evaluated. Chapter 4 describes the implementation of the solution. Chapter 5 contains the results. Chapter 6 will present the conclusions and possible future work. This is followed by the references used and the attached appendices.
1.7 Contributions

Although the project has been conducted without any direct contributors, credit must be given to the MediaSense research group, Acreo and OpenCare for their previous work. Credit must also go to STC@MIUN for the sensor platform which has been used in the project.
2 Theory

This chapter presents the theoretical background in order to gain a better understanding of the thesis. It has been gathered from different sources, such as research papers, books, internet sites and RFC’s.

2.1 Defining Context

The word context can generally be defined as a situation or surroundings of an entity. For example this could be in relation to a spoken sentence, whose context can be defined by looking at the situation in which it is spoken. A great deal of research has been previously conducted for understanding the concept of context. The earliest well known research was conducted by Brown [1]. Shilit [2] had also explained interesting concepts for exploring the area of context. Dey and Abowd [3] refined their definition in stages and finally arrived at “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”. There have been more recent studies into context such as in [4], which surveys the area of context awareness in detail. Context can be gathered manually by entering a user’s preferences or in an automated manner by using a sensing system. As this research is dealing with context awareness in healthcare, so research into smart devices is of significant importance.

2.2 Context Awareness

Context-awareness can be defined as “A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the users task” [3]. In ubiquitous computing, pervasive access to information is gaining prominence. This distinction is made through the use of smart devices such as smartphones which can gather information. A proliferation of smartphones in the everyday environment, means that contextual information is available everywhere irrespective of location. This ultimate availability of information has set innovative standards for the smart services.
The domain of context awareness in the healthcare can be discovered in many situations. Examples include smart homecare where biosensors play their role for context awareness. A person with sleep apnea can be monitored by means of wearable sensors. They will provide help in monitoring blood oxygenation, breathing and heart rate. After collecting contextual knowledge there are techniques such as “Journaling support” [5] which can be used for patient’s awareness. A journal is available for the patient by means of a smartphone or computer, which is based on the patient’s motion, heart rate and sleep.

Furthermore, examples of the context awareness can be found at airports, offices, and shopping malls etc, where interactive services can be provided depending upon the location. Users at an airport can be provided with offers regarding return tickets. Home users can be informed about security, the status of appliances and environmental conditions. For offices, users can be provided with information about meetings and the location of valuable equipment. Users in a shopping mall can be informed about discount offers on different products according to the user’s interest.

2.3 Context Modeling

After acquiring the contextual knowledge from different intelligent devices, it is possible to further represent and store it in different forms. Design is generally dependent on the system requirements. The following are some common methods, which are used for context modelling.

Object-Oriented Model: “These models are based on object oriented paradigm to face the representation of the dynamic characteristics of context”[6]. There are major advantages of using this model including polymorphism, encapsulation and inheritance.

Key-Value Model: In this model the context information is represented in key-value pairs and is widely used in distributed services frameworks because of its simplicity. Schilit et al. [2] also used this key-value pairs for context modelling.

Logic Based Model: This model has a powerful reasoning mechanism based on facts, expressions and rules for defining context. It manages the terms and allows for the addition, removal and updating of new facts [7]. Furthermore it also addressed the information quality but it has limitations regarding model merging.
Ontology Based Model: “Ontologies are a promising instrument to specify concepts and interrelations” [8,9]. These models define concepts and the relationships between them. It can be implemented by using the Web Ontology Language (OWL).

Graphical Models: In these models Unified Modeling Language (UML) and Object-Role Modeling (ORM) are used to represent context information.

Markup Schema Model: In these models there is a hierarchical structure composed of tags and fields. Fields can further contain tags and corresponding fields. These models are basically used for modelling profiles [10].

2.4 Wireless Sensor Network

These networks are responsible for sensing the environment as well as processing the information. This contextual information, such as time, people, location and activity can be used to provide better services for the users. Sensor nodes, which communicate within short distances, are the basic devices which give their services to these networks for information gathering, monitoring and decision-making [11]. An overview of wireless sensor network is visualized in figure 2.1.

Figure 2.1: Overview of wireless sensor network [12]

These low power and low cost networks can assist in building military tactics, home networking, environmental monitoring systems and ubiquitous healthcare systems [13, 14]. Medical systems based on sensor networks are capable of monitoring patients in a real time environment. Doctors, nurses and caregivers, who are connected to these healthcare
sensor networks, are fully aware of the patient’s health status such as blood pressure and heart rate [15] etc.

2.5 Ubiquitous Computing

Ubiquitous computing is also known as pervasive computing and was first described by Mark Weiser [16] who was a chief scientist at Xerox PARC. He provided the idea of an “invisible” computing which is available everywhere. In this context “Ubiquitous” meant not merely “in every place,” but also “in every thing”[17].

Common objects, from a cup of coffee to the art on the walls, can be taken into account for sensing as well as for processing of information and consequently many interesting new results can be achieved. Developments in this field are very significant and have clear implications for adopting the ubiquitous model. The two dimensional barcodes and Radio-frequency identification tags (RFID tags) are examples of the technologies that are adopted from novel applications and have been utilized in the scenarios for ubiquitous computing.

It is possible to consider all objects in the form of processed information embedded in different devices. Many of these objects have already been incorporated into different applications and cameras and smartphones provide the most obvious examples. These technologies have made an invaluable contribution to the realization of the “Internet of things”. In the near future the everyday objects will be transformed into “ubicomp” technologies such as RFID, GPS and so on. These embedded systems will finally provide the bridge to the broader networks [17].

Active badges [18] by Roy is one of the most significant advancement in the “ubicomp” field, and by wearing these Active badges, workers can automatically unlock those areas to which they have been granted the relevant rights. These badges transmit signals which provide information about their location. Ubiquitous computing also plays an important role in the field of healthcare. There are real time monitoring systems [15] for the patient’s health status, which provide reliable and fast medical services. These real time monitoring systems are based on low cost wireless sensor networks, discussed in Section 2.4.
2.6 Ubiquitous Computing Technologies in Healthcare

In ubiquitous computing there are many technologies for context gathering including sensor networks, smartphones, NFC (Near field communication) and RFIDs (Radio Frequency Identification) etc. Using services from these technologies can assist in building pervasive applications. Details for some of these technologies are discussed below.

2.6.1 Biosensors

In the ubiquitous healthcare field, an advancement in biosensor design has been the main factor for enabling context awareness in healthcare. An investigation into context-aware services in healthcare will show that biosensors are the building blocks for all sorts of context suppliers and are the most promising devices for making intelligent systems in order to monitor health. Biosensors have bio recognition systems based on receptors and detectors which are used for converting biological responses to electrical signals. Receptors deal with selectivity whereas the detectors act as a transducer for translating physical changes [19].

These biosensors can be attached to the patient’s body with the assistance of smart clothing [20], which then defines the contextual state of the user. Furthermore, these wearable biosensors can also be attached to jewelry [21] and wristwatches [22] etc. Before the existence of the biosensors, it was essential for a patient to visit the doctor even for a minor checkup which was a somewhat hectic task, but now many companies are providing low cost medical sensors which enable the monitoring of patients at home.

