UNIVERSITÀ DEGLI STUDI DI CAMERINO

SCHOOL OF ADVANCED STUDIES

DOTTORATO DI RICERCA IN SCIENZE DELL'INFORMAZIONE E SISTEMI COMPLESSI - XXVIII CICLO

SETTORE SCIENTIFICO DISCIPLINARE INF/01 INFORMATICA

SCUOLA DI SCIENZE E TECNOLOGIE

Real-Time Sensor Data in Smart Environments

Relatore

Rosario Culmone

Leonardo Mostarda

Dottoranda

Nadeem Qaisar Mehmood

Commissione

ANNO ACCADEMICO 2013-2016

UNIVERSITY OF CAMERINO

SCHOOL OF ADVANCED STUDIES

DOCTOR OF PHILOSOPHY IN INFORMATION SCIENCE AND COMPLEX SYSTEMS - XXVIII CYCLE

SCHOOL OF SCIENCE AND TECHNOLOGY

Real-Time Sensor Data in Smart Environments

Advisor

Rosario Culmone

Leonardo Mostarda

PhD Candidate

Nadeem Qaisar Mehmood

Exam Commission

ACADEMIC YEAR 2013-2016

Mehmood, Nadeem Qaisar
Real-Time Sensor Data in Smart Environments
Ph.D Thesis, University of Camerino,
Camerino, Italy. 2013-2016.

Committee

Coordinator Prof. Emanuella Merelli
University of Camerino, Italy

Supervisor Prof. Rosario Culmone
University of Camerino, Italy

Co-Supervisor Prof. Leonardo Mostarda
University of Camerino, Italy
Middlesex University, UK

External Reviewers
- Prof. Enver Ever
  Middle East Technical University, Turkey
- Prof Orhan Gemikonakli
  Middlesex University London, UK

Exam Committee
- Prof. Luca Tesi
  University of Camerino, Italy
- Prof. Francesco Tiezzi
  University of Camerino, Italy
- Prof. Orhan Gemikonakli
  Middlesex University London, UK
- Prof. Enver Ever
  Middle East Technical University, Turkey

In the name of Allah, the Most Gracious, the Most Merciful.

To my Parents,
To my Wife and Daughters,
To my Brothers and Sisters
...

Abstract

Rapidly increasing ageing society lacks fast healthcare services both at the homes as well as in the hospitals. This is more than a medical problem, which needs the role of Information and Communication Technology (ICT) to provide cost-effective and high quality remote healthcare facilities. We should find better healthcare systems ranging from an individual's body care to home care or hospital care. In Ambient Assisted Living (AAL) the Smart Environments are used for comfortable secure living of the elderly. They use the sensor technology for the monitoring purposes, hence produce bulky data. Such data-driven environments, where the data defines the control flow of the system, require flexible and scalable software architectures. This dissertation is aimed at sketching a software architecture, a methodology guide and an architectural framework skeletal, which reflect the properties of a healthcare system within a distributed wireless sensor network environment having pervasive characteristics. The further aim is to have temporal data modelling and storage of semi and unstructured data.

A prototyping approach is used to meet the objectives related to the foundation of system architecture, a methodology guide and architectural framework for the context data acquisition and temporal schema-flexible storage. The architecture of the final prototype have three components, i.e. (i) *Windows Daemon Service*, to acquires sensor data from machine to machine devices, to transform the data into light weight Java Script Object Notations objects, and to transmit objects to the remote server in real-time for schema flexible storage; (ii) *Communication Middleware*, to bind the two heterogeneous distributed remote components using their universal interfaces for a real-time event-driven full-duplex exchange of different formats of data over the Internet; and (iii) *Context Data Server*, to accept volumes of periodically arriving data messages, to store them according to their temporal characteristics taking the advantage of NoSQL storage mechanisms. Temporal schema models are defined out of the database using object mapping middlewares. The data integration algorithms are used to store sensor data in a hierarchical man-

ner while abiding by the schema model rules.

A layered architectural framework having a skeletal structure is designed, which define the suggested architectural artefacts, and describe how those artefacts are related to each other. The conceptual framework intended to serve as a support or guide in building soft real-time machine to machine (M2M) based sensor monitoring applications. The framework consists upon four layers, i.e. (i) Things realization; (ii) Interpretation and information processing; (iii) Real-time communication; and (iv) Data integration and application layers. Each of these layer holds key artefacts to play the role of building blocks in architectural development for scalable real-time SHE applications.

In the research work the presented methodology acquires physiological life vital parameters of a patient using the ANT+ wireless sensor technology, which contains standard device profiles to support interoperability and vendor in-dependency. There are many technical obstacles during the data acquisition and storage phases, such as devices' heterogeneity; distributed heterogeneous components, efficient transmission techniques; sensor data interpretation, processing and storage of multiple data formats; and volumes of sequential data. To solve these issues the constructed architecture utilizes the new generation technologies, such as Internet of Things (IoT), Big Data and real-time Web. The system is flexible to add hardware and develop further software components; and it is applicable at a global level since it is scalable both horizontally as well as vertically.

Keywords: Ambient Assisted Living (AAL), healthcare, Windows services, ubiquitous computing, wireless sensor networks (WSNs), distributed systems, IoT, big data, NoSQL databases, interoperability, event-driven architecture, real-time Web, architecture, framework

Acknowledgements

I am most grateful to my co-supervisors, **Professor Rosario Culmone** and **Professor Leonardo Mostarda**, from the Division of Computer Science, University of Camerino. Much of their devoted time and help have enabled me to complete this work. Whenever I had a problem during the studies they guided me towards the right direction. Since from the beginning of my research studies, Professor Culmone's efforts are very much admirable and I am especially thankful to him. He is not only a good professor; rather have also been for me a friend, mentor, and teacher. He has a happy polite attitude and time for his students and colleagues. He encourages his students for novel ideas and let their qualities flourish.

Secondly, I would like to acknowledge all those who helped me during this research work from the Division of Computer Science, University of Camerino. Especially to **Prof. Andrea Polini** for allocating time to discuss many technical things during the research. He is kind and encourage such discussions. Special thanks to the Unicam's librarian, who downloaded many papers for me for this research. I am thankful to Adane Letta's grammatical reviews to some of the publications. In the same manner, there are many people in this university whom support and kind wishes provided me motivation to take small steps for the long distance.

This work is part of the EUREKA (XXVIII Cycle) project which results from the cooperation between Servili Computers s.r.l., Italy; the University of Camerino (Department of Computer Science), Italy; and the state department Regione Marche, Italy. I am thankful to the Software Industry; Ministry of Education and Research; and the University, Italy, for the financial support of this project.

Last but not least, I would like to thank my parents, my wife who gave time to complete my Ph.D. studies, my daughters, brothers, and sisters, friends, and all the people who have supported me along the way.

List of Publications

Journal Papers

J2 Nadeem Q. Mehmood, Rosario Culmone, Leonardo Mostarda: Modeling Temporal Aspects of ANT+ Data for MongoDB. International Journal of Big Data. Submitted on July 19, 2016.

J1 Nadeem Q. Mehmood, Rosario Culmone, Leonardo Mostarda: A Flexible and Scalable Architecture for Real-Time ANT+ Sensor Data Acquisition and NoSQL Storage. International Journal of Distributed Sensor Networks. Published on May 23, 2016.

Conference Papers

C3 Nadeem Qaisar Mehmood, Rosario Culmone: A data acquisition and document oriented storage methodology for ANT+ protocol sensors in Real-Time web. In: The 30-th IEEE International Conference on Advanced Information Networking and Applications (AINA-2016),Crans-Montana, Switzerland (2016).

C2 Nadeem Qaisar Mehmood, Rosario Culmone: An ANT+ Protocol Based Health Care System. In: Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference On, pp. 193-198 (2015). IEEE

C1 Nadeem Qaisar Mehmood, Rosario Culmone, Leonardo Mostarda: An Ontology Driven Software Framework for the Healthcare Applications Based on ANT+ Protocol. In: Advanced Information Networking and Applications Workshops (WAINA), 2014 IEEE 28th International Conference On, pp. 245-250 (2014). IEEE

Contents

List of Figures

List of Tables

List of Abbreviations

AAL	Ambient Assisted Living
ACID	Atomicity Consistency Isolation Durability
ADT	Abstract Data Types
ADTec	Assistive Devices and Technologies
AI	Artificial Intelligence
AIM	Application Integration Middleware
AmI	Ambient Intelligence
ANT	Advanced and adaptive Network Technology
ANT-FS	ANT-File Share
AJAX	Asynchronous Java Script And XML
API	Application Programming Interface
ASP	Active Server Pages
ATA	American Telemedicine Association
AVRCP	Audio/Video Remote Control Profile
AWLP	ANT Windows Library Package
BCS	Body Composition Service
BDMF	Big Data Management Frameworks
BLE	Ultra Low Power Bluetooth
BPP	Blood Pressure Profile
BSI	Bluetooth Special Interest Group
CASIS	Context-Aware Service Integration System
CCOW	Clinical Context Object Workgroup
CDs	Chronic Diseases
CDA	Clinical Document Architecture
CERP	Cluster of European Research Projects
CGMP	Continuous Glucose Monitor Profile
COM	Component Object Model
COTS	Commercial Off-The-Shelf
CPS	Cyber Physical Systems
CSCP	Cycling Speed and Cadence Profile
DBMw	Database Middleware
DM	Data Mining
DNS	Domain Name System
DLL	Dynamic-Link Library
DSMS	Data Stream Management System

DOS Distributed Object Systems
DWS Distributed Web Systems
EBM Event-Based Middlewares
EDA Event-Driven Architecture
EDC Electronic Data Capture systems
EES Electronic Email System
EHR Electronic Health Record
EPC Electronic Product Code
FHIR Fast Healthcare Interoperability Resources
FIPA Foundation for Intelligent Physical Agents
FIT Flexible and Interoperable Data Transfer Protocol
FoLDoC Free-online Dictionary of Computing
FTP File Transfer Protocol
GATE Global Cooperation on Assistive Technology
GCL Group Collaboration layer
GFS Google File System
GLP Glucose Profile
GPS Global Positioning system
GUI Graphical User Interface
HAN Home Area Network
HAS Home Automation System
HCI Human Computer Interaction
HCIS Healthcare Information Systems
HCM Health Care Monitoring
HDP Health Device Profile
HIS2 Health Integrated Smart Home Information System
HRP Heart Rate Profile
HTP Health Thermometer profile
HTTP Hypertext Transfer Protocol
IA Information Architecture
ICT Information and Communication Technology
IDL Interface Definition Languages
IHCO ICT in Health Care Observatory
IHE¿Integrating the Healthcare Enterprise **IPC**¿Inter Process Communication
IPSO IP for Smart Objects
ISTAG Information Society Technologies Advisory Group
ISO International Standards Organization
IoT Internet of Things
IP Internet Protocol
IPC Inter-Process Communication
JADE Java Agent DEvelopment Framework
JRE Java Runtime Environment
JS Java Script
JSON Java Script Object Notation
JSON-LD JavaScript Object Notation for Linked Data
JSP Java Server Pages
JVM Java Virtual Machine
KET Key Enabling Technologies
KIT Keep In Touch

LAN Local Area Network
M2M Machine to Machine
MAC Medium Access Control
MADM Multi Attribute Decision Making
MAL Monitoring and Assistance layer
MCU Micro Controller Unit
MDC Monitoring Datacenter
MDS Model Disability Survey
MEMS Micro Electro-mechanical Systems
MOM Message Oriented Middleware
MPI Message Passing Interface
MQS Message Queuing Service
MVC Model View Controller
NFC Near Field Communication
NRT Near Real-time
NoAM NoSQL Abstract Model
OODBMS Object Oriented Database Management Systems
OOM Object-Oriented Middleware
OOP Object-Oriented Programming
ORM Object Relational Mapping
ORB Object Request Broker
OSGi Open Services Gateway initiative
OSI Open Systems Interconnection
PAC Presentation Abstraction Controller
PANs Personal Area Networks
PDA Personal Digital Assistant
POJO Plain Old Java Object
QoL Quality of Life
RDBMS Relational Database Management Systems
RDF Resource Description Framework
RHCM Remote Health Care Monitoring
REST Representational State Transfer
RF Radio Frequencies
RMI Remote Method Invocation
RML Response Management layer
RPC Remote Procedure Call
RSDL REST Service Description Language
RSSC Regional Social Security Card
RFID Radio Frequency Identification
SAGE Study on global AGEing and adult health
SDLC System Development Life Cycle
SO Smart Objects
SoC System-on-Chip
SC Smart City
SE Smart Environment
SE Smart Environments
SG Smart Grids
SH Smart Home
SHE Smart Home Environment(s)

SO	Smart Objects
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SLDs	Second Level Domains
SPE	Stream Processing Engine
TCP	Transmission Control Protocol
TPM	Transaction-Process Monitor Middleware
ubicomp	Ubiquitous Computing
UDDI	Universal Description Discovery and Integration
UIF	USB Interface Board
uID	Unique/Universal/Ubiquitous IDentifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAN	Wide Area Network
WBAN	Wireless Body Area Network
WPAN	Wireless Personal Area Network
WS	Web Service
WSAN	Wireless Sensor and Actuator Network
WSDL	Web Service Description Language
WSN	Wireless Sensor Network
WSP	Weight Scale Profile
WoT	Web of Things
XDS	Cross Enterprise Document Sharing
XHR	XML Http Request Object or XMLHttpRequest
XML	Extensible Markup Language
XSL	XML Extensible Stylesheet
XMLS	Extensible Markup Language Streams or Schema

Part I

Introduction

Chapter 1

Introduction

But words are things, and a small drop of ink,
Falling, like dew, upon a thought, produces
That which makes thousands, perhaps millions, think.
-Lord Byron

1.1 Ageing and Health

Ageing population of a country reflects the standard of healthcare within the society and the country's healthcare policies. Quality of Life (QoL) is always the main goal of an individual at any age but this becomes more important with the passage of time; especially during the old age, when the mobility and being productive become a challenge; due to weakness, depression, diseases or physical or mental disabilities. WHO facts about ageing are as: "Between 2000 and 2050, the proportion of the world's population over 60 years will double from about 11% to 22%. The number of people aged 60 years and over is expected to increase from 605 million to 2 billion over the same period" [288]. The proportion of individuals aged 80 or over is projected to rise from 1 percent to 4 percent of the global population between today and 2050 [70]. *WHO Study on global AGEing and adult health (SAGE)* conducts census based study by collecting data on adults aged 50 years and above, plus a smaller comparison sample of adults aged 18-49 years, from nationally representative samples in China, Ghana, India, Mexico, Russian Federation and South Africa. This organization is supported by the US National Institute on Ageing, Division of Behavioural and Social Research and cooperating national governments [290]. This is the result of simultaneous drop in fertility rates and longer life expectancies. Such demographic issues are equally applicable to both developed as well as developing countries and each country's ageing population, over 60 years of age, will increase every year, as depicted in

Figure 1.1 [70]. European Commission ageing report states that by 2060 one in three European will be over 65 [142]. The ratio of working people to inactive, is shifting from 4 to 1 today to 2 to 1 by 2060.

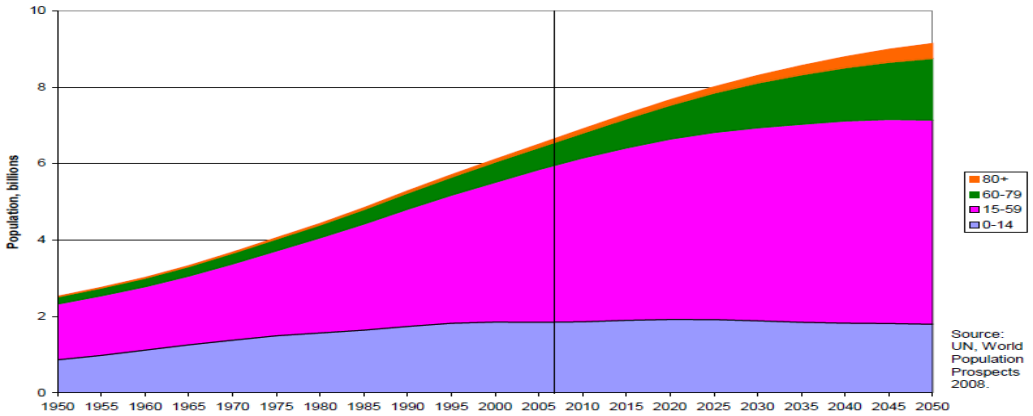


Figure 1.1: World, population by age group

Such changes can be normal but its magnitude and speed is surprising and has influenced the healthcare systems, labour market, affects the goods markets, government services, society's living standards and an individual's QoL. QoL has became a crucial component in context to a country's economic and social progress. The challenge is how to keep elderly people productive, active and energetic, because at an old age people lose self-control and independency due to physical or mental diseases. It affects their mobility and restricts them to their own houses, especially during the winter seasons in the cold regions, such as in European countries.

Beside this, more than billion people are suffering with some sort of disability through out the world, because of poor health conditions, lack of education, less economic growth and increase in poverty. Disability is more a human fundamental right to get support rather than just to be a social problem. WHO's *Model Disability Survey (MDS)* conducts general surveys to collect details about how people with or without disabilities conduct their lives and what difficulties they face during the entire life span [289]. To enhance the over all working conditions of a disable person different Assistive Devices and Technologies (ADTec) are used; whose primary motive is to improve an individual's functionality and independence to participate and contribute normally within the community. Examples of ADTs include hearings and visual aids; wheelchairs; walkers; prosthesis; Braille and special computer software and hardware that increase mobility; hearing and vision; or communication capacities. WHO is developing a flagship programme *Global Cooperation on Assistive Technology (GATE)*- for the people with disabil-

ities [**?**]. A large portion of disabilities are due to the barriers people face during access to health facilities. Disabilities at an older age are more difficult to get treatment due to weak body muscles and bad body resistance power. Resultantly quick measurement are required that could work in any weather condition and could be accessible to all kinds of people - including with disabilities or without disabilities - and the new information technology is the key to open the door.