2.6.2 Radio Frequency Identifier

Radio frequency identifier (RFID) is a diverse cutting edge technology because of its efficiency, cost, fast processing and global identification of the object by means of embedded tags. These tags are gaining widespread attention in context-aware systems for gathering context, shipping areas for inventory control and in the healthcare sector for monitoring the health of patients. Before the invention of these tags, critical care patient context was monitored by bedside monitoring devices, which involved extensive wiring. Managing these wires was a major challenge for both the patient and the caregiver in terms of patient comfort and mobility. When a patient had to move from one room to another all the wires were required to be stretched out. It is now the case that the majority of such patient monitoring is by means of wristbands.
which contain a RFID tag. These tags are then tracked by the readers installed in hospitals for monitoring and administrative tasks [23].

A thin medical patch, has been invented in the Georgia Institute of Technology, which is equipped with an ultrahigh-frequency (UHF) active RFID tag and a sensor for monitoring the health of the patients. The patch is known as the Prototype of the Integrated RFID enabled Agile Sensor Lab (PIREAS) [24]. Latest version of the patch includes a temperature sensor which can be fixed in a bandage. Furthermore, research is on-going for other sensors, which can assist electrocardiograph (ECG) and heart’s electrical pulses.

There are also RFID tags together with a GPS for long tracking, which are called Hybrid Tags. For building these types of tags, Numerex and Savi have introduced intelligent tags that combine GPS, RFID, and satellite communication. These tags are able to keep track of the objects irrespective of location. Overall these tags can be divided into two types, namely active and passive tags. Passive tags have no direct energizing source and are dependent upon the energy of the interrogating device. In contrast to this, active tags have an energy source which makes it possible to extend their range. [25][26].

2.6.3 Near Field Communication

Near filed communication (NFC) tags can also store contextual information, which can be used for context exchange between devices which are in close proximity. They can be defined as “NFC allows you to share small payloads of data between an NFC tag and an Android-powered device, or between two Android-powered devices [27].” It has two main operational modes, namely active and passive. In the passive mode the device starts the communication and produces a carrier field while in the active mode both appliances generate an RF field. These NFC enabled devices also have Reader/Writer, Card Emulation and Peer-to-Peer modes.

Data in NFC tags can be stored in different formats. The majority of the Android framework APIs use the NFC Forum standard, which is known as the NFC Data Exchange Format (NDEF). The main advantage of using NDEF is that it assists in encapsulating data and the android has a parsing support for it. Additionally, if the tag does not contain this data format, a manual read/write has to be performed in the form of raw bytes.
The use of NFC tags has become common in healthcare systems. Caregivers often have heavy workloads, which may contribute to human error such as giving an incorrect dose of medicine and by using these tags it is possible to reduce these errors [28]. Other reason for using NFC tags includes the assistance it provides for Alzheimer patients [29] and in monitoring heart failure [30]. These tags can be used at a variety of places such as smart posters, videos, doors, bottles or any kinds of display where a small amount of information is to be stored. Furthermore, they can also be used to store the information for getting access to a building instead of using an access card.

2.6.4 Android

During the past few years, smartphones have played a vital role in context aware systems and have revolutionized the way in which mobiles are used. Before the emergence of these smartphones, mobiles were limited only to voice communications [31]. Smartphones have camera, Wi-Fi, accelerometer, compass, sensors for location awareness and for detecting nearby devices etc. They have operating systems such as Apple's iOS, Nokia's Symbian, RIM's BlackBerry OS, and embedded Linux distributions such as Maemo [32]. Android is a mobile operating system from Google which enables the users to install different services or to build their application by using different APIs. It has the following architecture [33].

![Android architecture](image)

**Applications**: The applications in Android are Java based and include maps, contacts, calendar, email client and sms program etc.

**Application Framework**: Android offers an open development platform which enables developers to access hardware, running background services and for gathering location information etc.
**Libraries:** It has C/C++ libraries and developers have the ability to build application using these libraries.

**Android Runtime** The applications build for android have their own process and instance of Dalvik Virtual Machine, which is supported by Linux kernel.

**Linux Kernel** It is responsible for drivers, memory management, network stack and provides bridging between hardware and software.

### 2.7 Overview of Context-Aware Frameworks

There are several existing frameworks for context awareness which required to be highlighted so as to provide a better understating of context aware system. These frameworks have different architectures for the gathering and representation of the context, which are explained below.

#### 2.7.1 Context Managing Framework

Context managing framework has the ability to cope up with different types of context information and can be customized for any specific scenario. In addition to customization and context retrieval it facilitates the simple registration of newly distributed heterogeneous data sources [7]. It is based on several modules, including context manager, resource servers, context recognition services, change detection service and security.

![Figure 2.3: Context managing framework](image)

Context manager, as shown in figure 2.3 serves as a central server for storing context information and communicating with other servers. The resource server is responsible for transforming the sensor’s data from a...
high frequency to a low frequency level i.e. low level context, and posting it in the blackboard (data centric view). Low level context has four further phases which include measurement, pre-processing, feature extraction and the last phase includes the quantization and sampling phases. The processing of low level context is performed by the recognition service in order to obtain more meaningful high level context. In addition to these services it also provides change detection and security services [34].

2.7.2 Context Toolkit
Context Toolkit, illustrated in figure 2.4 is an approach for building context-aware applications. It has a widget architecture style in which there are widgets (software components) and a distributing infrastructure to host them. These widgets are used for context sensing by keeping the details hidden [7].

The Enactor framework can generate eight types of explanations (Inputs, Output, Certainty, What, Why, WhyNot, How To, What If). The services given by this toolkit cover the infrastructure for sharing the context data, network API for accessing the context data, encapsulation of the sensors, interpreters for the abstraction of the context data, context data storage as well as history and privacy protection [35].
2.7.3 **CoBrA**

“Context Broker Architecture (CoBrA), is an agent based architecture for supporting context-aware systems in smart spaces (e.g. intelligent meeting rooms, smart homes, and smart vehicles)”\[36\]. It assists in building distributed applications with an object oriented environment, in which objects are used to invoke methods between the client and server through an object request broker. CoBrA includes a policy language \[36\] which enables the users to define their contextual knowledge. The broker uses this defined contextual knowledge in order to provide an appropriate notification to the users. Cobra is mainly based upon a semantic web, as illustrated in figure 2.5.

![Figure 2.5: CoBrA](image)

**Semantic Web:** Semantics can be defined as the meanings behind what we say. It enables systems to form a data store, create vocabulary, identify words and an association between them. It is possible to describe a semantic web as a "web of data" that facilitates machines to understand the semantics, or meaning, of information on the World Wide Web \[37\]. In CoBrA, context ontology is represented by semantic web languages such as the web ontology languages OWL and RDF. The components of a semantic web are as follows.

**Ontology:** “Ontology specifies a rich description of the terminology, concepts and properties explicitly defining concepts” \[38\]. Contextual data coming
from the sensors can be structured in a useful way by using ontology. Agents can then use this ontology for context acquisition and provision.

**RDF:** (Resource Description Framework) is a language for describing web resources, data models and for the linking of data whereas the RDF schema is used for elaborating the properties of these resources. The resources may include content, patent information, author and modification. Furthermore, N3, XML, RDFa are also used to represent the data model [37].

**OWL Web Ontology Language:** It is derived from DAML (DARPA Agent Markup Language) and OIL (Ontology Inference Layer or Language). OWL has been placed on top of RDF because it contributes its vast vocabulary. The vocabulary contains classes and their association with other classes based upon cardinality, equality and symmetry etc [39].

**SPARQL:** (Simple Protocol and RDF Query Language) is the recommended querying language for the semantic web.

### 2.7.4 Gaia

Gaia infrastructure has extended operating system concepts to elaborate context awareness. It provides the services of using existing resources [7]. Interfaces are available to build the portable applications which can run in arbitrary active spaces. An overview of the current system has been illustrated in figure 2.6.

![Active Space Application](image)

**Figure 2.6:** Gaia

For distributed computing, CORBA is used which performs different task by means of the availability of different agents. These tasks include context-sensing, event distribution and discovery etc. Agents are further connected with devices which exist in the environment. Users are also
assigned with agents which can provide a proxy and personal information storage. There are also application agents that assist users in performing various tasks such as the playing of music and a power point application [40].

2.8 Related Networks Technologies

Having a suitable network for both indoor and outdoor context acquisition is very important for precise context awareness. This thesis will use some network technologies which are briefly explained. IEEE 802.15.4 standard is for cost effective and low-rate wireless personal area networks and is suitable for use in monitoring applications. The implementation of the standard in ZigBee uses the Carrier Sense Multiple for channel access and offers 250kbps with an excellent battery life which ranges from months to years. Its name has been derived from the zigzag flying of bees that forms a mesh network among flowers [41] and supports cluster, tree, star and mesh topologies. Bluetooth and Wi-Fi also operate in the same frequency band as ZigBee which is 2.4 GHz but they consume a higher power. For the outdoor context scenarios 3G has been used, which is the third generation wireless technology and provides global roaming.

2.9 Cloud Computing

Cloud computing is an emerging field for the distributive computing paradigm and an awareness of cloud based approaches for context information provisioning is rapidly increasing. It is becoming another key technology after Web 2.0. This infrastructure provides innovative services to context aware systems including computing power, networking, data storage, collaboration tools and massive scalability by providing the resources on demand. Instead of using a remote server, the jobs are assigned to a huge number of distributed computers and all the resources are made available in a virtual pool. Users are provided with new features by abstracting the configuration so that the end user does not have to be concerned about the physical location and can enjoy different services with the allocation or reallocation of the computing resources [42].

Cloud computing offers drastic reduction of IT costs by utilizing its cornerstone characteristic of virtualization. The support for virtualization is available at software, hardware, operating system and many IT resources levels. It changes the traditional computing paradigm by its
resource deployment and its utilization. Users are provided with a pool of resources including servers which can act as multiple virtual servers, thus enabling room and costs to be saved for a new server. Furthermore, API provisioning enables the users to develop their applications. Service delivery by cloud infrastructures is at various levels and these levels are categorized into IaaS (Infrastructure as a service), PaaS (Platform as a service) and SaaS (software as a service).

![Cloud computing model](image)

Software delivery or software on demand is a thriving approach and it eliminates the intricacies associated with installation. End user is provided with global access, maintainability and thus causes no apprehension about buying or licensing the software. On the other hand, providers install only single software which is used by multiple organizations. Commercial services examples include saleforce.com, email cloud and it can be of any type such as CRM, finical planning or even word processing, which is named as SaaS Software, as a service [43].

Middle layer or PaaS [43] involves platform provisioning which facilitates all aspects from the building, integration, persistent storage and deployment of the application. In addition to these features, security and scalability are core traits of PaaS and are available from Google App Engine, Azure, Force.com and Engine yard.

Delivery of the infrastructure for processing and storage is provided by IaaS. These packages offer memory, CPU and networking based upon the user’s choice with instant scalability features [43]. In addition to these packages it includes security, load balancing and full OS access.
Reliability of the packages is ensured by many providers including Amazon, GoGrid, Rackspace and Linode.

2.9.1 Google App Engine

Google App Engine is built on top of the Google scale and is a platform as a service technology. It facilitates the services for the users to design, build, test and deploy their application with the support of several programming languages on the Google infrastructure. The applications, which are developed, can be scaled up in terms of storage and processing according to the demands of the user. The billing model is pay as you use. The high level overview [44] diagram is given below.

**Figure 2.8: Google App Engine**

**Cache Service:** For performance reasons, Google App Engine uses memcache for temporary data to which different instances can point. Retrieving data from memcache ultimately results in high speed access.

**Mail Service:** It provides the email services for the application by using Java Mail API and Google accounts.

**URL Fetch Service:** This service is used by the applications for fetching the resources over the Internet. It also enables communication with other hosts by using HTTP and HTTPS, whereas the Secure Data connector (SDC) provides encryption for the data.
**Image Manipulation Service:** It provides services to perform various operations on the image such as cropping, rotating and resizing.

**Task Scheduling (Cron) Service:** This service is used for scheduling tasks for the application.

**Persisting Data:** Java Data Objects (JDO) and Java Persistence API (JPA) are used for accessing the data from the data store.

### 2.9.2 Bigtable

App Engine datastore is powered by Google’s Bigtable which has a distributed storage system for managing structured data. It uses distributed Google File System (GFS) and can be defined as “*Bigtable is designed to scale to a very large size: petabytes of data across thousands of commodity servers*” [45]. Due to its dynamic control over the data layout, it can be a good choice for the storage of the context data. The majority of Google’s projects store data in the Bigtable, for example, Google Earth and Google Finance etc. It has a multi-dimensional sparse map. Indexing is performed by a row key, column key, and a timestamp for each cell which assists in maintaining multiple copies and versions of the same data [45].

![Google’s Bigtable data model](image)

In this table, illustrated in figure 2.9, row keys are arbitrary strings, in which each and every write or read operation of data is performed under an atomic single row key. It maintains the data in the order of a lexicographic row key. On the other hand, column keys are grouped into sets which are called column families. This column family is mandatory before storing the data under any column key in that family. API of the Bigtable offers several functions including creations, deletion, cluster changing and access control rights etc. The client application can easily write and delete data in the Bigtable by iterating individual rows or subsets of the data. Bigtable also supports client scripts making it easier for the users to run their custom scripts [45].
2.10 Related Technologies

There are several technologies which have been taken into consideration while building this project. A selection is provided below.

**REST Style Web Services** offers a solution by providing a uniform resource identifier (URI) for the resource and stands for representational state transfer. It is possible to perform different operations on these resources which include GET, POST, PUT, and DELETE. Ubiquitous environments are also resource oriented like REST. For these environments context information acts as a resource [46] which can be wrapped with REST style web services. After wrapping the resources they can be accessed by context-aware applications through predefined operations.

**JAVA** is a high level programming language developed by James Gosling at Sun Microsystems. It is platform independent and is used to build large scale application programs which include interactive GUI, image processing, dynamic web applications, network programming and robotics.

**JSP** stand for JavaServer Pages which provides a fast means of creating dynamic web content. The introduction of JavaServer Pages Standard Tag Library (JSTL) has further extended the specification of JSP.