Hence well-being at an older age is the ultimate goal for every one, ranging from an individual to a global society. This also provokes many research questions to identify the main factors influencing older people's quality of life, and to make measurements that would promote the healthy habits, better medical treatment, preventive measures and improvement in living conditions. Frailty, limited mobility or other physical or mental health problems affect an individual adversely to lose his/her ability for independent living. Although healthcare services are traditionally supplied within a hospital or at a medical centre. At an old age most of the old people need these services more proactively for a long time, such as home care, social care and assisted living, residential nursing or stays in hospitals with protection. The research shows that an individual's health management, exercise and physical training, activity monitoring, medication tracking and dietary habit monitoring on daily bases contribute greatly to prevent diseases. This keeps them active, energetic and productive. This helps them to reduce their physician consultation or hospital visits for treatment or diagnoses. Hence ranging from an individual to a whole community, all want health solutions and to have access to healthcare facilities at their door steps.

There is an ever-growing need to supply constant care and support to the patients having a disease or a disability. In European countries, the most common cause of death is due to a heart disease [151]. According to US National Centre for Health Statistics, major Chronic Diseases (CD), such as heart disease, cardiovascular disease, and diabetes account for 35.6% of death in US in the year 2005 [225]. In 2013, the 10 leading causes of death (i.e, heart disease, cancer, chronic lower respiratory diseases, unintentional injuries, stroke, Alzheimer's disease, diabetes, influenza and pneumonia, kidney disease, and suicide) accounted for 73.6% of all deaths in the United States [222]. According to the latest World Health Organization (WHO) data published in may 2014 Coronary Heart Disease Deaths in Sudan reached 9,491 or 3.64% of total deaths [154]. In Italy the main causes of death are, as: coronary heart disease, stroke, lung cancers, breast cancer, hypertension and alzheimers/dementia with 48.48, 35.35, 25.15, 19.94, 17.81 and 16.96 death rates per one lac population respectively [152]. In Pakistan the main causes of death are, as: coronary heart disease, stroke, influenza and pneumonia, lung disease, tuberculosis and diabetes mellitus with 110.65, 87.21, 62.83, 62.29, 52.80 and 39.65 death rates per one lac population respectively [153].

Good health facilities benefit the whole society in general and people of all

ages take advantages from them. Young people keep themselves fit and healthy to avoid early old age. They can calculate their intake diet or calculate their steps during an exercise to measure their fitness level to keep them active, healthy and energetic. During the training they use different healthcare equipments, such as a stopwatch to measure a lap's time; heart rate monitor to measure the heart beat; and weight machines to measure the body weight. Such group of people mostly spend an active life and visit different places very often. Hence they always seek facilities that would be available any time and everywhere, such as in a mobile; available on the internet every time, or available in each computer's operating system. They demand healthcare equipment which they can carry with them easily or can wear on their body such as a time watch or glasses.

European digital agenda states that the cost of health and social care will rise substantially to about 9% of EU GDP in 2050 [380]. We can see a complete list of income spendings upon health by each country here [317]. Although, spending upon healthcare contributes to achieve a smart social and economic growth; and this reflects a healthy society; but it is always a challenge to minimize the cost with the optimum results. Therefore we need new transformed healthcare systems that can ensure affordable access to evidence based medical treatment with care [160]. On the other side it has also opened up a new domain for the business stakeholders to invest in this sector. By 2020 Europe will face up to 2 million vacancies in health and social care [106]. Public spending accounts for between 45% and 56.1% of U.S. health care spending [330]. Private insurance accounts for 35% of total health spending in the United States [227]. In France and Canada private health insurance accounts for 13% or over all spending [227]. OECD Health Statistics, being the most authentic source for healthcare statistics, says that there is significant spending reductions across all parts of the health care sectors; alongside across the 34 OECD countries the spending for private health insurance accounts for only 7% of health spending [285]. There is need for scientific human resource to conduct research, that must be collaborative and cross disciplinary to address fully complex healthcare challenges [372]. New technologies are increasingly seen as sharing the part with the solutions to the challenges, but such advancements are not available without an expense. This encourages the business stakeholders to invest in the healthcare sector to provide active care facilities from a remote location to elderly at their own homes. There are enormous requirements to move the routine medical check-ups and healthcare services from hospital to a patient's home environment, thus releasing the hospital beds and other limited resources for the people who urgently need them [76].

With the advancement in the new technology and the availability of the scientific equipment to conduct biological, physiological and chemical experiments; healthcare is not just consulting with the general physicians and taking herbal medicine; rather it is more than a medical problem and requires more novel re-

search in accordance to the ageing population requirements. After direct healthcare means, such as family care, doctor, physician or nurse care; (ICT) has become second most powerful tool in delivery of cost effective, high quality and better remote healthcare services to achieve the goal of QoL. Today, there are 7 billion mobile subscribers ((95% of the global population) worldwide. Globally, 3.2 billion are using the Internet, of which 2 billion reside in developing countries [206]. The Percentage of households with Internet access has reached 1 billion worldwide, of which 230 million are in china, 60 million are in India and 20 million are in the 48 Least Developed Countries (LDCs) [207]. 3G mobile broadband will reach 69 per cent of the global population until 2015 [206]. "ICTs will play an even more significant role in the post-2015 era and in achieving future Sustainable Development Goals as the world moves faster and faster towards a digital society," said Mr Brahima Sanou, the Director of the ITUs Telecommunication Development Bureau [206].

ICT is the powerful ally as it empowers the society individuals of any age to better manage their health at any place. This helps the elderly society to stay independent, healthy, productive and active at work or in the society. European agenda states that the introduction of ICT and telemedicine alone is estimated to improve efficiency of health care by 20% in Europe; therefore the Commission funds several research projects with the same goals [106]. European Seventh Research Framework Programme has just finished and its successor Horizon 2020 is still on the agenda; which have many sub healthcare projects. This is with the vast goals to bring together the healthcare professionals, government, industry and a common community individual across the borders to enlarge and bridge the gap between a consumer and a producer, seller and buyer [107]. Remote healthcare through ICT helps to achieve the three main goals, as: (i) a better quality of life for European citizens (ii) innovation and growth for a competitive EU industry to help the economic market, and 3) more robust healthcare systems for the community.

Such factors has motivated the technology researchers to invest their time and resources to solve the healthcare problems for any age group. They take advantage of the developments in ICT fields to find tools for identification, monitoring, security and care. They are with the goals to design and develop different intelligent information systems that could meet the requirements of different age groups within a society, but not limiting the solutions only with the elderly people. The researchers want to facilitate the elderly at their homes and to provide them with all commodities and necessities; to maintain or increase their autonomy, security and health. While using ICT mechanisms the guiding principles are as: to watch constantly and protect them remotely; to help them to perform daily tasks using some assistance in case of some disability; to let them interact and integrate with the other community members; to keep them active and to see them in case of need; hence to be Keep-In-Touch (KIT) with them.

1.2 Gerontechnology

Gerontechnology, a composition of *gerontology* and *technology*, is an interdisciplinary field that integrates the state of the art technologies to the desires and needs of ageing and aged adults [79]. This is the study of the cognitive, psychological, social, and biological aspects of ageing. This is a technology based study about ageing concerns with a type of research that have roots in ICT, mathematics, biological, psychological, social and medical domains. An ageing societys survival depends upon effectively creating technological environment. That includes assistive technology and comprehensive design for an independent living at home and the social participation of adults with the community with comfort, good health and safety. Therefore, gerontechnology matches the health supportive technological environments possessing mobility, communication, leisure and work for the elderly. It benefits the senior citizens to enhance their performance and productivity by providing opportunities to work independently at their homes.

Improving the QoL of elderly, with the integration of ICT, healthcare and Assistive Technology (AT) in their daily life is a research topic. Healthcare services, such as Electronic Medicine, Electronic Health, Telemedicine, Mobile Health, and Tele-health, have taken care of the young and the elder society members [147]. According to American Telemedicine Association (ATA), telemedicine is technology-based services tailored to use for handling medical information exchanged from one location to another via electronic communication systems to improve a patient's clinical health status [54]. Telemedicine includes a growing variety of applications and services using two-way video, email, smart phones, wireless tools and other forms of telecommunications technology. Although telemedicine includes remote patient monitoring but AAL is more in context to the use of pervasive technologies to monitor the user's environment and response accordingly.

This requires computing and intellectual resources to be invested to build healthcare oriented Intelligent Environments (IE) for Ambient Assisted Living (AAL) and Ambient Intelligence (AI). Some of the aspects that need to be handled are: finding frameworks that can play a glueing role to integrate different stakeholders, services and activity, and life vital measuring equipments. Such frameworks should allow the ease of use and adaptability of these technologies for elderly usage and could expose ubiquitous environments with sense and react features for Remote Monitoring and Treatment (RMT) [249].

1.3 Smart Environments (SmE)

In healthcare this is not possible to nurse a society without measuring the body parameters, such as heartbeat, blood pressure or temperature. The patients suffer-

ing from chronic diseases, such as diabetes, asthma and heart attacks, can not visit their physicians or hospitals every day but they require continuous monitoring to log their life vital parameters. They need treatment at home especially during bad weather or during the old age and the situations become even worse if they have some disability. Hence a smart technology is required to help identify the location of an individual using some form of communication technology in order to monitor and transmit vital signs and readings to the medical practitioners or care providers. This demands a wearable technology or Smart Objects (SOs) [324] - this may be embedded inside the body, or placed directly on the body at a fixed position - which serves to monitor the body parameters and transmits them as wireless signals. Wearable and implantable bio-sensors are embedded computers to develop wireless networks around a person's body called Body Area Networks (BANs). They are to provide monitoring services, such as personal care, hospital care, clinical care, individual's security, and fitness applications [96, 40, 322].

An Intelligent environment can respond to the needs of older people to help prevent life risk issues, in order to reduce their dependency upon others later in the life. To perform response actions, an environment must identify its resources and should understand its users with in the surrounding context. For it, such smart environments build user profiles and save the context information about the overall environment factors, that include user location, building status (e.g. temperature or light or a communicating thing), vital signs (e.g. heart beat or blood pressure), environment maps and means to sense-and-react; hence this is to build Personalized Healthcare Systems (PHS) [36, 265].

The history of smart homes has its roots in the automation of homes and buildings. But building automation deals with remotely controlled tangible devices, such as lights, fans, TV, heaters or ventilations; to improve safety, security, comfort, energy optimization and luxury. But building automation does not consider the personalization of a user and his behaviour, nor the life vital parameters. So in smart homes just gathering the context information is not enough rather the system must process, analyse and do reasoning to make decisions against the reactions or actions of the patients. The decision making process highly depends upon the quality of information context as well as the dynamic means and methods of processing, analysing and reasoning.

1.3.1 Different Overlapping fields

Therefore a SmE is with the idea of integrating smart technologies and ICT into an environment with ubiquitous features. Such environments range from private to public and from fixed to mobile; such as patient's home, healthcare centres, hospitals, cars, Assistive Environments (AEs) and smart buildings. In order to explore and develop such technical environments different expertise are required

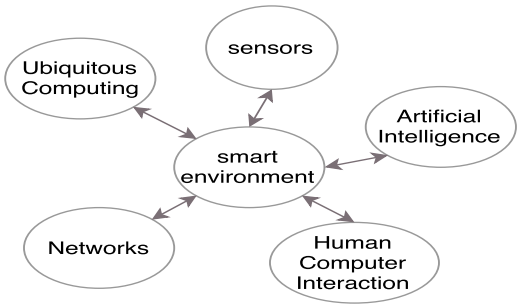


Figure 1.2: The different fields of ICT that relate with Smart Environment [56]

in many fields of ICT, such as **Communication and Networking Technologies (Wireless Sensor Networks (WSNs), Communication Protocols, Middlewares)** [119, 175]. In ICT the WSNs are used to explore applications such as environmental monitoring, situational awareness, and structural safety monitoring by building different networking topologies, such as star or mesh network topology. The networked devices use different wired or wireless communication protocols to talk to each other, like WiFi, RFID, ZigBee, ANT+ or Bluetooth [119, 175]. Whereas a middleware plays the role of glue to help bind the different distributed components of the system in a flexible and interoperable manner.

The other related fields are Ubiquitous Computing, **Artificial Intelligence** (AI) [320], Machine Learning or **Data Mining (DM)** techniques [353, 386], **Microelectronics** (*sensors, devices, mobiles and computers* [293]) and **Human Computer Interaction (HCI)** [216], as shown in Figure 1.2.

1.3.2 Ambient Intelligence (AmI)

AmI has ubiquitous environments, in which different appliances are, integrated into the surrounding to such an extent that they are used without even noticing with a conscious mind about their existence [34]. Such appliances intelligently sense and react to a user's actions. The European Commission's Information Society Technologies Advisory Group (ISTAG) [129, 130, 308] introduces the broad vision about a developing society , in which the emphasis of AmI is on greater user-friendliness, more efficient services support, user-empowerment, and support for human interactions. Under these four scenarios community members will able to work together using intelligent intuitive interfaces that are embedded in all kinds of objects. Such environments would able to notice the presence of an individual and will respond him/her proactively. Above all, the features of seamless, unobtrusive and invisible operations will dominate during all interactions. The intelligent interfaces will make enable the people and devices to interact with

each other and with the environment. Some other definitions about AmI are as:

- AmI is an emerging multidisciplinary area based on ubiquitous computing which influences the design of protocols, communications, systems and devices, etc [384].

- AmI tries to adapt the technology to the peoples needs by means of omnipresent computing elements, which communicate amongst them in an ubiquitous way [33].

It is with the vision to adapt the environment according to the users instead of the users to accept the environments. The five features which an AmI system provides [253], are: (i) Non-obtrusive: the user's life should get effect, (ii) Context-awareness: the system understand the context and react accordingly, (iii) Personalized: should understand its user and his needs by modelling his profile and behaviour, (iv) Adaptive: the system can respond in a dynamically adaptive manner based upon the context information and user's profiles and (v) Anticipatory: it anticipates the person's desires and environment status; this is context predictive and proactively enabler.

1.3.3 Ambient Assisted Living (AAL)

An AmI space, with an all around intelligence and equipped embedded devices, observes individuals without any inference with their lives and then serve them according to their needs, within a pervasive and transparent infrastructure. But we can design systems specifically fitted to provide care to the elderly. Ambient Assisted Living (AAL) is a context-aware environment consists of a set of ubiquitous assistive technologies embedded in the user's space to provide pervasive access healthcare services. It is a developing research area, in which AmI solutions based on context-awareness, interactive interfaces, adaptable environment, as well as decision making based upon Artificial Intelligence (AI), play an important role to solve the ageing problem as discussed earlier.

Using this technology elderly and disabled people can live independently in their own houses, even when their relatives are out for some time. KIT also includes PHSs, RMT, smart homes, assistive homes and smart cities, etc. The target is to integrate and improve the organization of healthcare providers, to improving rehabilitation, and to enhance security, care, risk prevention and QoL. AAL is a philosophy of care and services promoting health, independence and dignity.

In context to the human rights a research study done by Counsel and Care in UK have found that old people would prefer to live in their own homes rather than in nursing houses, hospital or elderly care centres. However they prefer and need

support to remain independent at their own home [114]. Researchers also proved that careful clinical therapy at a patient's home do not bring negative effects during the therapy process [126]. We need to provide them with the assistance within the required time and should measure their life vitals very quickly.

1.4 Related Computer Science Fields

Smart Home is an application of ubiquitous computing in which a home is monitored or affected by sensors or actuators, using Ambient Intelligence to provide comfort, healthcare or security services. The services which AAL systems provide can be further sub-classified as: (a) *emergency treatment healthcare services* to predict, detect and safety reactions such as alerts, (b) *autonomy enhancement services* deal with the assistance tools provided by the care providers, and (c) *comfort services* to provide relief, relaxation or comfort to improve the QoL [239].

For such a vast integrated field, the spectrum of computer science is very broad ranging from stationary systems to ubiquitous. The research shows that the *distributed systems* factor in combination with today's new technology (2016) is crucial, to develop such kind of Smart Home systems, and to meet the new demands of broad infrastructure. Advanced distributed networked systems build upon intelligent agents and cheap embedded computers or sensors as the system's resources are the fundamental building mechanisms to deal the above challenges [121, 38]. Following we have an over view of the fields or sub-fields that are most relevant to our goals and challenges.

1.4.1 Distributed Systems

A distributed computer system is defined to be a system of multiple autonomous processing elements cooperating in a common purpose or to achieve a common goal. "It seems that home intelligence will be employed in a distributed manner. This distributed intelligence may be applied in the form of smart devices" [38].

A distributed system must reflect the five main technical goals for its success, i.e. (i) easily accessible resources (by separating the applications from the platform with the help of a middleware), (ii) do not let the user know that the components are distributed over a network, (iii) it should be open to offer services according to a defined standard, (iv) it should be scalable and (v) adaptive towards new hardware integration and communication standards [354].

Most AAL applications are either too general or too specific, therefore a limited number of specification requirements should be considered while developing such a system within an environment for a specific user [240, 47], such as: the capability of the user, their accessibility, target platform's capabilities with respect to

the deployment environment and the healthcare device equipment configurations.