**AJAX** stands for Asynchronous JavaScript and XML which is another technique for creating fast and dynamic web pages. It enables divs or different parts of a web page to update them self asynchronously by exchanging small amounts of data with the server, and thus there is no requirement to reload the whole page.

**MySQL** is a relational database commonly used all over the world for different applications such as e-commerce, data warehousing and portals. It is open source which is available under the terms of the GNU General Public License.

**XML** stands for eXtensible Markup Language which is a set of rules used for encoding documents in machine-readable form. It stores the data in the form of tags. According to [47] its structure is bloated and its objects are analyzed as DOMs (Document Object Mode), which takes a long time as compared to JSON.

**JSON** stands for JavaScript Object Notation which is a lightweight data exchange format. Mostly it is used for serializing and transmitting
structured data over a network connection. It uses a text format which is independent of languages. For machine it is very easy to parse and generate JSON as compared to XML, therefore it is much faster than XML [47].

**DataNucleus** “The DataNucleus project provides products for the management of application data in a Java environment” [48]. Google App Engine also uses datanucleus-plugin for providing Java persistence JDO/JPA for Bigtable.

![DataNucleus Diagram](image)

**Figure 2.10: DataNucleus [48]**

Furthermore, it has implementations as illustrated in figure 2.10 for RDBMS, JSON, XML, ODF, HBase, Amazon S3, Excel, LDAP and ODBMS. There is also availability of the non standard JDO query syntax [48].
3 Methodology

This chapter describes the adopted methodology for the project. As the sensor integration portion was based upon the MediaSense framework, the first step thus involved a study of the existing prototype and investigating the possibilities as to how this framework could be extended for enabling context awareness in healthcare. After studying the existing prototype, a solution was required to be presented based on the following requirements.

3.1.1 Requirements

To answer the questions stated in the problem statement, there are some requirements which should be fulfilled by the proposed solution. They are as follows.

1. Designing different scenarios of healthcare for context gathering. The evaluation performed on the scenario will further assist that how frequently the prototype can be updated concerning the context of the user.

2. Implementing a persistent storage infrastructure for the distributed overlay and demonstrating its use for different sensor devices which include Zigbee, Bluetooth and Ethernet sensor module.

3. Context reasoning requires substantial computing power and smartphones have limited resources in terms of CPU and memory. Furthermore, knowledge management on a smartphone is the key factor for consuming battery power so rather than making smartphones as the information base, Google’s Bigtable will be used for maintaining the context of the user.

4. Exposing context information according to the user’s choice, as privacy always remains the main feature. Therefore the prototype should have an option in which users have the ability to control their personal information. By enabling these controls it will affect the location based advertisements but it will satisfy the stated requirement.
3.1.2 Considered Solutions

The solution for the research problem can be considered by designing different indoor and outdoor context scenarios. Entities in these scenarios will interact with different devices including tags, sensors, smartphones and internet tablets which will be connected to a distributed overlay for sharing context information and receiving automated recommendations according to the context provided. For the implementation of these scenarios in different environments, the existing MediaSense framework will be updated as there are certain limitations within the framework.

During the pilot project of the MediaSense framework, a standard relational MySQL database was used for the storage of context data. Using this database gives rise to certain issues such as the performance, single point failure and scalability. In such kinds of SQL databases, it is practically impossible to perform operations and searches which are dependent on the massive context information. Although it can perform some of the context dependent queries, it then becomes very complicated to handle. For information that is dependent on the context, a more scalable database is required.

Furthermore, the current design of the distributed overlay, in addition to its use of the MySQL database, also requires each device to store some context data, which was affecting the performance of the devices. Therefore, this system can be extended and instead of storing the context data on the device, the data can be stored in a context repository for performing different operations on the data.

Therefore, the focus is to create and verify the API which contains the functionalities of connecting distributed overlay to the Google app engine. After connecting to the Google App Engine, Bigtable will be used for storing and retrieving the context information. Using Bigtable as a context repository will improve the performance of the overlay as well as the devices.

Third party databases will be used for acquiring intelligence about the user, whereas DCXP signaling in the framework will provide the services in order to rank the sensors.
3.1.3 **Test Model**

A test model has been formulated, in order to satisfy the requirements that have been set for the prototype. The following measurements should be taken into account.

1. **Overlay Registration**: It is the time required by the prototype for registering a device with the proxy.

2. **Context Data Registration**: The time required to identify and resolve a new context in the prototype.

3. **Message Latency**: After joining the overlay the message latency is the time to send and receive a message from one device to another device.

4. **Responsiveness**: It is the time which the infrastructure consumes for receiving a message and to start processing its contents.

5. **Parsing Time**: The processing time consumed by the smartphone for xml or JSON updates. This will assist for adopting the parsing technique for the prototype.
4 Design

This chapter begins with an explanation of the scenarios designed for context gathering followed by a description of the sensor integration achieved by means of the modifications to the MediaSense framework.

4.1 Scenarios

Scenarios have been designed in order to satisfy the requirement described in Section 3.1.1 (requirement no 1). There can be several places including homes, offices and hospitals for context aggregation and dissemination, so the classification has been divided into indoor and outdoor context scenarios.

4.1.1 Applying Indoor Context Scenarios

An indoor context scenario has been proposed for the health center and the home which is based on the requirement of the prototype.

Health Center Scenario: The scenario for the health center has been proposed in figure 4.1, by using RFID Tags as they offer low cost solutions.

![Health center scenario](image)

Figure 4.1: Health center scenario
The assumption is that the hospital will be equipped with RFID readers at different positions including wards, doors, hospital entrance, exits, pharmacy, blood bank and operating theatres. Furthermore, it is assumed that smart RFID [49] tags exist which can be combined with the sensing system. These intelligent tags are capable of sensing, monitoring and adapting the changes in the environment. According to EPCglobal Class-1 standards these novel RFID tags can store vital biological information such as body temperature, blood pressure and heartbeat etc.

When a patient arrives at the hospital, the patient will be given a wristband containing an RFID tag. It will provide the patient with a unique identity. This identity can be associated with the patient’s social security number to enable the retrieval and maintenance of the previous medical histories of the patient. Health contextual values are then sent to the context server which will have servlets for posting the data to an App Engine for job processing and Bigtable will play the role of the context repository for these contextual values.

The implementation of the RFID tags will assist in building the prototype in several other ways. Restricted areas can be defined for the patient by setting the tracking alarm on the entrance and exit or at any place. By defining these restricted areas patient are unable to leave the hospital without permission. Furthermore, these tags can maintain the user’s medical profile during his/her treatments in the hospital by means of a unique identification. Maintaining these medical profiles with a unique identification can be helpful for setting different kinds of alerts. For example, alerts can be set if the wrong medicine or blood bag for blood transfusion enters into the patient’s room.