The *User's Capability* relates with the specific needs of each elderly person as he/she may have a different disease stage or many diseases and disabilities. He/She may also have different body structure, mental health, strength and habits. A user's needs may evolve over the time, such as different desires at day or during the night [240]. A user should have easy *Accessibility* to the services provided by an environment according to his/her needs as he/she may be ill or disable. For it the services or the devices should have intelligently developed user interfaces using which a user could enjoy his life and get comfort. The interaction component between a user and the environment is the most highlighting factor in this case.

In *Platform Capabilities and the type of the Environment*, we see that, same like an end user, an environment have many factors to consider, such as map, combination of available devices and their network, dependency between the offered services etc. An environment should be configured with respect to the end user and should not be too specific nor too general [240]. An environment could support device updates or faults. A user's behaviour or requirement can change at run time, or there could be a device fault therefore environment must show a dynamic behaviour. AAL applications are often deployed as pervasive systems therefore they provide services independent of the location. Whereas, *Device Configurations* points out the device administration and communication factor. As there are many devices so those should cooperate with each other and should identify and understand each other's messages during the communication operations.

Middleware

SHE are evolved by using different hardware and software components possessing limited open and distributed characteristics, hence incorporate heterogeneity. The integration and interoperability of heterogeneous demotic devices is achieved by an intermediary software layer called middleware [69]. Free-online Dictionary of Computing (FoLDoC) defines a middleware as: "*Software that mediates between an application program and a network. It manages the interaction between disparate applications across the heterogeneous computing platforms. The Object Request Broker (ORB), software that manages communication between objects, is an example of a middleware program*"– by *Denis Howe* (1985).

The role of a middleware is of a glueing mediator by staying between the OS, application program and the network to integrate heterogeneous resources [188]. An important goal of distributed systems is to separate applications from the platforms and then to allow them to integrate by providing a middleware, as depicted in Figure 1.3. The figure shows structured Remote Procedure Calls (RPC) or object-oriented Remote Method Invocation (RMI) as a glueing factor to resolve distributed component's heterogeneity. We shall discuss middleware and

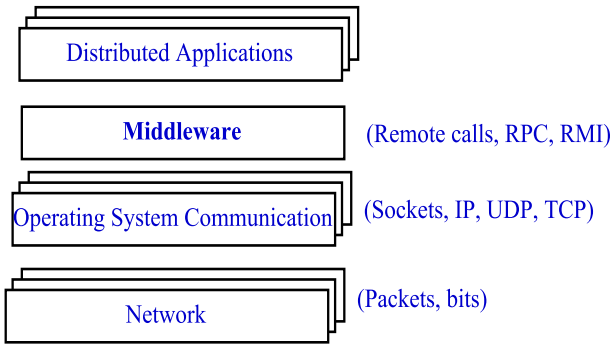


Figure 1.3: Middleware Layout [124]

these concepts in detail in Chapter 3. So these are the services to the software applications beyond those available from the operating system [124]. Middleware help decouple the software components, and Event Driven Architecture (EDA) is a famous architectural pattern (see 1.4.1), mostly applied during the design and implementation of systems which transmit events.

Architecture

This is the the logical organization of distributed systems into software components and their interaction [63]. The two perspectives for the proper organization of the components are: physical and logical. A software architecture is very important for the success of a system, as it tells us how the high level components are organized, how they interact with each other to exchange the data and finally how the components are joint with each other to make a single big system. "A component is a modular unit with well-defined required and provided interfaces that is replaceable within its environment" [371]. An important attribute of a component is that it can be replaced provided that the interface would be the same. The connectors between the components are important in context to the communication, coordination, or cooperation details [256]. Hence the procedure calls, data streams and message passing shape the nature of the connectors.

Architecture Styles

With such components and connectors we can define various configurations, which are classified as an architecture style. The four basic styles of architecture in distributed systems are, as: **1) Layered architectures, 2) Object-based architectures, 3) Data-centered architectures** and **4) Event-based architectures.**

The basic idea in the *layered style* is that components are organized in a layered fashion, where a component at layer L can call the functionality of the com-

ponent at layer L-1, so in this way the control passes from one layer to the other. *Object-based* organization is quite lose than this layered approach, where each object is a component and they can call to each other using (remote) procedure calls. A client-server architecture also works in the same manner.

In a *data-centered* style, the processes communicate through a shared data repository. The independent components access a shared data structure and affect or coordinate with each other through the shared data resource, which can be a shared distributed file system, or a database. Another example of it is in the form of a web based distributed architecture having a common data schema and the processes communicate through the web server.

In *event-based* architecture or Event-Driven Architecture (EDA) the processes communicate through the notion of *event* notifications. Such approaches generally associate to pub-sub systems [150], where processes publish the information by sending events and the other processes received that information only in the case if they are subscribed to such an event. One good advantage of this approach is that the processes are loosely coupled. Because of their independency from each other and having not referring to one another, so such systems can be referred as *decoupled* systems. If we combine the EDA and data-centered architecture, we can yield shared *data spaces*. The advantage of this shared space is that in this way the processes are also decoupled in time, as they do not have to be active during the communication processing. Such shared spaces can provide interfaces to the data repositories in a generalized manner instead of providing reference to each sub-component. This kind of architecture is very important in context to distributed systems and their distribution transparency.

Processing Modes

The different types of processing modes are: (i) *interactive processing*, is software that accepts input from the human being, such as spreadsheet, etc [320]; (ii) *transaction Processing*, is an individual, indivisible operation called transaction which completely succeed or fail, but does not complete partially [146]; (iii) *batch processing*, is a queue of independent program jobs on a computer without direct human intervention [354]; (iv) real-time processing, is the system having tight time constraints, like mission critical system, etc [354, 312]; and (v) *near or soft real-time* (NRT) processing, in which some delay may be introduced because of the data processing or network communication between the occurrence of the event and their usage [105]. It refers to the real-time web (see 1.4.3) in which an information is available for use to the users, as soon as it is published [237].

Considering the pervasive computing and architectural aspect of an AAL distributed system [376], some needed non-functional requirements, such as : interoperability, adaptability and distributed access points, are discussed as below:

Interoperability

This is "The ability of two or more systems or components to exchange information and to use the information that has been exchanged" [311]. Talking simply, devices or systems developed by different vendors are build and designed differently, so they might also communicate differently, regardless that they belong to the same type of usage. Interoperability relates with the compatibility of the devices or systems – to cooperate and understand each other – which lead towards the serious situations, such as in which we do not able to use two or more devices or systems together. Such heterogeneity in technology and communication protocols is very common during building SHE

Various types of heterogeneity in systems and services have been discussed in [300], and devices integration is an important requirement during SHE development. So researchers has developed solutions to solve heterogeneity between the software or hardware components. Some widely used solutions are, (i) promotion of middleware to solve heterogeneity [69], and (ii) the usage of standardization, hence supporting machine-to-machine (M2M) communication [46, 149]. They are responsible for: (i) interfacing with device drivers and (ii) to resolve heterogeneity between the software components. After their adoption by the industry vendors the devices have became compatible with each other, for example Bluetooth [74] and ANT+[1] wireless communication protocol enabled devices have own device profile standards to resolve heterogeneity and promote interoperability [119, 175]. Therefore, Bluetooth and ANT+ standard wireless communication protocols directly support M2M communication. Device interoperability has a vast affect, since by having the decision making capabilities and autonomous control, the M2M systems can be upgraded to Cyber Physical Systems (CPS) [381].

Adaptability

This is the ease with which a system can change or modify itself for use, within a given environment or platform, other than those for which it was specifically designed [311]. It relates with the flexibility of the system to adjust the change in the environment factors or according to a user's behaviour [328]. An AAL system must address this requirement and should provide its services according to a user's specific requirements, considering his disability, but should not also forget the network configuration and the relationships of the things within the focused environment. Patients vary in behaviour and profiles beside the progression or variation in the stages of their disease. These factors also effect in choosing a set of cooperating devices and assistive services for each patient. Different requirements of different users, according to their disabilities and capabilities have been

[1] www.thisisant.com

addressed in [230]. They also evaluate the requirements and the living spaces, including a dynamically adaptivity of a service without a user's intervention.

Distributed Access Points

The authors in [309], explain that distributed access points of an AAL system means the degree to which a component can be used by as many people as possible at different places with the focus of universal access points. This includes the system services, resources as well as devices. They argue that the developers should consider the importance of the end user interaction beside focusing on the system's main concept and the technical architectures and frameworks. A user may be involved during the system for an advanced and natural human interface for multi-modal user interaction and methodologies. Different users may have different roles so the services provided by the AAL system should be accessible from different distributed access points [239]. In this way all users will be able to participate in using the system and its services according to their roles.

1.4.2 Architecture Framework

An architect's responsibility is to design and describe an architecture [333]. An architectural artefact is a description consisting upon principles and practices in the form of documents, reports, analysis, or models; whereas, a framework is a skeletal structure that defines and describes suggested artefacts, along their different general relationships [333]. An architectural framework models or structures and architect's thinking by dividing an architecture description in sub-domains or layers; and latter offers models and views for guidance [393]. A framework's abstract skeleton guide an architect conceptually, through an architectural methodology in solving some or all problems related to architecture building for useful application development [333]. We have design patterns as core architectural elements of frameworks, to document and motivate effectively towards a meaningful utilization of a framework; and we can view it as a reification of families of design patterns to target a particular domain [155].

Software frameworks may be a bundle of support programs, libraries, APIs, compilers and set of tools; it bring all these components together logically to make enable the development of a project or solution. Software developers use an application framework consists of a software framework to implement the standard structure of an application [155]. However in this thesis we shall limit our scope and discussion to a conceptual general layered structure, rather than focusing on a standard structure for an application.

1.4.3 Smart Homes and New Generation Systems

This thesis is aimed to design, develop and assemble a multi-component flexible and scalable AAL framework which scrutinize the distinctive properties of a healthcare system within a distributed WSN environment with new challenges. New domains that need research while tackling with the new generation smart homes, beside those that have emerged while dealing with the distributed aspects, are, such as Internet of Things (IoT), Big Data and Event-Driven architectures over the real-time web. We should also review and design the targeted framework with respect to these emerging fields, to propose a robust solution that would be acceptable and utilizable in the real life of any person especially an elder one. These domains are discussed with respect to SHE, as follows:

Internet of Things (IoT)

Internet of Things (IoT) has emerged as a new paradigm, with the vision to connect worldwide things through the internet. The things are infact smart objects which are uniquely addressable, and are based upon the communication protocols, like Radio Frequency Identification (RFID) tags, Near Field Communication (NFC), to get integrate within the global network. In context to healthcare, such communicating smart objects can be body or environmental sensors which enable a continuous real-time monitoring. of the body or environmental parameters respectively. IoT systems, having continuous real-time monitoring characteristics, exchange and process volumes of heterogeneous data [375]. In 2014 a special IEEE report on "Internet of Things" described the phrase as: it is a network of items —each embedded with sensors—which are connected to the Internet [193]." This description addresses the physical aspect of this paradigm.

ETSI works for globally applicable ICT standards, including fixed, radio, mobile, broadcast and Internet technologies; although do not use the term "Internet of Things" in its document but describes a similar concept i.e. M2M communication [148], as: M2M communications is the communication between two or more entities that do not necessarily need any direct human intervention. M2M services automate decision and communication processes through standardization [194].

In most academic activities the different between CPS and IoT is not clear, but the difference exists. A CPS is a system of collaborating computational software elements controlling physical entities (i.e., mechanical and electrical components) [194]. In contrast, an IoT system have a start from the level of a single "thing" identification with a worldwide any time accessibility over the internet. Such identified objects can still be networked together, and if are controlled for a certain scenario in a coordinated manner; in such a case an IoT system can be considered to grow to the level of a CPS [194]. In order to build smart environments we

need devices that do not need any input from the human being and could able to identify and configure themselves according to pre-defined constraints. A M2M healthcare sensor may connect automatically with another device through internet. The discussion about the role of IoT in building SHE is delayed until Section 2.3.

Real Time Web

One of the basic requirements of a successful web based smart environment system is efficient processing and data transferral over a communication channel. Monitoring systems possess data-driven distributed components, where data controls the flow of the system and the delivery of the data to its consumers is required as soon as possible. The context data component can be an application or a context data middleware, as in our case. In healthcare applications, a real-time system is actually a soft real-time system, in which some latency is allowed [335]. Detecting critical urgent situations, such as a sudden fall or a heart attack and taking appropriate automatic measurements, such as calling a friend or hospital emergency, are always among the desired objectives of a healthcare system. Real-time Web healthcare systems involve exchange of large amount of sensor data, as well as bi-directional event notifications between the system components. The data arriving from the sensor networks will be large in volume and variety, where as the data from the centre location to the end user might be less in quantity and variety as this will mainly consist upon the control instructions or event notifications. So real-time data delivery feature is important and always required. Beside the real-time factor in the monitoring scenario, such systems transmit the data continuously upon certain intervals of time and temporal factor is one of the main property in sensor data payload. Such temporal factor is actually the order attribute in the transmitted observations by the smart objects. The new generation systems produce such data continuously and in abundance. Hence in the domain of healthcare we have to deal with the real-time data, in continuous or sliced snapshotted data streams containing time-stamp information with each reading.

Big Data

During long term monitoring processes of various domains (e.g. health care, banking, government or logistics) and sources (e.g. sensors, social networks or mobile devices) the acquired data is always in very large quantity and its storage and processing has always been an issue for the researchers, as described in [232, 171]. The previous methods of centralized or distributed storage with static schemas and unscalable infrastructures impose constraints on addressing the real-time, schema flexible and prolific data requirements. Such storage structures mainly rely upon the relational or structural approaches, which has given in for handling variety

of data [75], along side their resistance to horizontal scalability is a big issue for handling volumes of data [241]. These issues are discussed in detail in coming chapter 7 Sections 7.3. Therefore we need to find robust temporal oriented data storage mechanisms that can support real-time integration of huge volumes of differently formatted data for efficient query processing; and NoSQL approaches are keys to solve these issues [141, 117].

Therefore new generation systems want flexible data storage mechanisms which can accept different types of data arriving in real-time from millions of diverse data transmitting objects, thus enabling general robust storage for the data acquisition purposes is a main requirement of a smart environment.

1.5 Research Methodology

This is the analysis of the broad spectrum of methods used to solve the problems in the field of designing and developing distributed healthcare systems and their architectures. Following methods were used to collect information for the real world problem in order to achieve the aims and objectives. Literature review, field studies and overview of different existing healthcare use cases provided the foundation knowledge about the research domain and also the initial requirements for the first healthcare prototype. A study of Gerontechnology 1.2, SmE 1.3, and the related fields revealed that the research involves the monitoring mechanisms to measure the physiological parameters. SmEs have applications in other domains also such as energy grids, security, as discussed in Chapter 2. The thesis, the use cases and applications address the healthcare domain particularly; whereas the technical architectural goals are not limited to only one domain. The research as a whole, was a cycle of conception for an idea, practical experiment with evaluation and then describing the system with its functionality, methodology and architecture components. The conception process was always preceded with a literature review. These phases are described bellow briefly and their relationship is highlighted in the Figure 1.4.

Figure 1.4: Research Methodology

1.5.1 Literature Review

For the purpose of a brief overview we did a literature review in the domain of AAL systems and tele-medicine. This provided the foundation knowledge about the healthcare systems and their architectures along the identification of currently encountered challenges. This also provided the initial clarity and focus to recap the central research question of the dissertation. Next generation healthcare systems encountered with different problems which lie their solutions in utilizing and taking advantages from the state of the art ICT technologies. Therefore for the broad goal, of designing and developing flexible and scalable architecture for healthcare systems, the need was to leverage the state of the art technologies.

Diseases and Disability Perspective

During the initial stages of the research a survey about different diseases was conducted involving their categories and possible solution mechanisms. But to find the desired architecture we would not focus upon a specific disease, rather this was beneficial to have a general architecture possessing the state of the art technical features to process physiological and life vital sensor data in real-time, especially for the elderly society who wants always alive connections to be KIT. For the experiment purposes we focused upon the heart rate patients and used sensors accordingly. We defined a simple use case to solve an old patient's problem. As this research work focuses healthcare, therefore monitoring and KIT services are also beneficial for the fitness of people of any age. Hence, QoL and healthcare in general is common for both young and elderly society.

Similarly disabilities also vary with respect to the categories, and each disability demands a special treatment including the assistive technology. So this research do not directly solves the problem of a specific disability directly, however there are plenty of chances that the developed system's methodology surely contributes to the systems which want real-time processing of data and flexibility in their infrastructure.

1.5.2 Conception

An in depth literature study was conducted which overlap many different fields of research. In order to narrow down the broad vision and to have more focus, we set some research questions for different overlapping research fields, which lead us towards the initial requirements for the AAL system. Through out the research activity these research questions were evaluated in context to a set of hypothesis and experimental results. This conception formation and pragmatic research helped to move forward and to contribute in the field of computer science.

1.5.3 Experiment

Initially a conceptual idea with a small use case was planned, whom development and design leaded towards the first prototype which was a basic healthcare system. For this a set of sensors were selected, which were used according to the ICT principles to develop a system for a Window's operating system. After observing the experimental results more practical exercises were planed based upon the conception formation. Each experiment activity brought the development results, which was followed by a description activity about the system and its methodology.

Real-time Architectural Focus

The first prototype revealed that efficient delivery and processing of the biomedical data is crucial for the elderly people. Therefore in the second pragmatic development phase, the focus was upon the processing and storage of monitoring data in real-time. Finding an in general flexible and scalable architecture was the main goal which may involve different stakeholders, therefore in parallel to each successful practical experiment a study of architectural construction and methodology was essential. Applications of real-time architecture are not limited to the healthcare domain, rather are also helpful for energy grids and security domains.

1.5.4 Development Process Model

A process model guides the software teams in software development through a sequence of activities. It is a systematic way of accomplishing tasks and is defined as a collection of procedures, techniques, tools and documentation that help a developer to speed up and simplify the software development process [306]. In addition, this also defines the system development life cycle (SDLC) to develop a software product. A process may be liner, incremental or evolutionary [306].

Prototyping

Because the project phases may be iterative and the functionality of the system may be expanded over the span of time, therefore for the pragmatic research the prototyping process model was used, which is evolutionary in its development methodology. This guides the stakeholders by attaining the feedback and exploring additional requirements, if they are not clear in the beginning. The feedback of the first prototype identified the further requirements therefore this helped to foster towards the development of real-time processing of healthcare data by leveraging new generation technologies. As a result the second prototype was developed with a more robust architectural characteristics.

1.5.5 Theoretical Description and Architecture Modelling

The pragmatic research phases were followed with a theoretical description of the work-done in the form of publications. This also involved the architecture modelling of the developed prototypes along the description of the main components. Apart from contributions to the body of knowledge, this activity guided the researchers towards new directions for further advancements. These tasks were especially important for the practical experimental explanation and future anticipation. The system elucidation in the form of modules, components and their interaction mechanisms depicted the overall system architecture.

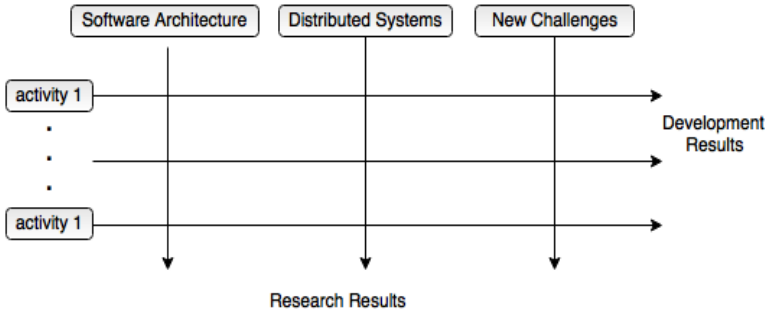


Figure 1.5: Development and Research Results

1.5.6 Development and Research Results

As shown in Figure 1.5 there were two type of results, i.e. (i) development results, and (ii) research results. The development activities were planned in the domain of distributed systems, system architectures and utilization of the sate of the art new generation technologies. Similarly the research results also bring up the description artefacts that overlaps such fields.

1.6 Research Questions and Hypothesis

This AAL research is to foster the exposure of innovative ICT-based systems, services and products for ageing well at home, in the community and at the work. The target is to involve the stakeholders like elder citizens, professional caregivers and insurance companies and the main desire is to improve the QoL, autonomy and comfort. This research is with many objectives to achieve and to tackle a lot of technical and theoretical challenges. In order to focus the research the formation of research questions is absolutely essential, as they are mainly used to narrow

down a broad topic. This thesis seeks to provide answers to a main research question that we wish to explore is as follows.

1.6.1 Main Goal and Objectives

How to design an AAL software system, depict its system architecture and framework by developing a prototype, having distributed system characteristics that could support a patient's context data acquisition, processing and storage for a flexible and scalable scenario, targeting the state of the art features of new generation systems, such as Internet of Things (IoT), Real-Time Web and Big Data.

This thesis is aimed at sketching a software architecture, methodology and framework, which reflect the properties of a healthcare system within a distributed wireless sensor network environment having pervasive characteristics. The further aim is to have temporal data modelling and storage of semi and unstructured data. These aims have the objectives with concrete attainments, which can be achieved by following certain number of steps. The research question stated above is the main question we want to answer; however such questions are not possible to answer without breaking them into further sub questions. Based upon the related computer science fields we studied in the Section 1.4, the research as a whole is differentiated among four main categories as: (i) to develop an AAL system and identify the building block mechanisms and components, (ii) to explore and implement the distributed system aspects of an AAL System, (iii) utilization of new technologies; and (iv) definition of an architecture and architectural framework. All new generation technologies are relevant, since it is a product development that meet many new challenges which the current distributed systems or networked applications are encountered with. For the exploration the new generation technologies, such as IoT, Real-Time Web and Big Data will be focused particularly. Given the above goals and objectives, we can develop our vision and the broad scope. In context to the main four objectives, following is a list of general questions which will be answered during the thesis.

Research Questions: Healthcare Monitoring System

HC1 What kind of Smart Objects (SO) or wireless sensors would be appropriate for a given AAL problem?

HC2 How to communicate with the sensor devices?

HC3 How to integrate the sensors to build a monitoring network?

HC4 How to process and interpret different sensor messages?

HC5 What components are required to develop a basic healthcare system?

HC6 How to acquire, represent, transmit and store any kind of sensor data?

Research Questions: Distributed Systems

DS1 What are the distributed system characteristics and how to have those in an AAL system?

DS2 What data flow phases are involved in a distributed AAL system?

DS3 Given the application domain, what kind of middleware will best integrate the different components of the system and how it will handle the flow of sensor or control data within those distributed components?

DS4 What communication means would be useful for the sensor data sharing with the context data server?

Research Questions: New Generation Technologies

NG1: How IoT helps us to deal with device heterogeneity?

NG2: The data-driven AAL monitoring applications produce volumes of data, then how to manage such prolific data?

NG3: Given the new trends how we can provide a general storage to any device?

NG4: How to handle timestamp streaming data coming from monitoring devices?

NG5: How a client or server be notified about a significant change in system state; and how stakeholders and different components will receive events?

NG6: How to provide communication for user critical services in real-time?

Research Questions: Architecture and Architecture Framework

AAF1: How the common technical artefacts can be represented as an abstract software architecture for a real-time HCM system?

AAF2 What framework skeletal and its underlying artefacts, would support architecting a new real-time SHE to meet the current technical challenges?

1.6.2 Research Hypothesis

A hypothesis provides a vision to look forward and is mainly presented or supposed by a researcher or a scientist after conducting initial studies about a topic. A researcher first collects data related to the topic and then conduct experiments to furnish them as a proof to his statements. A hypothesis may be the validation of a previous proposed theory or a new one with suitable justification. A hypothesis may be considered as a temporary supposition, as we see in different sciences. One is not sure either it true or false. If the hypothesis becomes true, he/she will accept it, otherwise will reject it.

Following is a precise discussion of the above research questions, step by step, and then formalization of some hypothesis that would help to look forward and to understand the research subject. This is to develop logical theoretical grounds towards conducting experiments to check the validity of the proposals.

Before establishing the possible aspects or the related hypothesis in context to the different questions stated above. Let's first explore the objectives of the main research question. The **first** objective is to develop a Healthcare system, which involves the identification of the main building block, available mechanisms and the required components. The **second** objective asks for the applications of the distributed systems and their characteristics within a healthcare system. How to attach the remote heterogeneous distributed components and to allow communication between them through middleware. The **third** objective fosters the development of the healthcare system, to meet the new generation system requirements with a provision of IoT, Big Data and real-time Web communication. The **fourth** objective is two fold: (a) the development of a general flexible and scalable architecture that would portray the properties of the developed prototype meeting the aims; and (b) the demonstration of an abstract skeleton guide, i.e. architectural framework, as a reference for other researchers to build architectures possessing same characteristics as developed during current research study.

Following we discuss all these aims with their specific questions and shall establishment the related possible hypothesis. Upon these hypothesis we shall carefully formulate our research experiments for the purpose either to accept them or to reject them. The main goal is to extend the knowledge boundaries after shaking them thoroughly through pragmatic means to check their validation.

Healthcare Software System

The first question **HC1**, in section 1.6, deals with the available sensor devices for the monitoring purposes. It demands a sensor technology that would monitor the life vital signs of a patient correctly and could transmit the measurements to the receiver. Depending upon requirements, a user may able to wear the device easily

on his body or may able to install it in his room. The miniaturization perspective of the wireless sensor devices is highly important in this context. Such a technology must be good in consuming less energy power; since the monitoring scenarios are for long durations and it is not possible to make frequent battery replacements.

This is natural to obtain data from the sensor devices in pervasive computing. We need flexible, low cost and efficient mechanisms to collect information, from the sensor enabled environments. Such sensors are with low powered energy and memory resources, therefore special importance is required to communicate with these sensors; with the objectives of low power utilization and optimized bandwidth maintenance. For the addressing and channel access control mechanisms, with in a network of the sensor devices, we need to follow some rules which are defined in the Medium Access Control (MAC) protocols. For this purpose different MAC protocols, such as Wireless Personal Area Networks (Wireless PAN) or 802.15 [195], 802.15.4 [31], S-MAC [389] and B-MAC [305] have been proposed and developed. Such a task also invokes the possibilities of making enable the control system, to be compatible of communication with the SO technology, may be using some secondary means, such as USB devices, wireless gateways. The second question (i.e., **HC2**) highlights the choice for such a protocol, which can help in enabling a system for communication with sensor technology.

The third question **HC3** deals with the usage of the wireless technology in an integrated manner. Because many sensor devices are applied as a group to provide the services in general. Such an integration of the devices have made the application developers much inflexible, due to lack of interoperability between thousands of drivers, ad-hoc APIs and protocols. Therefore, building efficient wireless networks using sensors is a highly technical task; and this involves the correct identification of each device and its easy seamless integration into the wireless network after the authentication. The devices would be integratable to build different network topologies according to the user requirements. In such a network, the sensors could communicate wirelessly at their specific message rate and within the network range. During the integration process, a network can allow communication to a device which belongs to it, hence to have network security.

Sensor devices transmit different kinds of data, and data interpretation is one of the required tasks of a monitoring system, and such a task is highlighted by the fourth question **HC4**. The fifth question **HC5** deals with the software modules which we may require to develop a healthcare system. Such components are of diverse functionality and depends upon the use case main goals.

The sixth question **HC6**, in section 1.6, relates with the representation of different types of data and its transmission to a distributed location for the permanent storage. Therefore it involves the mechanisms which will be used to transfer the data and the multiple formats of the data transmission. More it also involves the storage of the different data structures at a server in a robust manner. Such a stor-

age may be structural, like in relational database management systems (RDBMS); semi-structural or no structural, such as in graph databases [146, 277]. A database management system must support the storage to the sensor's as soon as the data is available or on batch bases containing data insertion set of jobs.

Hypothesis 1 Interoperable and energy efficient Smart Objects (SO) technology - allowing easy device portability, identification, integration and communication to build WSNs, with different topologies - will support the development of a pervasive HCM system for a patient.

Hypothesis 2 A general approach towards the device's message processing not only simplifies the data interpretation, but is also mandatory to have a specific unique module for this purpose.

Hypothesis 3 Relational Database Management Systems (RDMS) supports any sensor data format storage in any scenario.

Hypothesis 4 A software that understands different data formats at the client or server side may solve the transmission problem of different data structures using wrapping and transformation technologies.

Distributed Systems

The first question **DS1**, in section 1.6, is with the aim how to develop the healthcare system having distributed system characteristics. It deals with the over all basic layout of the system where the components are not at the same location and many of them have their own repositories to store data with own autonomously processing functionality having heterogeneous platforms [354]. The research question highlights that what technical goals a system must meet to be successful as a distributed system. This research question also points towards the domain application area i.e. AAL with local or remote healthcare. This also refers towards defining the theoretical foundations of a use case in the domain healthcare and then adopting pragmatic measurements towards solving it in context to the principles of distributed systems.

The second question **DS2** deals with the processing stages of the sensor data starting from its acquisition until its storage within a distributed scenario. It also involves the other data-driven stages, such as data processing, interpretation, transformation or representation, transmission and storage. The target is to process data of any sensor device within a distributed perspective.

The third question **DS3** deals with a platform that facilitates the application integration using a glueing factor between and within the different high level components and layers. Such a higher-level layer consisting of users and applications,

and a layer underneath consisting of operating systems and basic communication facilities, is said as a middleware [354]. Healthcare scenarios involve the situations that are highly critical and components need to sense and react quickly, therefore for this they generate different events. Beside this, the data and control flow between different components must be independent of the direct involvement of the client and the server, so it must not bind both of them together and should keep them independent to each other, such as the distributed coordination systems.

The **fourth** question highlights the communication perspective of the distributed systems. This asks to decide the type of communication between the heterogeneous components, which may involve calling the remote procedures, or remote objects or sending the messages in a synchronous or an asynchronous way. This also involves the connection types and data transmission speed.

Hypothesis 5 In the domain of distributed healthcare, a carefully designed adhesive middleware holds different components that overlap upon different system nodes. Each of the data processing stage provides useful abstractions to the developed or underdeveloped components, to cooperate with each other for the implementation of the applications that can access information and services from heterogeneous domains.

New Generation Technologies

The first question **NG1**, in section 1.6, deals with the problem of devices integration; which is an obstacle due to different so many vendors and their own standards; like different Application Programming Interfaces (APIs), communication protocols and drivers. Hence machines do not able to communicate to each other due to these heterogeneous factors. The question is how IoT helps us in this problem, and for this we can investigate IoT domain to find interoperable devices.

The second question **NG2** indicates the data-driven perspectives of the HCM applications, where the data defines the control flow of the system. On regular bases and in a periodic manner the monitoring applications perform measurements of the patient's body parameters or of the environment depending upon the use case requirement. Since such systems produce big volumes of data, therefore they also need data management systems that would able to provide management characteristics beyond the traditionally available management systems. They could accommodate bulky data and can scale upon additional data sets.

The third question **NG3** deals with the data format flexible storage features, where one can save messages coming from any kind of device with any format. Here we need different schemas for different data formats of a same device type, such as temperature, blood pressure.

The fourth question **NG4** points towards the handling of data streams, because AAL environments are data-driven applications which continuously trans-

mit signals. These signals hold timestamps information with them, like life vital parameters; therefore this question demands a solution which may processes the sequential data and stores it accordingly.

The fifth question **NG5** points towards the creation, identification, utilization of, and response to the events. The client and server must support the event driven based communication, where an event may generate another event or service. An EDA supports loose coupling and well distribution of autonomous components.

The sixth question **NG6**, in section 1.6, points towards the efficiency of the communication mechanisms of the system. Upon detecting a critical situation, the system must be able to communicate it as soon as possible to the relevant components or stakeholder. A system usually infer critical situations by using the context information and generating a relevant real-time event, such as fire alarm, medicine intake, fall detection etc. Therefore the system must support efficient delivery of these event notification to the distributed components in real-time.

This pragmatic research uses an evolutionary methodology in the form of prototyping, as in 1.5.4, which produces experimental results with every refined version, and this follows a phase involving system description and documentation. Therefore taking benefits from the new generation technologies may become more effective as the research progresses.

Hypothesis 6 Devices integration and interoperability, EDA, different data format processing and storage, schema-flexible storage of high volumes of time-series data are the problems whom solutions exist in taking advantage from the new generation technologies. The possible new candidate technologies are Internet of Things (IoT), Big Data and real-time Web.

Software Architecture and Architecture Framework

The first question **AAF1**, in Section 1.6.1, deals with the representation of a SHE software architecture, and involves the study of architectural styles and patterns used to depict a system. Architecture and software engineering patterns conveys a system as an image using high level components and the relationships among them [306]. Different architectural styles, such as modular, layer, central, distributed; and architectural patterns, such as MVC, EDA, pub-sub or RESTful are useful to design a high-level SHE architecture [89].

The second question **AAF2**, asks to depict the building block and their relationships as a framework, required to develop SHE real-time Web architectures. Such a conceptual view in the form of a layered stack may guide the architects to use this framework as a reference to solve same kind of problems. The layered stack contains different models or documents representing the framework, which is reusable to provide solutions to a commonly occurring problem. Following we list two hypothesis that would guide us during the research study.

Hypothesis 7 Try to cover the scope of the system by organizing the elements of the system and showing their relationships. Try to break the system into smaller sub-systems, according to physical and logical characteristics [].

Hypothesis 8 Selecting the key artefacts and depicting them as a layered stack may support the design of a conceptual skeleton as a reference framework.

1.7 Thesis Organization

The flow of the information context is organized in a manner that the thesis starts with in general discussion and leads towards the technical details of the implemented prototype. Which is followed by revisiting and analysing the goals that are set in the introductory chapter. The introduction and the overview chapters build a readers mind in a step by step manner to learn and understand the concepts and complex system environment gradually. The project involves many research directions and the work can be classified under different topics, such as M2M based SHE, middleware, real-time communication, and NoSQL storage of the sensor data. Although all these topics are different and the thesis cover them separately in different chapters, but a consolidated SHE survey covering all these dimensions is provided in a single chapter with respect to two main application domain, comfort and healthcare. This provide critical discussion about the existing studies and also talk about the potential contributions. The thesis is organized into five different parts. This introduction chapter and chapter 2 are in part 1; the latter is a review of systems and technologies used in building SHE. A comprehensive study of related developed systems and research is presented as an overview of smart homes, in Section 2.4. The chapter contains a study of conceptual architecture models and standards for healthcare information systems (2.5). The last portion contains fundamental concepts about WSN (2.7), especially Personal and Body Area Networks (PAN 2.7.1 & BAN 2.7.2). This follows with a discussion and comparison of energy efficient wireless technologies (2.7.4). The last Section presents ANT+ wireless technology (2.8), with a especial subsection about how to do development using this technology. Remaining parts are as follows:

Part 2 provides an in-depth study about related employed technologies in this research work. Chapter 3 discusses the middleware, and its role in building distributed HCM systems; by providing high-level services (3.3), such as RPC, Object Oriented Middleware, RMI, etc. Different SHE useful middleware standards (3.4) are given at the end. This follows with two chapters 4 & 5; where the former gives an overview about the Windows services along their pros and cons; whereas the latter discusses the role of MVC in building distributed applications.