Tagging will not only assist in monitoring the patient but will also provide support for maintaining hospital assets, pharmacy and blood bank etc. Blood bags and medicines can be tagged which will assist in determining whether the stock is running out. High value equipment can also be tagged in order to track its location. The administrative staff can keep the track of the bills relating to the patients. After leaving the hospital, patients will have prescriptions, diet plans, bills, caregiver’s visit plans and different health services on their smartphone, web and TV. By considering the above proposed solution it becomes possible to design intelligent hospitals for monitoring patients.
**Home Scenario:** When the patient returns home they will be provided with NFC tags. These are an extended and secure form of the RFID and can also be used in hospitals but they only have a very short range. At home these tags can help to monitor the health of the patient and those services which are provided by the caregiver. Installation of biosensors and a video camera can further assist the caregiver to monitor the patient’s movements.

![Figure 4.2: Home scenario](image)

Figure 4.2 illustrates the scenario when a caregiver is visiting the patient’s home and having a smartphone such as the Google nexus S which has the ability to scan the NFC tag. After providing different services (*medication, checkups, exercises and cleaning*) to the patient, the caregiver will update the health centre about his/her activities performed and the patient’s health profile by scanning the tag. This NFC tag will store information such as id, service’s description, date, check-in and checkout times which will provide assistance in acquiring information for the health centre about the caregiver and the patient. After acquiring information about the patient’s health, the health centre can set the recommendations based on the updates from the caregiver and the profile from the sensors. Furthermore, the patient can also scan these tags to check the report submitted by the caregiver and is able to obtain information about the medicine dosage.
4.1.2 Applying Outdoor Context Scenario

For outdoor environments, the user generally use a smartphone for context-aware services, therefore the outdoor scenario will use these mobile phones in order to connect with the user’s Bluetooth-enabled biosensors. The proposed model for the outdoor context scenario is divided into several layers and is visualized in figure 4.3.

![Diagram of Outdoor Context Scenario](image)

Figure 4.3: Outdoor context scenario

A user jogging on the track with either a wearable sensor or smart clothing will be updating the cloud infrastructure with health contextual values which may include blood pressure, temperature and glucose level etc. These values will be monitored by the health center and updates will be provided against these values. Furthermore, a user in a shopping mall or in car will be provided with the nearby location by using the Google location API. A layer based approach has been used in this case because the cloud infrastructure will be receiving the data from multiple streams. From these layers decisions can be made as to what information is required to be stored and what kind of information should be passed directly to the user.

**Sensors and Actuators Adapter Layer:** This layer deals with commanding the actuators and sensors for context aggregation and its representation in the physical environment. Context manager is set in order to
filter the contextual knowledge (local and distributed) from the different sensor’s adapters. The other task set for the context manager involves triggering up the context proxy, set in the higher layer.

**Context Reasoning Layer:** Context proxy is responsible for marshalling and unmarshalling the data of the sensor adapters by using the local and distributed interfaces. Remote service objects are set in order to provide the abstraction to context proxy. These objects are then used for setting and updating the contextual repository at the server.

**Third Party Context Layer:** This is the point at which the entity connects with the distributed overlay and where the broker has to update the context model. It will acquire the profile from the third party databases and services from the service repository. After acquiring intelligence about the entity it fuses into a contextual model and makes it available for the overlay to use which is visualized in figure 4.4.

![Figure 4.4: Third party context layer](image)

**Application Layer:** This layer is responsible for setting the user interface. Users will be provided with the same information as discussed in the indoor scenarios such as health graphs, doctor’s recommendations, services and locations for nearby pharmacy, restaurants and hospitals on their smartphones.
4.2 **Sensor Integration using MediaSense Framework**

Sensor integration was performed by updating the MediaSense framework. The reason behind using distributed overlay was that it provides reliable communication among the nodes and is independent of the type of sensors attached to it. The revision of the MediaSense framework involved performing several steps which are detailed below.

4.2.1 **Design of Context Information Integration Ontology Model**

Having a suitable contextual model for context gathering is useful while investigating how to persist and expose contextual information to the user. Figure 4.5 shows the simplified domain model in which entities, devices and related factors will be contributing to the building of the context information source intelligence.

![Figure 4.5: CII ontology model](image-url)

The Unified Modeling Language (UML) class diagram represents the correlation associated with the classes which is known as *cardinalities* where * stands for *any many*, 0..* means that there is no limit on the number of instances and 1..* means that at least one instance exists. This model is a simplified version of the entire architecture and it is based on the entity or object to which context can be attached. Smartphones, tags and sensors are used for gathering the entity context. Data representation is defined for setting up the data templates together with the provision of the time interval.
4.2.2 Persistent Storage for MediaSense Framework

The cloud storage infrastructure has been introduced for the distributed overlay, as illustrated in figure 4.6, to fulfil the requirement mentioned in Section 3.1.1 (requirement no 2). It will provide persistent storage for the nodes attached to the overlay and will overcome scalability and policy management issues. Furthermore, exposing clouds infrastructure to overlay will assist in complex and fast queries on the contextual data which resides in Bigtable.

Figure 4.6: Persistent storage for the overlay

In the updated MediaSense framework, the sensors are managed through a connected sensor gateway, which is further connected to the overlay through a sensor gateway API. The Zigbee sensor module illustrated in figure 4.6(a) was used in the proof-of-concept applications. This Zigbee sensor module was built at Mid Sweden University for test purposes and is capable of measuring temperature and humidity by connecting the box as a power strip. It can also sense the power con-
sumption of the devices plugged into the socket. Furthermore the actuators in this module can control the power sockets individually.

Device shown in figure 4.6(b) is the Ethernet sensor module which was also used for gathering the results and satisfying the requirements. It can also sense temperature and humidity but it has the advantage of communicating over the Ethernet and IP based infrastructure.

The overlay shown in figure 4.6 is the backbone of the architecture which is responsible for the communication of the distributed devices. It has the ability to perform the indexing mechanism among the nodes that are attached to it. The participating devices can act as a gateway for the services and applications. Sensors nodes broadcast a DISCOVER message for joining the overlay which in return send a DISCOER_RESPONSE message which is shown in figure 4.7.

![Diagram of Sensor discovery](image)

**Figure 4.7: Sensor discovery**

Furthermore, in DCXP a new primitive has been introduced called the TRANSFER primitive. By introducing the TRANSFER primitive, this protocol now has the functionality to shift and allocate the resources to those nodes where the demands for services are higher, which ultimately results in reducing the overhead of the network messaging. After the sensor discovery process, the distributed overlay now has a list of all the sensors which exist in the network and can be ranked. The ranking mechanism has been described in the next section.
4.2.3 Estimating Sensor Ranking

In the above given indoor scenarios for those patients at home, their sensors profile which is connected to the health centre should provide it with precise knowledge. In order to facilitate this, they should connect to accurate and reliable sensors reporting the correct values. For example, a temperature sensor close to a hot radiator in a room might not be the best sensor because it will be reporting several degrees higher than the actual temperature.