The last two chapters of this part relate with the state of the art new generation technologies and their role in developing RHCM systems. The Chapter 6 discusses the real-time Web (6.3) and the Data Streams (6.4). In it AJAX, WebSockets and Comet technology is discussed, which is followed by a detailed discussion about real-time application development frameworks (6.3.5). Data streams section discusses streams data along the techniques of processing sequential data.

The discussion about Big Data, in Chapter 7, is highly valuable to manage high volumes of varieties of data. Section 7.3 discusses limitations of DBMS, and is follows with Big Data Management Frameworks (7.4) Section. To handle the temporal data the section 7.5 discussed the modelling techniques and provides a rational approach towards selecting different models. At the last it gives a use case study to build the grounds towards time-stamp based data models 7.5.4.

From **part 3**, the chapter 8 starts describing the pragmatic approach and presents an example of a company which tries to solve a health problem of a patient (sec. 8.1). For this purpose the company initially develops the first prototype (sec. 8.2), which although does not fit completely to the needs but provides conceptual grounds towards building a highly robust technical framework (sec. 8.3).

In **part 4**, the chapter 9 introduces in detail with the final research work by presenting the second prototype (sec. 9.3) that has a flexible and scalable architecture for real-time ANT+ sensor data acquisition and schema flexible storage of the ANT+ sensor data. The system consists upon two main distributed components i.e. *Window's Data Pusher Services* to acquire and transmit data to the *Node ANT Context Server*, which stored the data in a non relational database. A specially designed real-time communication middleware bridges the gap between both the parties (sec. 9.3). The next chapter 10 presents a temporal data modelling approach and its flexible evolution for a non relational database using an object modelling middleware service.

Part 5: This chapter 11 contains the summary and conclusion of the thesis along the discussion of the research questions and validation of the research hypothesis (sec. 11.1). This concludes the work with the future work perspectives.

Chapter 2

Review of Systems and Technologies for Smart Home Environment (SHE)

"The most profound technologies are those that disappear.
They weave themselves into the fabric of everyday life
until they are indistinguishable from it"
-Mark Weiser(1952-1999)

The purpose of this chapter is to present the historical as well as the state of the art current perspective upon the smart environments. It explains that building automation is the predecessor domain of smart homes. The origin of smart homes is explained along the importance of remote healthcare and Ambient Intelligence (AmI). This also discuss the organizations and classifications of Smart Home Environments (SHEs). Structural and information architecture depict the main components of such a SHE. There are many smart home projects that vary with respect to application domains and goals.

2.1 Introduction

Maintaining good health to have an independent for quality in the life have always been essential desire of a normal person. Alongside the population of elderly people is increasing quickly and is now a prominent aspect in modern societies which need attention from the administrative as well as from the research bodies with in any country. These societies are facing a list of issues, such as increase in diseases and disabilities [151, 53], healthcare demand and cost [380, 317, 160], remote home healthcare to avoid long hospitalization [76] and decrease in the

labour force [288]. Societies ranging from normal to patient or elderly, now need constant Health Care Monitoring (HCM) mechanisms.

The recent development in ICT research , such as miniaturization of electronic goods; increase in computing functionality and performance; and robust communication mechanisms; have influenced broadly upon the new ways of living. Now the focus is to design and develop supportive environments for older as well as for the healthy people based on the concept of Ambient Intelligence (AmI) [322]. AmI for healthcare is to provide solutions based upon efficient remote HCM services to the home users. This technical paradigm has triggered to design new protocols, communication mechanisms, systems and devices [355]. This as a branch of ubiquitous computing has the goal to embed smartness into a user's environment for comfort, healthcare, security and safety to have QoL. Alongside, the new low-cost, low-powered communication technologies made this possible for an ordinary object to become a part of human-centric networks, hence creating the Internet of Things (IoT) [392].

This chapter is structured as follows. Next section 2.2 is an explanation about smart homes with respect to their architecture, technologies and projects. It contains discussions about the basic organization of a smart home and the classification of SHEs. The IoT perspective in SHE is covered in Section 2.3. Section 2.4 is an overview 2.4.1 of SHE surveys, and a review of the past and recent projects in 2.4.2 having comfort and healthcare domains. Section 2.5 presents healthcare information systems with their models (2.5.2) and standards (2.5.3). This follows with a discussion about WSN in 2.7, WSN challenges in 2.7.3, and a special focus on energy efficient systems in 2.7.4. The last section is a detailed discussion about ANT+ protocol in Section 2.8. A summary is provided at the end.

2.2 Smart Home Environment

The concept of smart home is not new, rather it appeared in the beginning of the 20th century, even long before the revolution of ICT [58]. Whereas this vision has only came to reality with the advancements in ICT, such as sensors, networks, information systems, artificial intelligence, and now it is one of the main agenda of many nations. The end of 20th century and the start of 21th century is fully fledged and equipped with this research domain and the technology researcher have realized its importance to build the modern global societies with more comfort, care, security and QoL. Today (2016), there are many research projects with the vision to build smart communities, such as [107, 48, 108, 109, 101].

Alam et al. [38], provides the complete definition of a smart home, as :*"A smart home is an application of ubiquitous computing that is able to provide user context-aware automated or assistive services in the form of ambient intelligence,*

Figure 2.1: Smart Home Environment (SHE), source: TU Deft website

remote home control, or home automation."

The term "smart home" refers to a living place that is equipped with technology in its surrounding area that allows monitoring of its inhabitants for their independent living, safety, maintenance and healthcare [93]. It is much related to home automation or demotics; the difference is that, a smart home accounts the behaviour of the inhabitants and responds accordingly; where as an automated house focuses upon the automatic functionality of the day to day routine tasks, such as door opening, light control, fire detection, water and energy resources management. This knowledge about the goods and the user's behaviour is the context-awareness of a smart environment. However smart home possess more smart functionalities based upon AI techniques to address diverse high-level goals of safety, comfort and well-being of the inhabitants. based upon their preferences and behaviour activities. In [39], the early researchers provide an overview about smart homes with respect to history (starting from the 20th century's domestic technology), society and philosophical perspective. The emergence of smart home concept started from dating back to the 1960s, with the arrival of their automation with having wiring in homes to increase the functionality. The following extracts from a magazine shows that how the revolution moved towards this paradigm.

> "Advanced home control systems go by several names, including smart home, home automation and integrated home systems. By any name, these systems conveniently control home electronics and appliances including audio/video, home office, telecommunications, intercom, security, lighting, HVAC, and lawn sprinklers. Control systems can also provide information residents can find out how much electricity they have used on specific appliances or systems, and utilities can read meters remotely. The systems can be accessed from remote locations by phone or computer, allowing residents to turn on the heat, for example, on their way home from

work" (–Home Energy Magazine Online May/June 1998)

The efforts towards developing human-to-human interfaces using technology, in 1980s, was the driving force towards the creation of ubiquitous computing discipline, with the objectives to install technology in the environment during daily life [178]. Consequently, from the start of 90's, with the advancement in the networking technologies, along miniaturization of the devices capable of more computational power with lesser in size, gave a push to ubiquitous and pervasive computing and has created the current research trends of smart home research. The current era is considered as post-PC era, in which mobiles and hand-held devices are interacting with a user's environment to affect and share information. Moreover, in recent developments the factors like miniaturization, affordability, and de-wireization are the main driving forces behind the IoT trend [220], which imagines a connected network of all small and large different digital and physical end systems with identification and information sharing capabilities using latest communication technologies [392]. Following is provided a discussion about the organization of a smart home in sub section . This is followed by a study to classify smart homes based up the properties of its environment in sub section 2.2.2.

2.2.1 Organization in a Smart Home Environment (SHE)

Mark Weiser, the forefather of Ubiquitous Computing (ubicomp), defined a smart environment [385] as "the physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network". User-centered, controlled home-based environments are the bases of current or future AmI or AAL with the goal of QoL by providing local or remote healthcare services to the user, such an organization is depicted in figure 2.2 [93]. Such life caring, safety and entertainment services first collect the health status and activity data of the users, to transmit it to the health centers or hospitals in order to increase the elderly independency. Vital sign measurement device (such as blood pressure or temperature sensor) monitors the body or physiological parameter of a human body. The activity detectors, such as RFID tags or cameras, detect a user's movement and transmit it to build and analyse personalized behaviour. Leveraging from IoT, the Internet as a common global infrastructure (from any-time, any-place connectivity for anyone, hence anything [298]) may allow seamless integration of SO and their data transmission to build new different applications or services along their integration [392]. Beside this there are other social network based detectors to build personalized personalized profiles of the users with focus upon their tastes and preferences [156, 338]. A SHE consists upon the following factors [93]:

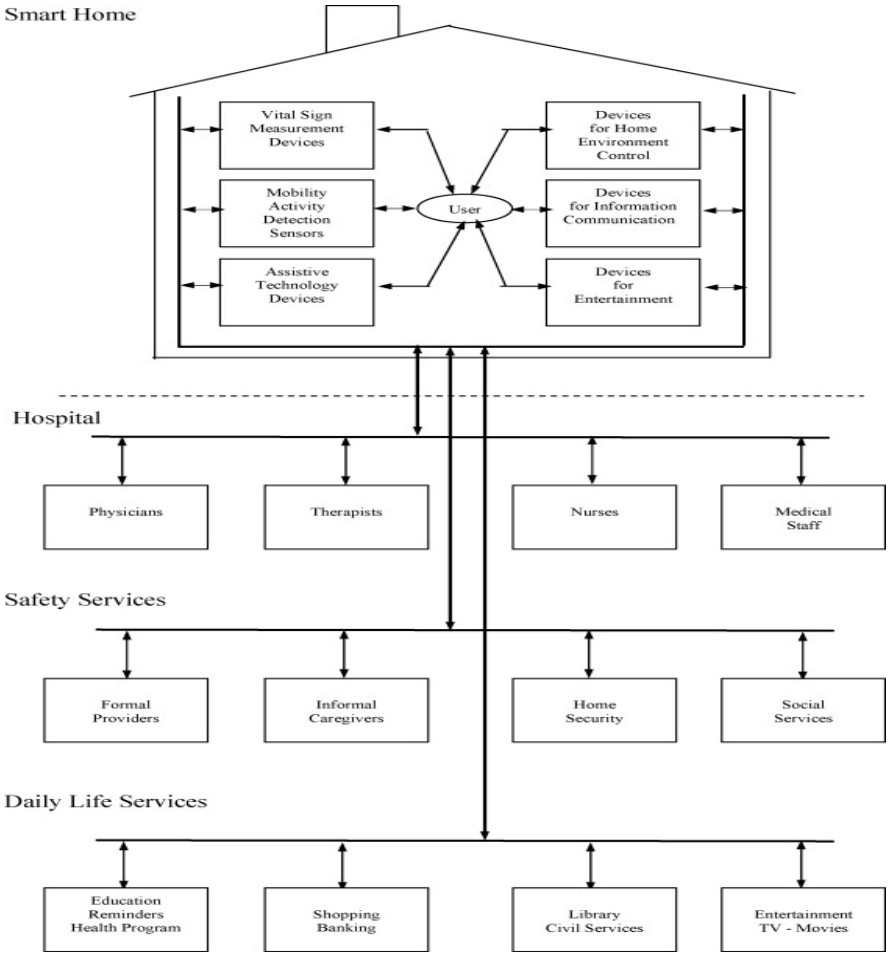


Figure 2.2: Key Organization in smart homes [93]

1. A hospital as main stakeholder, having physicians, nurses etc, provides virtual healthcare consultancies to the users.

2. Devices which sense, collect and transmit data about an AAL environment [293]. The different types of devices, such as activity detector, assistive, entertainment, physiological measurement, are depicted in the figure 2.2.

3. Some management services are only available in a hospital.

4. Patient receiving care regarding any disease.

5. The information center holding the patient historical record.

2.2.2 Classification of Smart Home Environment

Smart home is a multidisciplinary research area and involves the integration of networks, embedded systems and theoretical and applied computer science. These complex environments have different components, platforms, interfaces and processing capabilities [304, 210]. Following is a classification of SHE based upon the environmental characteristics [58, 38].

Devices based Home Automation System (HAS): It contains a set of wired or wireless home appliances and consists upon electronic devices to perform several automatic functions, like smoke detectors, televisions, washing machines, refrigerators, heaters and light systems, etc. These can be wearable, portable or implantable devices, such as activity detectors, heart rate and ECG monitors etc. Such devices can be categorized into sensors or actuators and stationary or mobile devices. A sensor performs functionality, such as monitoring, whereas an actuator performs actions within the environment, like closing an electric heater etc.

Control System: It works as a gateway to combine an ambient users with the environment. It uses the information send by the sensors and gives the instructions to the actuators to perform actions.

Home Automation Network: It is responsible for the proper communication among the SHE components, such as devices and control system. Mainly they exchange context information and the control instructions.

2.3 IoT Perspective in SHE

With respect to the IoT perspective a SHE allows smart objects within its environment to connect each other and communicate using Internet. Therefore, sensing environments enabled with WSN technologies are playing a vital role in the development of next generation systems. McEwen *et al.* [255] have strived for finding the components of IoT and condensed them into one simple equation:
Physical Object + Controller, Sensors, and Actuators + Internet = IoT

The total RFID tags sold in 2015 was worth $10.1 billion, with $9.5 billion in 2014 and $8.8 billion in 2013 [192]. The total RFID market will be worth $18.68 billion by 2026 [192]. With the needs of anywhere and at any-time communications, an approximate number of mobile phone subscribers were around 7.0 billion in 2013, up from 738 million in 2000 [206]. There were around 3.2 billion Internet users globally, with 2.0 billion in developing countries [206].

IoT with its vision and objectives has excelled the ensembling of different things into a common platform having a number of applications in many different fields. Smart Homes, Ambient Assisted Living (AAL), Smart Industries, Smart Cities and Smart Regions, Smart Car, Public safety, Smart energy and Agriculture

are the part of a future IoT Ecosystems and will attain popularity [392].

2.3.1 IoT for Smart Cities

Ubiquitous, Pervasive Computing and AmI are the heart of IoT [34]. Similarly, SHE has a vast impact upon the IoT paradigm, where ubiquitous technologies perceive their environment to offer personalized services to the end users [34]. Currently ICT researchers are with the broad focuses to develop Mega city corridors, and by 2020 we will see networked and integrated grand branded cities. The applications of CPS at a broader level by deploying state of the art wired and wireless communication infrastructure and novel applications/services over a city-wide scenarios [266]. In such a scenario the goals are always to optimize the city-wide management and organization and the ultimate outcome to achieve is the QoL of the inhabitants. Modelling SHE and then Smart Cities (SC) is a research paradigm of IoT, where Internet and other relevant technologies combined to develop SHE to offer life quality improving services [78].

There will be 30 mega cities worldwide by 2013, with 55 percent in developing economies, such as China, India, Latin America and Russia [375]. The requirements that emerges from a city would be transferred to solutions using IoT enabling technologies. FP7 Smart Santander project is with the aims to deploy IoT infrastructure across several cities [55]. Similarly, OUTSMART [291] project focuses upon utilities and environment of a city. BUTLER project [162] presents a smart city as with a target to integrate technology into a smart life.

2.4 An Overview of Smart Homes

Several SHE software applications, prototypes and projects have been developed, over many decades throughout the world, to support independent living with different ideas, focuses and functionalities. Their applications are being extending with special interests of the researchers, government organizations, hospitals and societies. This section provides a review about the recent surveys in SHE with a focus on the main building blocks and associated technologies.

2.4.1 Smart Home Environment Surveys

Solaimani *et al.* [340] visualizes the state of the art literature regarding SHE research in a systematic way and provide future research directions. According to them the research lies in four main clusters, i.e. service, technology, organization and finance (STOF domains). Technology being the biggest cluster has the maximum research publications, i.e. about double of the sum of the remaining all

three domains. Between 2003-2013 the number of technology domain publications has a rapid increase of about 600% than the number of publications between 1991-2002. They identified that the important technological aspects are: design, middle-ware, architecture and utilization of smart technologies in SHE etc.

Karami *et al.* [215], with a recent overview, discuss the state of the art SHE surveys during their proposal for a general architecture in which adaptiveness is based upon the activities of different users to form a personalized SHE. They simulate a SHE and theoretically propose an in-general system, to mark a user's preferences and his/her activities classification using DM techniques [353, 386]; whereas this research work emphasis the pragmatic research with actual sensors, for their real-time data acquisition and flexible storage. After discussing their in-general decision making process they describe the installation of sensors in context to their geographical locations within a SE, but don't discuss the interoperability issues nor their possible solutions. The data processing and acquisition process is simulated for ZigBee networks [341]; and stores the data into MySQL database [275] which has a static structured format and is less scalable. Our research architecture is flexible and scalable, along the real-time data acquisition and delivery. Currently it lacks the DM functionality for the user's preferences, but the architectural flexible supports further enhancements.