The MediaSense framework provides the means to rank sensors based on closeness and relevance. There exists, different approaches through which the sensors can be ranked, for example availability and usefulness. The proposed algorithm used in the MediaSense framework is capable of ranking the sensors and determining the most relevant and interesting sensor, by applying it in a distributed environment.

\[
\text{loop}
\]

\{
\text{at the local presentity, } i \}
\text{determine size of the local world } W_i
\text{for all data points } s \text{ attached to presentity } P_i \text{ do}
\quad \text{determine the ranking value with respect to } W_i
\quad \text{assign this as the local ranking value } SR_s
\quad \text{forward } SR_s \text{ to the domain owner } O
\text{end for}

\{
\text{at each domain owner, } j \}
\text{for all data points } s \text{ residing at } O_j \text{ do}
\quad \text{receive and aggregate all } SR_s
\quad \text{calculate its global rank } GR_s \text{ from all } SR_s
\quad \text{calculate domain rank } DR_j \text{ from all } GR_s
\text{end for}
\text{end loop}

In this algorithm \textit{W} is the local world, \textit{s} is the sensor data points, \textit{P} is the presence entity, \textit{SR} is the local sensor rank, \textit{O} is the domain owner, \textit{GR} is the global sensor rank, and \textit{DR} is the sensors domain rank. There are two main steps in this ranking algorithm. In the first step, a value with respect to the local world is determined, in order to work out the localized ranking, while the second step aggregates values to determine the global ranking value.
4.2.4 **Associating User with Overlay**

User can be associated with the overlay by introducing the concept of a schema. This schema can contain all the possible available information about the context of an entity, for example, information about a user’s profile containing sensor values and status etc. Associating a user with the overlay is visualized in figure 4.8.

![Figure 4.8: Associating user with the overlay](image)

Entity interacting with the prototype in order to gather information will SUBSCRIBE to the schema. REGISTER message is used for registering the user with the overlay which has a list of all the sensors being identified in the network by the DISCOVER and DISCOVER_RESPONSE messages. The overlay will update the schema by means of an UPDATE_SCHEMA message and a point of information i.e. sensor node will be assigned to the entity.

Users who have associated their schema with the overlay can set options for location based services relating to health care. The database compares the current and previous health contextual knowledge and returns the closest location for the availability of services according to the options defined by the user. Furthermore, this context schema can be associated with other context objects which enables other entities to access a user’s current context.
4.2.5 Connecting Devices to Overlay by using Smartphone

Google Nexus S phone was used for both reading and writing on the NFC tags and connecting with the Bluetooth sensor as illustrated in figure 4.9. This smartphone further connects with the overlay which stores the data into Google Bigtable by using the proposed MediaSense API. In this way, instead of storing the tag’s and Bluetooth sensors data on the smartphone, Google Bigtable was used to satisfy the requirement presented in Section 3.1.1 (requirement no 3).

![Diagram](image)

Figure 4.9: Connecting devices to smartphone

Device shown in figure 4.9(a) is a Bluetooth sensor module for generating raw sensor values. It can connect via Bluetooth to nearby mobile phones and has a lithium battery as its power source. This was used for the proof with regards to the outdoor context scenario mentioned in Section 4.1.2, in which users have Bluetooth-based wearable biosensors to monitor their health. The NFC tag shown in figure 4.9(b) was used for the proof regarding the indoor context scenario described in Section 4.1.1, where a caregiver visits the patient’s home to provide health services and scans the tag.
Context Storage and retrieval is achieved by the activities and services in Android. Activities are responsible for both window creation and interaction with the user, where services are long running applications in which different operations can be defined according to the task accomplished. For example, a network service can be defined for sending or retrieving the data from the server. Network adapters are used to issue an HTTP request such as GET and POST. Bringing a mobile phone into proximity of the tags will act as an initiator and behind it there are activities defined to handle it. Android team has made this reading and writing very simple by providing an android.nfc package which has a set of classes for tag discovery.
5 Results

In Chapter 5, the results are presented for the different scenarios discussed in the design chapter. Measurements and comparisons are also given for overlay registration, message latency, UCI registration and parsing time.

5.1.1 Acreo Lab Results

Application was tested for the TV screen in order to satisfy the requirement stated in Section 1.4 (Goal A in implementation). Most of the features are the same as on the smartphone and will be discussed in detail in the next section. Snapshot shown below has been taken at the Acreo lab. The TV set used for testing purposes was a Samsung 32 inch HD (1080p). Result is visualized in figure 5.1 on the first TV from the right side.

Figure 5.1: Result of TV display at Acreo Lab
5.1.2 Smartphone Interface

The results of the goals which have been discussed in Section 1.4 (Goal B in implementation) and the different scenarios that are presented in Section 4.1.1 and 4.1.2 for the smartphone are illustrated in this section. Tests were performed by using a Google Nexus S because it has the ability to scan the tags. In the following headings for example Home (Goal B.1 Results) the first text “Home” provides guidance concerning the activity name and the text within the brackets (Goal B.1 Results) describes the results achieved.

Home (Goal B.1 Result): This was discussed in the goals regarding menu screen and thus, after the application installation on the smartphone, figure 5.2 will be the first activity for serving the users. This activity is responsible for controlling all the other activities. At this point, users can navigate through the recommended services and can set their context in the context repository. There are different types of submenus which include Settings, Health, Reservation, Graphs, Food, Services, Scan Tag, Tracking, Direction, Emergency, Guide and Feedback.

![Figure 5.2: Main menu](image_url)
**Settings (Goal B.2 Result):** Login information using an email address and password is a difficult task for elderly people. Users registered with the healthcare centre can be identified by their personal number. The setting screen shown in figure 5.3 can allow the user to enter their personal number for the system login. Personal numbers will be stored in *shared preferences* for a one-time setup only. In the case of a loss of phone services, availability can be stopped from the control panel.

![Figure 5.3: Login Settings](image)

**Health Profile (Goal B.3 Result):** As discussed in the indoor and outdoor context scenarios, there will be different sensors attached to the user, so the activity shown in figure 5.4 displays the sensors values. On the basis of these values other activities such as *Services, Diet Plans, Reservation* and *Suggestions* are given. Suggestions for the user include a doctor’s visit and pills to be taken for an initial treatment.

![Figure 5.4: Sensors values and suggestions](image)
Reservation (Goal B.4 Result): In the previous activity the prototype has recommended the user to see the doctor, so the user has the ability to look for an available time slot with the doctor and reserve that time as shown in figure 5.5. If the user requires to change this visit then it is possible to cancel this time from the same activity as shown in figure 5.6, and this time slot will then become available for other users.

![Figure 5.5: Reserve time](image1)

![Figure 5.6: Cancel time](image2)

Graphs (Goal B.5 Result): When the patient reserves the time, the doctor will receive the notification about the reservation together with the patient's profile. This profile includes previous treatments, services and health graphs as shown in figure 5.7. Both the doctor and patient can view the medical history on the web, TV as well as on the smartphone. Based on reasons relating to battery consumption, different modes have been set for setting the time interval for updating the graphs.

![Figure 5.7: User health graphs](image3)
Diet Plans (Goal B.6 Result): Diet plans are set automatically according to the sensors values attached to the user schema, as shown in figure 5.8. For example patient A, who has high blood pressure, has different diet plans than those for patient B who has low blood pressure. The criteria defined for the diet plan suggestion are based on all possible sensor values. Furthermore, users have the option to order food by using the same activity.

![Diet Plans](image1)

Figure 5.8: Diet plans

Services (Goal B.7 Result): The activity visualized in figure 5.9 is also based on the user’s sensor profile and is similar to the diet plan available to the user. In this case, the user is informed about the daily available services by the caregiver instead of diet plans. There are different types of services which can be given by the caregiver including checkup, medicine administration and training etc.