Alam *et al.* [38] describe a list of SHE projects regarding comfort, healthcare and security. They cover their main building blocks, such as sensor devices, communication protocols, AI techniques and the care services. Their survey summarize the local and remote monitoring for comfort and healthcare projects. The AI techniques from different domains are summarized in a table with their significance according to scope of their usage. Project's classification according to the applications domains is the most interesting. Chan *et al.* [93] provide a general perspective of SHE projects, with a discussion on wearable and implantable devices and assistive robots. They provide a geographical based overview about the SHE projects, which use SO technologies, decision support using AI techniques and automatic data transmission. But such a location based overview may not be interested for reader interested in architectural studies [38]. They also discuss advantages and disadvantages of SHE for the users and societies. In comparison to this, our focus will be more on the projects in context to their architectural aspects, M2M features of SO technology, real-time data handling within a distributed heterogeneous components scenario. DeSilva *et al.* [121] discusses projects that use audio-based, video-based, audio-visual-based, sensor-based and multi-modal techniques to detect user's activities. They discuss SHE in the area of HC, childcare and SG. Finally the research directions of multimedia retrieval for ubiquitous environments are presented. Badica *et al.* [58] present an overview of SHE with respect to their architectures, technology standards, applications and AI methods. They also classify different projects in four application domains, i.e HC,

EG, comfort and safety. They discuss AI based high level functionality, and provide a survey about the SHE projects based upon their architectural goals. Their approach relates to ours in resolving heterogeneity, as they propose the usage of middlewares for integration [354]. SHE are composed of three main constituents, i.e home automation system, control system and home automation network.

Salih *et al.* [322] discuss the importance of remote HCM to support the patients with Chronic Diseases (CDs) especially the elderly or disabled people. They emphasise upon the importance of AmI, AI and DM for a personalized healthcare to provide quick comfortable healthcare services. They consider the usage of wearable sensor technology to monitor elderly patients in the home as the most important building block. They provide a list of applications with focus on health, wellness, home habilitation and disease diagnostics. A summarised review over architectures and platforms for AmI application is also provided. They discuss the integration of wearable and ambient sensors in context of achieving home monitoring for older adults and patients with CDs.

2.4.2 Review of the Projects

Many SH projects have been conducted by different researchers with a number of focusses, functionalities and goals. This section presents a study of projects with respect to the application areas and the technological mechanisms. Figure 2.3 depicts the two main application areas, i.e. comfort and healthcare. Each of these domains need a critical overview with respect to different technical research directions, such as sensory devices interoperability (see 1.4.1), as devices' heterogeneity is a barrier in their integration; middleware/real-time processing, for the remote interaction of the remote components having EDA and real-time processing of messages between different components; and NoSQL storage, for the schema flexible storage of the volumes of sensor data. This discussion will also compare the previous projects with the work done during this research thesis. Following review discuss each of these domains along the technological directions.

2.4.3 Comfort

This class of SH projects address the entertainment, solace and comfort of the users with the help of local or remote home management. Optimization of daily activities by activity identification and usage if remote controls and assistive technology are the typical mechanisms. Automation of home appliances, such as lighting, heaters, music, automatic door system using voice or gesture based user identification are few examples of these. The following set of projects aim to automate the home for the comfort purposes, and we discuss them in a summarized manner in context to the directions depicted in Figure 2.3.

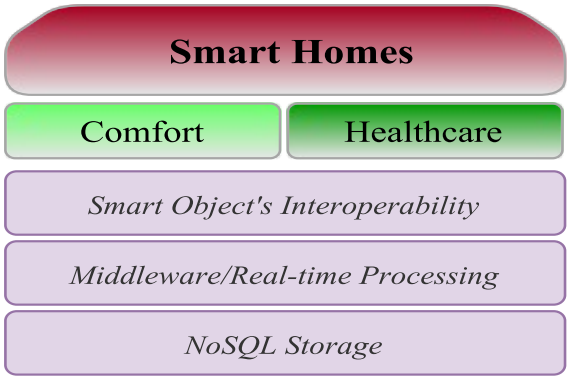


Figure 2.3: SHE : SO's interoperability, Real-time/Middleware and NoSQL storage perspective

Smart Object's Interoperability

We learned from previous discussion that interoperability (see 1.4.1) is achieved through developing either middlewares or adopting standardization approaches having direct M2M communication platforms, as in the case of Bluetooth and ANT+ sensor devices. Following brief discussion provide a list of projects in the "comfort" domain, and which have resolved heterogeneity using such approaches.

Peruma *et al.* [299] develop a prototype having a device management framework, which supports the seamless real-time integration of the devices into the system. Their prototype works for smartphones and M2M gateways, where as our research work is for Windows based services using ANT+ based M2M protocol. Their system also hold service analytics and device discovery modules, which our project misses; similarly their system misses real-time delivery and event-driven architecture. Daş *et al.* [120] uses M2M technologies in a SHE to automate an air-conditioner in-accordance to a temperature sensor plugged into a Raspberry Pi 2 board [301]. They develop an intermediate server between the Pi2 and the air-conditioner for the mediation. The project is at a very initial stage and lacks many features in comparison to our research, such as a methodology to support all the ANT+ sensors for M2M interoperability. Sayuti *et al.* [325] develop a general M2M system using TelG mote to acquire and process sensor data for IEEE 802.15.4 protocols [31]. They work for both comfort and elderly care domains and use a priority scheduling algorithm which allocate higher priorities to the healthcare data during message processing. Despite many valuable features, they dont provide a methodology to work with all the device profiles for the 802.15.4 protocols. Moreover they also miss a device management module.

Middleware/Real-time Communication

Middleware have large application spectrum with different goals, which involves integration, decoupling, interoperability and coordination; as discussed in Section 1.4.1; and will be elaborated along fundamentals and communication details in Chapter 3. This section discusses middleware developed for SHE for comfort.

Ma *et al.* [247] are similar to us technically, and they monitor a building environment in real-time instead of a human body. For the data display and delivery, they use WebSockets (also *Socket.IO* [25]) in the browser and for the data storage they also prefer NoSQL databases, but their focus is more in the security domain. The WSNs are composed of ZigBee and 6LoWPAN [334] instead of ANT+, and the M2M additional features for building vast ecosystems with device interoperability lacks. Resultantly, they don't provide a general approach to deal with all ANT+ device profiles; nor they use a Windows service for the data acquisition purpose. In context to the technical goals they are very close to our approach.

Managing an Adaptive Versatile Home (MavHome) project was developed with the goal to ease an inhabitant's life [391]; and this involves multidisciplinary fields, such as AI, robotics, mobile computing, middleware and sensor technologies. They use reinforcement learning to train their system for the identification and prediction of activity patterns; where as our prototype is still young and mining process for the predictive mechanism are still to design and develop. The system framework involves four cooperating layers as: (i) *physical* layer, for the hardware; (ii) *communication* layer, available to all layers for communication and service discovery. Operating system, device drivers, low-level component interfaces, device proxies, and middleware are the part its parts; (iii) *information* layer, gathers, stores, and generates knowledge useful for decision making; and (iv) *decision* layer, takes information for learning and making predictions. Their framework only relates to ours in context to the perception process, which is based upon a bottom-up approach. Both starts from the sensor monitoring and makes the information available through the communication layer to the other components. Both involves middlewares for the communication, but our middleware deals with the real-time processing and has an EDA, but MavHome's uses two types of middlewares: (1) CORBA, for object-oriented message passing in point-to-point scenario; and (ii) Zero Conf, for pub-sub architecture using OS sockets for the multi-casting, and is similar to us as it also separates the coordination from the main system logic and allows easy decoupling through pub-sub architecture. Their physical layer does not deal specifically with the M2M hardware, hence there will be no methodology for the device profile based data interpretation as in our case. Finally, they use PostgreSQL database for the storage, which is structured and inflexible in providing storage to different data formats, comparative to the NoSQL databases as our system framework proposes.

Microsoft's EasyLiving project [83] develops a SHE to monitor multiple residents, and for this uses architecture and technologies that allow integration of multiple I/O devices. Its components include as: (i) middleware, for distributed computing; (ii) perception, collects environment context information; and (iii) service description, to decouple device control, internal logic and user interface. The project develops *InConcert*, a middleware solution, to address the issues of communication, such as asynchronous message passing, and machine independent addressing and XML-based message protocols. Our middleware do not use XML based structured approach because it is inefficient and inflexible. Since our project is on going with development, therefore many features are still to develop.

Deriu *et al.* [125] works for a multi-platform or generic framework to enable interactive applications for distinctive environments. This uses a C++ programmed MOM for the data acquisition purpose to classify different situations with the help of the behaviour data. A major difference with our work is that they focused upon software framework where as our current targets are architectural frameworks. Bonino *et al.* [77] present an Intelligent Demotic Environment based on (i) DOmotic OSGi Gateway DOG, (ii) DogOnt ontology, and (iii) Semantic Web oriented rule-based reasoning to deal with the heterogeneity of different demotic networks and devices. Their approach is based open standard middleware (i.e. OSGi) which is quite different than ours. Similarly, Helal *et al.* [185] develop a SHE using knowledge representation mechanisms for the integration of actuators and sensors. The basic idea is to develop programmable pervasive spaces in which a smart space exists as both a runtime environment and a software library. Their service component is also based on open standard OSGi middleware gateway.

NoSQL Storage

Ma *et al.* [247] works for a real-time SHE and use NoSQL databases like us for the storage of big data. They also prefer document-oriented storage and for this use MongoDB database [259]. In context to the storage of the timestamped with the sensor data, they also don't prefer the first model for storage, in which a timestamp values is required to store with each record. They use a second historical storage model, called model with versions; in which all the timestamped data is created and tracked by having records for a given key in the dimensional table. Such records have different versions during storage. This system is discussed in the middleware and real-time communication subsection also (see 2.4.3).

During the discussion of the projects we learned that a lot of institutes and scientistic organizations are doing research in the same directions. The society want robust and efficient real-time systems having M2M interoperable features to accommodate all the relevant vendors; and finally to leverage the state of the art

big data technologies by using NOSQL storage mechanisms. The current project is a small step with these directions.

2.4.4 Healthcare

Many SHE projects are developed with the healthcare domain, and with the task to identify the health conditions of the users, ensure assistive services and to trigger local warnings in the form of alarms, such as fire alarm, water flowing alarm. Such systems analyse the user history data to find the long term trends of the user health. These projects are mainly to target the elderly people, patients and disable people, but these are not the only sub-domain as there are projects with the focus of fitness, well-being, and sports. We shall treat them similar in this section. Many researchers have put their efforts to build both local as well as distributed components to support healthcare; and following Section will provide an overview of them in context to the three dimensions depicted in Figure 2.3

Windows Service for Data Acquisition

There are many data acquisition and managing healthcare systems and are discussed some of them in this overview. The chapter 4 list few Windows services, and to the best of our knowledge, a methodology has not been provided by anyone to use Windows service for ANT+ sensors' data acquisition and management. Such services have proved their effectivity in context to the long term monitoring operations, as discussed in Chapter 4.

Smart Object's Interoperability

This sub-section provides an overview about the interoperability, which is the seamless communication, understandability and together working of hardware or software components resources; as we discussed in Section 1.4.1.

Johansson *et al.* [212] develops a biofeedback system for balance control system for the post stroke and gait related rehabilitation patients. In contrast to the approach presented in the thesis, which uses a Windows services for the monitoring purposes, these researchers develop a WSN to provide remote E-health services using mobile technology. They use two ANT+ sensors, however our initially developed prototype uses three sensors and monitors more than 8 features of the user. Their prototype possesses a modular architecture with its interoperability and storage features, which make it distinguish from the rest of the developed systems. They don't provide a methodology to deal with all the device profiles of ANT+ sensors, or any other M2M technology.

Valchinov *et al.* [373] present a system which monitors a patient's cardiac activity in real-time at home using ANT+ sensor technology, but they do not present a methodology for handling all the device profiles. For the data acquisition purpose they don't use a Windows service like us. In [57], the researchers develop a mobile based system which monitors the users who are mostly interested in sports. However our prototype is focused on heart patients and uses parameters, such as temperature, weight, distance travelled, cadence, footsteps taken and calories burned. They do talk about devices interoperability and provide seamless connectivity to some sensor devices of all ANT+ vendors, but they don't provide any methodology to support all the sensor devices. Our research prototype is flexible and have reusable adaptable components to support scalability. They use HTTP based connection on occasional bases to connect with the server, hence is not real-time nor bi-directional. The system in [57] facilitates the doctors to provide feedback, which is an additional feature; and we planned to extend out prototype with such features in future. In [339], the authors have described the implementation of a wireless BAN for hip rehabilitation care called HipGuard. They support a M2M technology but misses a general approach to handle all the device profiles for ANT+ protocol.

In [346], the researchers provide interoperability based upon HL7 (see 2.5.3) standards, i.e., CDA and XSD; but our prototype misses this standardization approach. Their system work for ANT+ protocols with a focus upon getting mobile based data acquisition and provides feedback. They don't provide a general approach to handle all the ANT+ sensors.

Middleware and Real-time Communication

Kozlovszky *et al.* [224], acquire cardiac and diabetic patient's data, using many protocols, such as ZigBee/XBee, Bluetooth, and ANT+, with the help of a local data acquisition device called DataHub, based upon a Generic Sensor Network Data Acquisition (DAQ) solution, which sends the data to a distributed context data storage component called Monitoring Datacenter (MDC). They have used both schema approaches, namely structural or predefined schema and non-structural or schema flexible, in the form of MySQL [275] and MongoDB [98], respectively. However, they do not use WebSockets' based communication within an event-driven architecture. Hence their middleware does not provide the features as in our case. Besides this, they do not provide a general methodology to deal with all the sensor device types for any single targeted protocol, such as ZigBee/XBee, Bluetooth, and ANT+. Whereas they focus on cardiac and diabetic patients. The provision of a multi-protocol solution with visualization and alert signalling approaches are their additional features.

Berndt *et al.* [65] enable the implementation of a complex system, by conceiv-

ing a flexible Telematics Platform, which guarantees secure real-time communication and visualization of the fitness and stress data for both mobile phone and for a website. They use Bluetooth sensors to transmit data to a mobile based gateway, which further transmits the data to a processing database via UMTS/GPRS. Later, the data is stored in a middleware, a Telematic Platform, which allows integration of other applications and services. For predictive analysis, they propose a new fuzzy logic based stochastic method to model the stress state of an end user. They are similar to us in context to the platform development and its scalable features, however, they do not handle ANT+ sensors. The sensor data storage is not in real-time and the storage server uses a pre-schema definition approach. Zhang *et al.* [394] enable the development of a scalable, real-time, multi-parameter, remote healthcare monitoring system, based upon HTML5-WebSockets web communication technology. However, the system does not address an in general data acquisition and flexible schema storage features. With further advancements, Zhang *et al.* [395] builds a context-aware mHealth System, for the remote monitoring of physiological patient parameters, targeting fitness and stress management application domains. The system provides the technical grounds to perform conditional evaluating of the physiological parameters and also provides two main components: (i) an online activity recognition algorithm, and (ii) a predictive model, based upon conditional-reasoning knowledge base, to filter potentially dangerous abnormal heart rate instances. Their system supports both Bluetooth and ANT+ sensors and runs on a mobile platform. The system acquires the data in real time and transmits it in near real-time to the context data server, using HTTP and Sockets. Their server is more mature, in context of the service provision and predictive data mining based decision support, however, they do not provide a general methodology to support all the sensor device profiles.

Jovanov *et al.* [213] uses ANT+ heart beat and foot pod sensors for the real-time stress monitoring to help the nurses. They process and analyse the ANT+ data in real-time for an iPhone locally, and transmits the data to the server using an internet connection. Technical details are not available how they transmit in real-time to the server, or either they transmit using batch processing. They also maintain a local database for the data storage. In context to the storage they do not use a flexible schema approach like us as they rely upon relational databases. They do not use windows services or an in general ANT+ messages data interpreter. Their efforts are more for the data processing and analyses along the questionnaire based assessment of stress causes.

Gharghan *et al.* [168] provide a survey about monitoring systems based on PANs, using three common wireless communication protocol standards - i.e., Bluetooth [396], ZigBee [250], and ANT+. They focus on fitness in general rather than the core body parameters; such as blood pressure, heart rate, etc. They review, present and compare several communication solutions; such as low power

consumption, long distance communications, small size, and light weight. After observing the examples, of an advanced and adaptive network technology (ANT), the authors conclude that: ANT+ protocol is better due to reduced power consumption and prolonged battery life. Furthermore, during their later study in 2015 [169], they repeatedly prove the energy saving advantages of ANT+ protocol over the ZigBee protocol; and for it, they conducted corresponding experiments within the communication range of 65 and 50 meters (m).

Strasser *et al.* [346] transfer ANT+ sensor measurements to a middleware called JFrogman, which is proprietary and is responsible to send the data to the server which accepts HL7 messages data. A mobile device communicates directly with the server using RESTful interface through passing JSON message objects. The AssistiNG ELders At Home-ANGELAH project is with the focus to help the elderly with cognitive impairments and it provides ubiquitous assistance for the in-house safety and autonomy [351]. This project is infact a middleware-level solution, which by using the networking technologies integrates both "elder monitoring and emergency detection" solutions. It is with two main features, as: (i) to enable devices (sensors, actuators) integration using OSGi [200] technology, and (ii) to provide a framework containing community subgroups whom could take responsibility of taking prompt steps to care elders in case of emergency situations. Its middleware architecture, implemented on top of the JVM, consists of three layers: (i) monitoring and assistance layer, integrate the sensors and actuators using the OSGi middleware infrastructure and processes the sensor data ; (ii) response management layer, sends signals to the volunteers upon receiving any emergency signals; and (iii) group collaboration layer, dissolves and manages the emergency response groups in a wireless environment.