![Services](image2)

Figure 5.9: Services plans
Scan Tags (Goal B.8 Result): Activity shown in figure 5.10 is for monitoring both the caregiver and the patient with the assistance of the NFC tags. Caregiver will scan the tag while entering the patient’s home and set the CHECK-IN button. After performing the services, he/she will write the medicine dose, services performed by him/her, report about the patient’s health and will press the CHECKOUT button. The UPDATE button will inform the health center about all the activities performed by the caregiver and it is also possible for the patients to scan the same tag in order to view these activities.

![Scan Tag](image)

Figure 5.10: Scanning NFC tag

Privacy Settings (Goal B.9 Result): User can accept and reject the GPS tracking for privacy concerns as shown in figure 5.11. As was discussed, in Section 3.1.1 (requirement no 4), if the user does not want to expose their context profiles to restaurants and advertising companies for offers, they should have the option to stop the tracking.

![GPS Tracking Options](image)

Figure 5.11: Privacy settings
Directions (Goal B.10 Result): The activity displayed in figure 5.12 is designed for outdoor context scenario by using Google map API. This API provides both XML and JSON response type. By using this activity user can find health center, nearby pharmacy, restaurants and shopping malls etc.

Figure 5.12: Location awareness

Emergency, Guide, Feedback (Goal B.11 Result): An emergency calling system has been set for the user which is shown in figure 5.13. Figure 5.14 indicates the guide book that has been designed for the user. The guide book contains guidance for every activity. Figure 5.15 illustrates the feedback system which has been developed for the user. The user has the ability to send the feedback for feature requests, praise and bugs etc.

Figure 5.13: Emergency Figure 5.14: User guide Figure 5.15: Feedback
5.1.3 **Web Interface**

Web interface was developed for the implementation of the goal discussed in Section 1.4 (Goal C in implementation). This interface provides the same services as for the smartphone which has been previously discussed in Section 5.1.2.

![OpenCare Web Interface](image)

Welcome Hans Hentzell  
Personal No 3142

<table>
<thead>
<tr>
<th>Sensors Profile</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation</td>
<td>Visit Doctor</td>
</tr>
<tr>
<td>Home Services</td>
<td>Enabled</td>
</tr>
<tr>
<td>Food Services</td>
<td>Enabled</td>
</tr>
<tr>
<td>Mobile Services</td>
<td>Enabled</td>
</tr>
<tr>
<td>Medical History</td>
<td></td>
</tr>
</tbody>
</table>

**Monitoring System**

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Temperature</th>
<th>Glucose</th>
<th>Blood Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-03-28</td>
<td>5:00 PM</td>
<td>102.0</td>
<td>80.0</td>
<td>90.0-130.0</td>
</tr>
<tr>
<td>2011-03-28</td>
<td>05:15 PM</td>
<td>101.0</td>
<td>85.0</td>
<td>95.0-130.0</td>
</tr>
<tr>
<td>2011-03-28</td>
<td>08:30 PM</td>
<td>101.0</td>
<td>85.0</td>
<td>95.0-140.0</td>
</tr>
</tbody>
</table>

[PREVIOUS] [1] [2] [3] [NEXT]

Showing 1 to 3 of 7

Figure 5.16: Web interface
Figure 5.16 illustrates the type of services which the user has enabled and the health monitoring system. The monitoring system shows the sensor values attached to the user's profile. Admin panel has been set for the health centre where administrative tasks can be performed on different activities. Doctors can view the reservation from the patient in addition to their health profile. These health profiles include health graphs based on sensor values similar to those for the smartphone. User can also view the available services, medications, recommendation, diet plans and locations etc.

5.1.4 Internet Tablet Interface

An application for the internet tablet has also been designed for the user. It provides the same services as for the smartphone and web as previously discussed. The view of the application illustrated in figure 5.17 is similar to the smartphone with certain changes. The application was tested by using an Archos and Samsung internet tablet.

Figure 5.17: Internet tablet interface
5.2 Measurements and Comparisons

Measurements were performed for the implementation of the goal that was discussed in Section 1.4 *(Goal D in implementation)*.

**Overlay Registration**: The process of connecting the sensor to the overlay consists of two steps. In the first step a DISCOVER message is sent and then in the second step a REGISTER message is used for joining the overlay. Latency depends upon the number of times the sensor node is redirected due to the received REGISTER_RESPONSE message. A test case was also conducted for direct registration. The results shown in table 5.1 are based on 10 samples.

<table>
<thead>
<tr>
<th></th>
<th>Shortest</th>
<th>Longest</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 step Registration</td>
<td>190 ms</td>
<td>231 ms</td>
<td>219 ms</td>
</tr>
<tr>
<td>2 step Registration</td>
<td>278 ms</td>
<td>371 ms</td>
<td>320 ms</td>
</tr>
<tr>
<td>Direct Registration</td>
<td>167 ms</td>
<td>213 ms</td>
<td>183 ms</td>
</tr>
</tbody>
</table>

Table 5.1: Overlay registration

**Message Latency**: After joining the overlay the message latency is the time to send and receive a message from one sensor node to another. Estimation is performed by using the formula in which $\text{Latency} = \frac{(t_1 - t_2)}{2}$. In this equation $t_1$ denotes the time for sending the request and receiving the response at the first sensor node, whereas, $t_2$ represents the time for receiving the request and sending the response from the second sensor node. The results illustrated in table 5.2 are based on 5 samples.

<table>
<thead>
<tr>
<th></th>
<th>Shortest</th>
<th>Longest</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Latency</td>
<td>122 ms</td>
<td>387 ms</td>
<td>243 ms</td>
</tr>
</tbody>
</table>

Table 5.2: Message latency
Universal Context Identifier Registration (UCI): UCI scheme is used to identify the context object at the time of registration as well as resolving the client. The process of registering a new UCI is similar to the process for connecting to the overlay, but, in the UCI registration there is no DISCOVER message used. It only requires REGISTER_UCI messages for discovering the correct sensor node. The results presented in table 5.3 are based on 15 samples.

<table>
<thead>
<tr>
<th></th>
<th>Shortest</th>
<th>Longest</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 step UCI Registration</td>
<td>271 ms</td>
<td>431 ms</td>
<td>377 ms</td>
</tr>
<tr>
<td>2 step UCI Registration</td>
<td>203 ms</td>
<td>391 ms</td>
<td>282 ms</td>
</tr>
</tbody>
</table>

Table 5.3: Universal context identifier registration

Context Requests: Context request is based on two steps. In the first step, UCI is resolved by RESOLVE_UCI and RSOLVE_UCI_RESPONSE messages, while in the second step, REMOTE_GET or REMOTE_SUBSCRIBE is used for the resolved UCI. In return, the proxy sends REMOTE_NOTIFY messages back to the node. The results given in table 5.4 are based on 10 samples.