The Context-Aware Service Integration System (CASIS) [211] project addresses the elderly society to improve the QoL using Ubiquitous computing. This multi-agent service framework is an event-driven service-oriented framework to integrate different services and OSGi infrastructure. It provides remote monitoring to the patients from their physicians, family members and caretakers by allowing remote view to an context measurements through web enabled PDA or PC. The project is not having real-time bi-directional communication features.

NoSQL Storage

Some related work in context to data modelling for the NoSQL databases will be pointed out very briefly during the chapter 7 in Section 7.5. A lot of data modelling studies has already conducted by different researchers for NoSQL databases, but non has provided in depth study for the ANT+ sensor data especially to preserve the data based on temporal properties, therefore this research is novel in context to time series ANT+ data. In this remaining Section we shall analyse and describe

some relevant research specifically.

The authors in [50] and [218] make models for the NoSQL databases, such as MongoDB; and present both relational and non-relational database queries to have a comparison between them. The latter research presents more general results. However they do not deal with modelling for the temporal aspects. Similarly Bugiotti *et al.* [84] design for a selected NoSQL framework, based on best practices and guidelines. They provide a methodology, independent of the specific target system, which depends upon the initial activities of software design. They make novel data model for NoSQL databases, named as NoAM (NoSQL Abstract Model). After outlining the commonalities of various NoSQL systems, they specify a system-independent representation of the application data. They treat collections separately as abstract model or table, as in this system.

We will discuss in Section 7.5.4 how to model time series schema design in MongoDB with different variants during the thesis and prototype development [295]. The authors of [208] and [294] use the same approach but for different application domains.

2.4.5 Conclusion of the Overview

During the discussion of the projects we learned that a lot of institutes and scientific organizations are doing research in the same directions. The society want robust and efficient real-time systems having M2M interoperable features to accommodate all the relevant vendors; and finally to leverage the state of the art big data technologies by using NOSQL storage mechanisms. The current project is a small step with these directions to make robust and efficient SHE for the society.

2.5 Healthcare Information Systems (HCISs)

Healthcare is an information intensive industry, in which reliable and timely information is a critical resource for the planning and monitoring of service provision at all levels i.e. organizational, regional, national and international [245]. Series of literature review highlights that, in most of the cases it was the technological-pushing forces that had triggered the health care organizations tend to adopt ICT-based solutions without any discrimination [87].

2.5.1 Healthcare Information Systems (HCISs)

They are to acquire, process, communicate and store the life sign information to the decision makers in a timely manner for better coordination of the healthcare. Nowadays, healthcare providers are facing the issues, such as to improve care,

quality and effective treatment; and they are under the pressure of competition also beside to reduce the treatment costs. This thing has changed their behaviour [113] and have motivated them to adopt innovative Information Systems Modelling Techniques and to allow ICT to improve the quality of healthcare and the treatment processes. ICT in Health Care Observatory (IHCO) is an institute with the focuses on the analysis of ICT-driven innovation in the health care industry, with a specific emphasis on HCISs in Italy [113]. Its objective is to promote research through multi-company collaboration frameworks.

Rodrigues et al. define HCISs as, "these are powerful ICT-based tools able to make health care delivery more effective and efficient" [316]. Rada et al. consider HCISs as the application of management information system in health care. Therefore, HCISs consist upon different applications that support the need of health care organizations, clinicians, patients and policy makers in collecting and managing all data related to both clinical and administrative processes [310].

2.5.2 Architectural Model

The three levels of functionality [113], which HCIS provide, are as

- *Central government at national and regional level*: It involves resource management, rules to follow, procedures to follow, financial management, monitoring of quality and safety. Each national healthcare system may have different organizational structure at the regional level in different countries.
- *Primary Care Health Services*: These are general services that must be available to the community members at a national and regional level. It includes all the service providers like general local practitioners etc.
- *Secondary Care Health Services*: This is the level of the healthcare services that are provided by the healthcare providers. HCISs must reflect the functional characteristics as working at this level also so that the individual health providers must able to perform their processes also.

The most significant factor among all these three layers is the data exchange, which make ICT essential for data acquisition, modelling, sharing or communication at a larger scale. It also demands that HCISs must reflect scalability and flexibility both in terms of data format or record types as well as its volume. It is not just to bridge a relationship between a patient and his physician rather it is more than this and involves the data management at larger level in context to its efficient and robust acquisition and transmission. .

For example, Italian healthcare system is completely public, which provides healthcare services to the community members as their basic fundamental right [113]. The different Italian HCISs have been based on a set of pillars [244]:

- The digitalization of the information flows at a national and a regional level.
- The development of a national as well as regional social security card (RSSC).
- The regional infrastructure development to support online services.
- The development of a strong set of interconnections between health care providers (secondary care) and general practitioners (primary care).
- The creation of a regional *Electronic Health Record (EHR)* to be subsequently integrated at a national level.
- The digitalization of the service-delivery processes in secondary care.

The healthcare providers become more fragmented and evident when the duties of financial as well as quality assessment factors become the responsibility of the regional bodies, as it is the case in Italy. An individual community member with his RSSC may able to access the healthcare services, and the network linking between the health care providers, general practitioners and regional entities will help track the EHR of the patient and the financial matters.

Information Architecture

Information Architecture (IA) is the structural design of shared information environments. It represents the higher level abstraction of the information and puts emphases upon the situation awareness of the environment. It also deals in terms of how the subcomponents interact according to the execution of the processes. Although there is no widely accepted definition but for the understandability following is provided:

- "It is defined, as a professional practice and field of studies focused on solving the basic problems of accessing and using, the big amounts of information available today" [314].

Hospital Information System and Architecture

ICT's most contributed factor in HIS is its provision of efficiency and reduced cost, especially during the information flow between the core or secondary processes. A HIS does not have just to manage the flow of information and the complex processes and their interlinking but has also to store and make available the information required by the doctors, nurses and physicians [113]. Caccia et al. [90] describe a classic value-chain model with two main process, in order to describe the organization of the processes of a general health care provider.

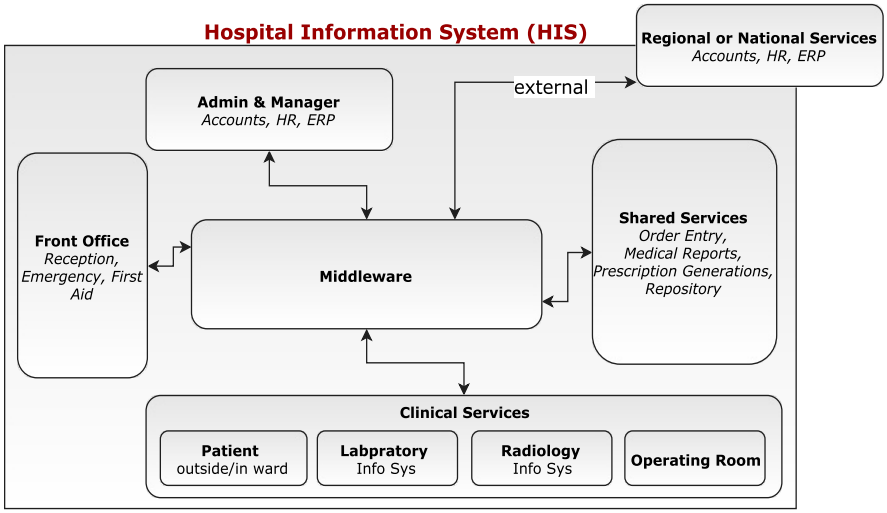


Figure 2.4: Conceptual Architecture of HIS [113]

- Primary processes: relates to a patient's direct care, such as *admission, anamnesis* (Patient clinical situation), *diagnosis* (defines therapy or care plan) and *discharge* and *follow-up*.

- Secondary processes: these are the supporting processes involving the technical aspects, such as *strategic* services (planning and controlling, supervision of regulations), *administrative* services (finance, HR, quality and IS) and *technology* services (machinery, biotechnology, building automation).

The figure 2.4 describes a comprehensive view of the different components that interact with each other through a glueing mediator component, called as: ***Middleware***. It is responsible for the interlinking of all HIS components using a data format standard. This component is the only one among many in which a pervasive infrastructure is required, that make enable the implementation of an integrated technological platform with the goal to digitally manage the healthcare activities locally or remotely [113].

2.5.3 Healthcare Standards

There are certain standards, such as Health Level 7 (HL7[1]), which are developed for the interoperable working of different applications developed by different healthcare providers. HL7 refers to a set of guidelines for the exchange, integration, sharing, and retrieval of electronic health information of clinical and

[1] http://www.hl7.org

administrative operations between the software applications. These standards are applied by working at the OSI model layer 7, i.e application layer. Following is brief discussion about some of the HL7 standards [204].

- Clinical Document Architecture (CDA) an XML based exchange model for clinical documents. Its types include discharge summary, imaging report, history and pathology report. Integrating the Healthcare Enterprise (IHE[2]) is a non-profit organization, based in the US, with the initiatives to improve the ways computer share health information. IHE offers a model for cross enterprise document sharing called Cross Enterprise Document Sharing (XDS), that allows hospitals to share electronic records.

- Fast Healthcare Interoperability Resources (FHIR) - a next generation standard for the exchange of resources and supports RESTful architecture. Resources are the modular components easily integratable in to a working system. This has strong implementation focus by leveraging HL7 v2, CDA and web standards, such as XML, JSON, HTTP, OAuth, etc [203].

- Clinical Context Object Workgroup (CCOW) an interoperability specification for the real-time integration of user applications. It is vendor independent and allows applications to present information, within all opened applications at a user interface, according to a single patient's perspective. The primary "Context Management" adjusts a particular subjects of interest, with which different applications are virtually linked and the user see them altogether in a unified cohesive way after a single authentication [202].

2.6 High-level Component Architecture

High level programming languages and Abstract Data Types (ADT) give the urge to the programmers to think even more in an abstract manner upto to the level of a component. This is the reason that Garlan et al. start introducing component architectures as: "To place the field of Software Architecture into perspective let us begin by looking at the historical development of abstraction techniques in computer science" [164]. In a software architecture we see the components and their organization at a macroscopic level.

Bass et al. define the software architecture as : "The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. Architecture is concerned with the public side

[2]http://www.ihe.net

of interfaces; private details of elementsdetails having to do solely with internal implementation-are not architectural". Hence, the identification and definition of the main system components and the interaction among them constitute the high level component architecture of a system. In a SHE the computational capabilities as well as the resultant service provision highly influence the architecture capabilities of such a component [58].

An architectural style, also called as an architectural pattern, is a set of principles applicable to a frequently occurring problem to promote design reusability. A pattern provides a framework to the systems of same characteristics [262]. The famous two architectural styles are: (i) centralized, and (ii) distributed. In SHE with the influence of ubiquitous computing and the requirements of the remote monitoring the system architectural styles are now shifting from central to distributed [58]. The computational requirements of SHE are actually the goals of the Control System [58]. So the architecture of a SHE is actually the architecture of the Control System with the control and manage capabilities [58]. Following we discuss the two architectural styles with respect to SHE.

2.6.1 Centralized

In a centralized SHE, the component that is responsible for the data acquisition, user interaction as well as the control algorithms is the Controller Component of the smart home system [58]. Such a centralized component also interprets the data and control messages of the devices and also sends signals to the sensors as well as the actuators. This central computer of a SHE is called as Home Gateway [388], which allows the resources to interact with the internet as well as provide services to the home residents. It also collects the context data about the environment for storage and analysis purposes. Such centralized servers some times do not perform extensive data processing tasks, rather they share the data with more powerful computers outside the SHE, using bulk/batch or other non real-time transmission capabilities.

2.6.2 Distributed

In context to the definition of Distributed System in section 1.4.1, a distributed architecture in SHE is where the control is spread over the network nodes. In this architecture the independent processes or components, at different physical locations, execute common tasks through coordination based upon event or message oriented communication.

Distributed systems have three different kinds of models [69], such as *physical models, architectural models* and *functional models*. The first relates with the physical layout of the system, where as the second i.e. architectural model defines

the system in terms of aggregated computational elements that are interconnected within the network. And the last one, functional model, as the name depicts, is an abstraction upon the functionality of each refined component. All the SHE projects mentioned in section 2.4.2, except than the local monitoring or centralized architecture are actually having distributed system characteristics.

In a SHE the sensor devices and the heterogeneous distributed components, all posses different special as well as functional properties. Hence they need to co-operate and communicate to perform their tasks, inspite of the fact that they may have different operating systems, different functional services or use different platforms [370]. The solution to solve this heterogeneity and to enable cooperation and communication with in a synergy is the use of an effective *middleware* layer.

2.7 Wireless Sensor Networks (WSNs)

Smart Environments (SEs) represent the next generation system development steps towards building industrial, home, transport and energy automation systems. Such SEs first and foremost rely upon sensory data from the real world, which comes from multiple sensors of different forms in distributed locations. The field of WSN provide a prosperous, multi-disciplinary research area in which a large number of mechanisms and concepts can be applied to deal a large number of applications. To build robust WSNs, the scientists have to face a variety of challenges, such as limited amount of energy, short communication range, limited processing and storage in each sensor node [99, 238].In SHEs an array of sensors are applied in an integrated manner [99]. Their integration builds up a sensor network; such a network forms different topologies, which vary from star network, tree to multi-hop wireless mesh network [242, 99]. The sensor data messages generated by different nodes travel within the network; so their routes directly depend upon the complexity of the underlying network topology [318]. The data flows through the nodes until it reaches a special node called the base station, which links sensor network to another network for further data processing purposes. A single network may consists upon many interconnected sub-networks of different topologies. Additionally networks can be classified as Local Area Networks (LAN), e.g., within a building, school or laboratory; or Wide Area Networks (WAN), e.g., between different buildings, institutes. WANs are mainly used to connect the LANs; and for this purpose they use gateways, which work as a bridge between different networks. Home Area Network (HAN) is an examples of LAN, which is mainly used to build SHE (see 2.2.2), whereas an example of WAN is Internet, which being an IoT platform facilitates SHE development.

2.7.1 Personal Area Network (PANs)

Zimmerman [397], in 1996, presented the concept of PANs, in which he allows the electronic devices, on or near the human body, to exchange the digital information. Arguably, such a definition did not enter the mainstream IT industry until the introduction of Bluetooth wireless technology in 2000 [175]. Therefore in WPAN the devices operate within an individual person's workspace over a short-distance, approximately within the range of 30 meters (m), where different devices can be interconnected to each other, such as personal computer with wireless mouse and keyboards etc. Such WPANs also use environmental sensors, deployed in a patient's room, such as temperature, video cameras, or sound, pressure, RFID readers, luminosity, and humidity sensors. They may also contain activity detector sensors, which detect the movement of a user and transmit the data directly.

Within a PAN range, a wireless "Personal Assistant" device, which supports the communication with the other devices, may have a controller module and context-awareness information with a user-friendly interface, can allow any user or a patient to control the network [40]. In such a case having a modular system design and scalable network allow different users, devices and functionalities to get integrate easily [40]. An extended forms of PAN are HAN, Body Area Network (BAN) and Vehicle Area Network (VAN) [175] and implantable or wearable sensor devices are being indispensable key components to build up such networks. Institute of Electrical and Electronics Engineers (IEEE) 802.15 [195] is a working group of the IEEE 802 standards committee; which works for the standardization protocols for the wireless personal area network (WPAN) [31]. An overview of WSN applications in the domain of healthcare is presented in the figure 2.5.

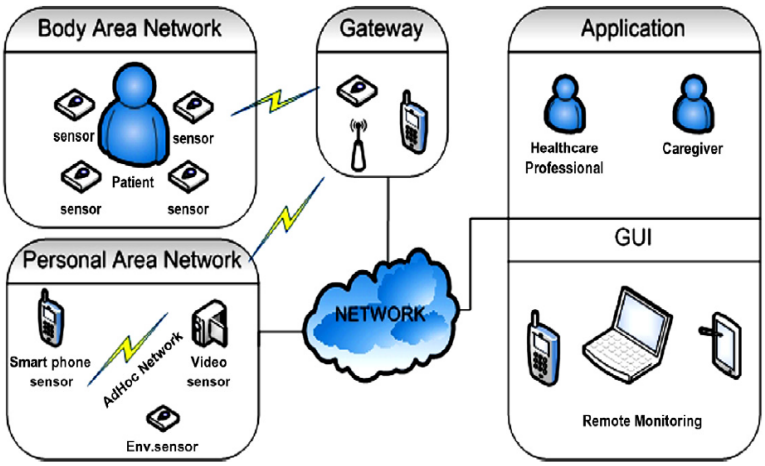


Figure 2.5: Simple WSN applications' overview in healthcare domain [40]

2.7.2 Wireless Body Area Network (WBAN)

In parallel to PANs, the Body Area Networks (BAN) has emerged as a topical and popular topology that particularly characterize the wearable technology to provide real-time health monitoring of a patient [175, 96]. Using wearable devices, in a close vicinity, is an indispensable key components to measure the body parameters, in BAN monitoring scenarios [293]; such as in healthcare to provide services, such as remote clinical care, diagnostics, monitoring, individual security, enhanced sports and fitness training applications [96]. IEEE 802 has established a task group called IEEE 802.15.6 for the standardization of WBAN, with the goals to define communication standards to optimize the energy power of in-body/on-body wireless nodes [226].