<table>
<thead>
<tr>
<th></th>
<th>Shortest</th>
<th>Longest</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolving UCI</td>
<td>201 ms</td>
<td>397 ms</td>
<td>289 ms</td>
</tr>
<tr>
<td>Get Procedure</td>
<td>171 ms</td>
<td>231 ms</td>
<td>196 ms</td>
</tr>
<tr>
<td>Total</td>
<td>372 ms</td>
<td>628 ms</td>
<td>485 ms</td>
</tr>
</tbody>
</table>

Table 5.4: Context requests

Parsing Time: For the third party, an API has been provided to select the output format with the response timings given in table 5.5.

<table>
<thead>
<tr>
<th></th>
<th>Xml</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parsing Time</td>
<td>1021 ms</td>
<td>897 ms</td>
</tr>
</tbody>
</table>

Table 5.5: Parsing time
The results were compared with the existing MediaSense project in order to check the performance. These tests were conducted by using a Google NEXUS S with 1GHz processing power. The comparison of the results is shown in Table 5.6.

<table>
<thead>
<tr>
<th>Averages</th>
<th>Existing</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 step Registration</td>
<td>597 ms</td>
<td>219 ms</td>
</tr>
<tr>
<td>2 step Registration</td>
<td>734 ms</td>
<td>320 ms</td>
</tr>
<tr>
<td>Direct Registration</td>
<td>413 ms</td>
<td>183 ms</td>
</tr>
<tr>
<td>Message Latency</td>
<td>621 ms</td>
<td>243 ms</td>
</tr>
<tr>
<td>1 step UCI Registration</td>
<td>290 ms</td>
<td>377 ms</td>
</tr>
<tr>
<td>2 step UCI Registration</td>
<td>513 ms</td>
<td>282 ms</td>
</tr>
<tr>
<td>Resolving UCI</td>
<td>536 ms</td>
<td>289 ms</td>
</tr>
<tr>
<td>GET Procedure</td>
<td>567 ms</td>
<td>196 ms</td>
</tr>
<tr>
<td>Resolving UCI + GET</td>
<td>1103 ms</td>
<td>485 ms</td>
</tr>
</tbody>
</table>

Table 5.6: Analysis and comparison

The comparison table shows that the new approach has improved the performance as compared to the earlier implementation in some tested categories, although there were some limitations encountered during the node registration. One of the main limitations, which still exist, is that the registration approach is the same as the previous approach and only involves connection to the proxy. Other than this, the main reason for achieving the high performance during the node registration in the modified and existing framework was the use of the smartphone which had a high processing power. Using a smartphone with low processing power can affect the efficiency of the node registration.

Overall, other than node registration the performance of system has been improved. The measured response times of the overlay registration are 219 ms for 1 step registration, 320 ms for 2 step registration and 183 ms.
ms when direct registration was performed, which are approximately half of the previous times. Similarly, for the Message latency, devices that are in active network take 243 ms in total whereas it was previously 621 ms.

It is important to note that the result of the UCI registration in 1 step for the existing framework shows a better performance than for the modified framework, but it is slightly misleading. This performance is only because of registering the UCI locally in 1 step. A lower performance is observed when the UCI is not registered locally, which can be seen in 2 step UCI registration, which takes 231 ms longer as compared to that for the modified framework.

In the case of a context request, a significant reduction in time was observed as compared to that for the existing framework. The time differences in resolving the UCI for a context request were 247 ms and 371 ms for the GET Procedure. Overall the modified framework takes 485 ms for retrieving the context, which was 1103 ms in the existing framework.
6 Conclusions

The goal of thesis was to present a context-aware system for ubiquitous healthcare by updating the MediaSense framework and this has been successfully achieved. The problems faced in this thesis involved negotiation with different types of devices and the storing of large scale dynamic context information. In addition to the negotiating and storing, the framework was to be updated for the performance reasons. This chapter further describes issues faced while working towards the accomplishment of the research and is followed by a list of suggested future work with some concluding remarks.

This testbed has provided TV, smartphone, internet tablets and web interfaces. After registration with the prototype and when equipped with different types of smart devices as discussed in the indoor and outdoor scenarios, it is possible for users to frequently update the health center about their health status. Based on the provided health context both the user and the health center have the ability to view the health graphs, reservation from the doctors, diet plans and services automatically. Furthermore, the application has the ability to inform the user about the location of the health center, nearby pharmacy and hotels etc. Control panel further ensures the administrative task for every module.

Earlier research on the MediaSense framework has addressed context awareness by using mobile devices and MySQL database as a context information base which was the main factor for reducing the performance of the overlay and devices. Therefore the updated framework has showed the utilization of a cloud infrastructure as a context information base. Using this infrastructure ultimately results in the provision of massive storage with rapid responsiveness and makes the context information ubiquitously available. In addition to providing a cloud infrastructure as a backbone, this research has identified how other technologies such as RFID and NFC tags can also be used with a MediaSense framework.

The results suggest that the modified solution has shown a considerable improvement as compared to the previous approach. Overlay registration can now be achieved in a few ms in the best case scenario. Same pattern is visible for message latency, UCI registration and context
request. As a result, context data can now be handled in about half a second as compared to an existing framework but it is still not the ideal and requires further improvements. There are certain issues which still exits, including the case when a node leaves the overlay without sending an UNREGISTER message, the time it takes to notify the surrounding nodes ranges from 0 up to 20 seconds, which is a considerably long delay in context-sensitive environments. Furthermore, from the analysis on the testbed it can be concluded that there are possibilities to make the testbed more efficient by eliminating certain limitations, which are discussed in a further heading.

6.1 Future Work

Some avenues, which arose when conducting this thesis, require further attention. One aspect is towards security implementations for the testbed as there are possibilities to eavesdrop, thus this could involve possible future work. The functionality of the distributed overlay for the new smartphone with high processing power is very good, but it slows down for the smartphone with low processing power, as it requires too many processing cycles. The possible solution to this problem can be achieved by changing the message format. Other approaches can also be explored, such as discovering additional programming patterns, which are more suitable for the testbed. Further studies and development for the testbed could be performed by simulations for the billions of user in order to determine the stability and performance. For maintaining security issues, a project named InSIKT has been initiated by Mid Sweden University together with Acreo in order to develop a secure and robust network.

6.2 Concluding Remarks

This thesis has designed and developed a research testbed for supporting context awareness in healthcare. The testbed is easily scalable because the cloud infrastructure acts as a backbone and has the ability to negotiate with different devices for gathering a massive amount of context information. Furthermore, the modified MediaSense framework can manage different interfaces including TV, web, internet tablet and smartphone in different scenarios. This prototype can be considered as a major step towards truly pervasive applications. It is to be hoped that this work will open the possibilities for further studies in the area of context-aware computing.
References


Beth Bacheld, “Georgia Tech Researchers Create an RFID-Sensor Medical Patch” RFID Journal.

Beth Bachedor, “Hybrid Tag Includes Active RFID, GPS, Satellite and Sensors”, RFID Journal.


References


Appendix A: Contents of CD

Publication of code is not allowed. All right are reserved to Mid Sweden University, Acreo and OpenCare.

The CD contains the following folders:

- **Thesis Folder**: contain thesis.
- **Presentation Folder**: contain presentation.
- **Guide Folder**: contain installation guide.
- **Mobile Folder**: contain Android project.
- **Server Folder**: contain server side code.
- **MediaSense Folder**: contain updated framework.