2.7.3 Challenges in PAN and BAN

Low power wearable sensor devices operate in environments where monitoring operations are required for extended period of time and frequent battery replacements are not possible. Research shows that the technologies, such as Bluetooth and Wi-Fi, have failed to support this, because they offer one or two weeks runtime battery [265, 68]. The solution is the proper usage of the wireless communication channels [68]. This fundamental constraint has triggered many researchers to develop different Media Access Control (MAC)[31] protocols with the objective of bandwidth maintenance and **low energy consumption** [389, 305, 390, 231]. Another issue is of output transmission power of the sensors, which must be minimum to support the communication and network range coverage. Ren *et al.* [313] shows the adverse biological effects (e.g., thermal and athermal) causes due to the wireless radio frequency radiation, especially for long term operations. Elias *et al.* [145] studies for the optimal WSN designs for data routing and relay positioning problems to increase the network life time.

Design of a sensor device is very important, and miniaturization along the functionalities has always had a priority. Beside this the portability of the devices and allowing other technology integration are considered important, such as RFID and Near Field Communication (NFC) as proposed in [228]. Reliable communication and real-time WSN's availability are major issues, especially during the critical life vital data acquisition scenarios. Specifically in context to the PAN, the self-organization between the nodes sometime become a major issue, such studies are presented in [163]. Beside this security is also a major concern during such monitoring scenarios, as in [61].

2.7.4 Energy Efficient WSN Technologies and Protocols

Having energy efficiency goals the short range wireless communication technologies are: Bluetooth [396], Bluetooth Low Energy (Bluetooth LE), ZigBee [250], ANT+ (see 2.8) [358, 221], RFID [382] and Near Field Communication (NFC) [228, 119]. Table 2.2 makes a comparison between these. The main factors that differentiate these technologies are: transmission power measured in power ratio in decibels (dBm); data rate measured in kilo bytes per seconds (kbps), the range of communications measured in meters (m), power consumption measures in Watts (W), and weight measured in kilogram (kg).

Bluetooth

The classic Bluetooth technology, also known as IEEE 802.15.1 standard, operates in the 2.4 GHz ISM band, with a bandwidth range from 1 to 24 Mbps depending on the Bluetooth version [282]. Its first basic flavour is Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR), which was adopted as version 2.0/2.1 [74].

Table 2.1: Bluetooth power classes with different range [71]

Class	Maximum Permitted Power (dBm)	Range (m)
1	20	100
2	4	30
3	0	1
4	-3	0.5

It's communication range is power-class-dependent, and varied from approximately 0.5/1m/10m/100m with respect to class 4/3/2/1 respectively [71], as depicted in the table 2.1. Bluetooth technology also offers different standard device profiles [73], such as Health Device Profile (HDP), Health Thermometer profile (HTP) and Heart Rate Profile (HRP), etc.

Ultra Low Power Bluetooth (BLE) or Bluetooth *Smart*

The second advanced flavour of Bluetooth technology is Bluetooth with low energy (LE), which was adopted as version 4.0/4.1/4.2 [74]; and feature a bandwidth of 1 Mbps with a range of 15 to 30 meter (m) [282]. It is good for long range radio connection, which makes it good for the IoT applications, that require long battery sustenance but may not need continuous monitoring [74].

Bluetooth Special Interest (BSI) Group [72] is with the goals to support the development of novel applications in the domains of healthcare, fitness, security,

and home entertainment industries [74]. Continua Health Alliance [3] is an international open industry group, non-profit interests of nearly 240 healthcare providers, works with the BSI Group to promote Bluetooth *Smart* devices in healthcare applications and have worked for the standardization of the profile specifications, such as for the healthcare domain: Blood Pressure Profile (BPP) for blood pressure measurement; Health Thermometer Profile (HTP) for medical temperature measurement devices; Glucose Profile (GLP) for blood glucose monitors; and Continuous Glucose Monitor Profile (CGMP) [175]. Additionally, for the sports and fitness categroy the profiles are: Body Composition Service (BCS); Cycling Speed and Cadence Profile (CSCP) to measure cadence and wheel speed during cycling by attaching sensors to the bicycle; Cycling Power Profile (CPP); Heart Rate Profile (HRP) to monitor heart rate; Location and Navigation Profile (LNP); Running Speed and Cadence Profile (RSCP) and Weight Scale Profile (WSP).

ZigBee

ZigBee is a Low Rate WPAN wireless technology, based upon IEEE 802.15.4 standard, that operates in the 784 MHz in China, 868 MHz in Europe, 915 MHz in USA, and 2.4 GHz worldwide frequency bands [60]. The other protocols that run on the same 802.15.4-based standard, including IEEE 802.15.5, 6LoWPAN, WirelessHART, and ISA100.11a. This protocol was developed to deal low energy consuming applications, with additional features of low cost, low data rate and long battery life [169]. Its data rate varies between 20 Kbps (868 MHz band) to 250 Kbps (2.4 GHz band). The Zigbee devices mostly spend their time in sleeping mode, especially in scenarios where their participations are limited, hence permit the batteries to prevail for several years without replacement [180]. If Zigbee network is deployed within an approximation range of 100 meter (m), the device can survive for years on a small coin battery, for monitoring applications [169]. It provides self-organized, multi-hop, and reliable mesh networking with long battery lifetime [233, 234]. Zigbee alliance [341] has developed a number of standards, each of which targets a specific application area, such as home automation (HA) [44], health and fitness [45], smart energy [46], and smart building [43].

RFID

While introducing about the RFID technology Roy Want [382], the IEEE Pervasive Computing editor in chief, says that it uses radio-frequency electromagnetic fields for the automatic identification of things holding RFID tags when they come close to an RFID reader. Unlike bar-code reading technology, it identifies the objects at a distance and don't require a line of sight. RFID tags posses more data

[3] www.continuaalliance.org

than the bar-code tags. The two classes of this technology are active and passive, where the former require a power source, such as an attached battery; and in the latter case the RFID tags don't require the battery support. Passive tags have an indefinite life and are easy to fit into any device; and they consist upon three parts: an antenna, a semiconductor chip attached to the antenna, and some form of encapsulation. Through various modulation techniques, an RFID device may transmit or receive near- and far-field-based signals.

Generally, more frequency is required to transmit at a longer range, but this is more expensive. Therefore, low-frequency RFID systems (30 KHz to 500 KHz) have short transmission ranges (generally less than six feet). High-frequency RFID systems (850 MHz to 950 MHz and 2.4 GHz to 2.5 GHz) offer longer transmission ranges (more than 90 feet). Due to the miniaturization process the RFIDs are easy to incorporate in other items. For example, the entomologist scientists have successfully glued RFID micro-transponders to live ants [278].

Near Field Communication (NFC)

Another wireless technology that has won popularity in recent years is Near Field Communication (NFC), based on Radio Frequency Identification (RFID), operates for peer-to-peer data transmission within a short-range of up to approximately 10 cm (3.9 inches) [175]. It works between a NFC tags and smart objects (such as mobile) by bringing them closer and employing electromagnetic inductive coupling for identification. Passive NFC tags can be completely unpowered, and will become active only when a NFC field is present, therefore NFC and RDF are significantly different to the other low power wireless technologies discussed previously and both operate at the 13.56 MHz frequency[119].

2.8 ANT+ Protocol

ANT is a proprietary protocol for applications where monitoring operations are desired for a long time with low battery consumption. It divides its bandwidth into 125 channels of width equal to 1 MHz and operates on 2.4 GHz spectrum with transmission duration less or equal to 150 microsecond/frame for 8 bytes of data. It uses special network keys to distinguish different networks [358]. ANT provides management of physical, data link, network and transport layers of OSI stack, see figure 1 [221]. However ANT+, an extension of ANT, manages session, presentation and application layers to provide data and devices interoperability. The format of information that ANT represents at the higher layers, if agreed upon by vendors and is defined mutually in a set, is actually ANT+. These standards are called device profiles and are typically tied to a specific use

Table 2.2: Protocol Comparison: Bluetooth, BLE, ZigBee and ANT+

Standard	Bluetooth	BLE	ZigBee	ANT+
IEEE Specification	802.15.1	BLE	802.15.4	ANT/ANT+
Topologies	P2P, scatternet	P2P, star	ad-hoc, P2P, star or mesh	broadcast, P2P, star, tree, mesh [140]
Frequency band	2.4 GHz	2.4 GHz	868/915 MHz, 2.4 GHz	2.4 GHz
Max signal rate	1-3 Mbps [35]	1 Mbps [35]	20, 40 and 250 Kbps for 868 MHz, 915 MHz and 2.4 GHz resp. [35]	1 Mbps, broadcast (200 Hz), Burst (20 Kbps), Adv. Burst (60 Kbps)
Nominal range	10 m	280 m [221]	30-100 m	30 m
No. of RF channels	79 [235]	40 [221]	1/10; 16 [235]	8 [221, 140]
Node Type	master, slave and standby nodes [235]	single mode (BLE) or dual mode (Bluetooth and BLE)	Full Functional Device (FFD) or Reduced Functional Device (RFD) [221]	2 node types: central (to collect data) and simple (transmit the detected information to central)
Battery Life	? rechargeable	1 year	6 months, non-rechargeable	up to 3 years [221]
Power Efficiency (power per bit)	-	0.153 uW/bit [221]	185.9 uW/bit [221]	0.71 uW/bit [221]

case, for example heart rate, blood pressure. Once both the devices agree upon a unique standard, then those start understanding each other's communication messages, such a thing has been presented in the figure 2.6 on the right side, showing open interoperability for both side understandability. So a device must be able to interpret these profiles, which are definitions of possible applications and specify general behaviours that ANT-enabled devices use to communicate with other ANT+ devices. ANT+ profiles allow both sides to understand each other, as both have already agreed upon a specific format [358]. A detailed discussion about the ANT+ protocol is provided in the Appendix 11.4.2.

ANT supports the establishment of numerous public or private managed networks. Each ANT network possess a unique identification. For two ANT devices to communicate they must be in the same network. There exists an ANT channel between two devices which must possess same frequency, message period, device type and transmission type (i.e. slave or master). Each node representing a sensor device, consists of an ANT protocol engine and a host Micro Controller Unit

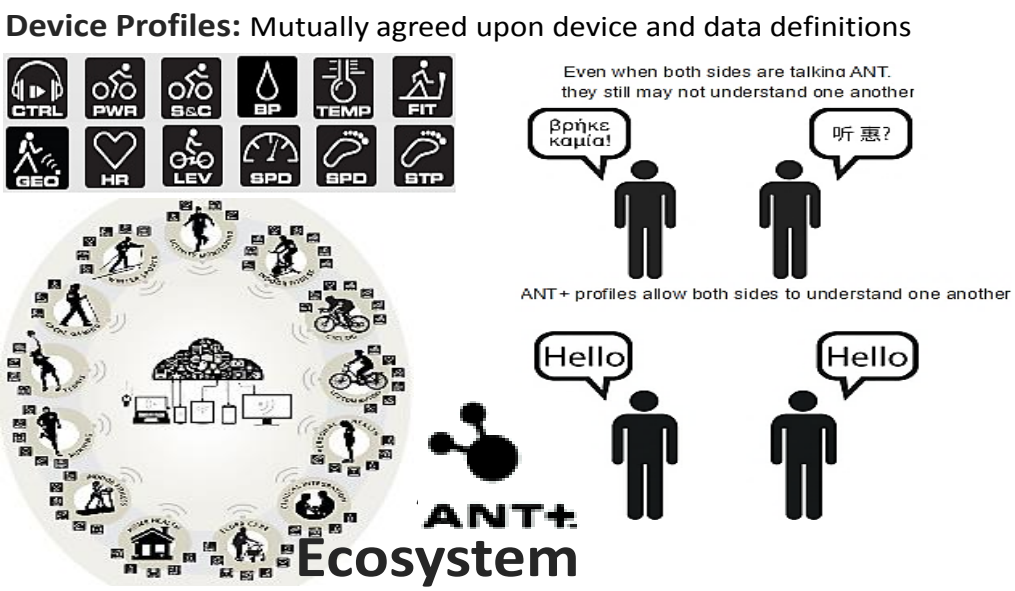


Figure 2.6: Open interoperability of ANT+ to build vast ecosystems [358]

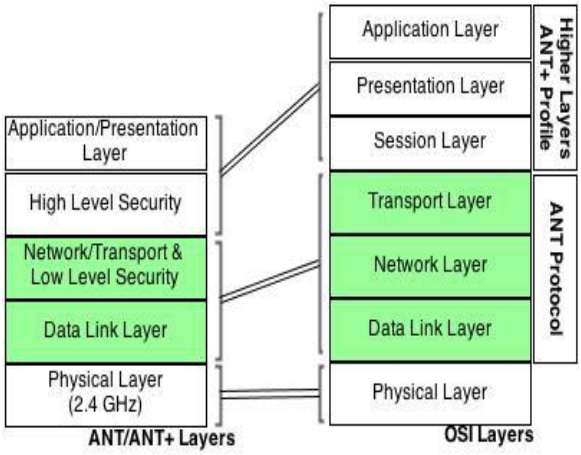


Figure 2.7: ANT model, OSI layers and ANT+ profiles correspondence [221]

(MCU). It has both master and slave nodes which can participate in one or more networks [140]. There are three different ways for channel usage, i.e. independent, shared and scan channels [140].

ANT sends each of its 8 bytes data packet payload over Radio Frequency (RF) channel using four different data types exclusively, such as broadcast, acknowledge, burst and advance burst [221, 140]. Beside the other basic data message payload (i.e., 8 bytes), ANT+ also permits additional data to the host receiver.

Such an extended format consists upon 12-bytes; with more augmented information, such as device identification, device type and trans type [140]. ANT devices transmit device related information mostly after every 65 sensor data pages [140]. It supports more than 60 different types and almost same number of generated events. Establishing a channel is an act of pairing two devices and agrees upon same Channel ID, frequency and message period (See Appendix .1.2). This forms different relationships as: permanent, semi-permanent or transitory [140, 361].

2.8.1 Flexible and Interoperable Data Transfer (FIT) Protocol

Embedded devices are very limited with the resources, and they cannot communicate directly with the Internet as it require a lot of energy and long range communication protocols. Beside this data collecting SOs, such as display watches, are also possess limited storage resources and cannot store big amounts of data sets. Such embedded devices, after accepting data from the ANT+ sensors, store the data in a compact and compressed format. The Flexible and Interoperable Data Transfer (FIT) protocol is designed specifically for the local compact data storage and interoperable data sharing. It is defined for extensibility, interoperability and compactness of the information stored as a set of FIT message templates, such as user profiles, heart rate profiles and activity data in files. The detailed process of FIT encoding and decoding is provided in the section 3.2 of FIT File Protocol document [364]. Figure 2.8 depicts the scenario for the data flow between devices using FIT protocol, based upon the following steps.

1. ANT+ devices measure parameters, such as heart rate, foot steps
2. Real-time data is broadcasted using ANT+ protocol
3. Session events and real-time physiological or activity data is saved as FIT files on a data collecting device, such as a display device etc.
4. The FIT file is transferred to the PC or a mobile using ANT-FS

The objects of FIT protocol are: (i) data interoperability across devices; (ii) devices integration; (iii) forward compatibility for protocol growth; and (iv) auto platform compatibility e.g. endianness etc. An FIT format consists upon the following components, such as a global list of FIT messages and FIT, fields together with their defined data types. An FIT file contains a series of records that, further contain a set of headers and the contents. The record content can be of two types: (i) definition message, that is used to specify upcoming data, and (ii) data message, the data format of each messages field; and methods of data compression. Such FIT files can be downloadable at a later time and any FIT-compliant device

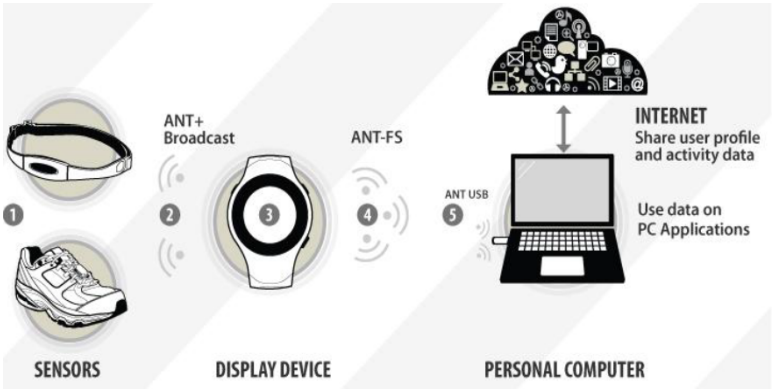


Figure 2.8: Data flow between devices using FIT protocol [364]

can interpret them [364]. The two type of FIT profiles are: (i) Global profile, contains an outline of all messages in the global FIT profiles and (ii) Product profile, contians an outline or subset of product-specific messages. Examples of FIT file [362] are: Device, Settings, Activity, Weight, Blood Pressure etc.

FIT based Development

ANT Alliance provides a Software Development Kit (SDK)to generate code and libraries specific to a product's requirements [363]. The SDK enables efficient use of a binary format at the embedded level, to significantly reduce the programming effort by targeting the programming language and the platform, thus allow the rapid product development. It is used to configure target products and generate the necessary FIT code and libraries [364].

2.8.2 ANT File Share (ANT-FS)

ANT File Share (ANT-FS) is an extension of the ANT protocol that provides a robust framework for transferring FIT files wirelessly between two ANT enabled nodes using less network communication resources. It is often used in ultra low power display devices with storage capabilities, such as fitness watches. For security each ANT-FS network possess a unique key [360].

2.8.3 A simple ANT+ Network

A simple four-node network, depicted in figure 2.9, describes an application where information from multiple nodes (B, C and D) is received and possibly analysed, by a single central node (A). Nodes B, C and D shall establish only one channel each, however node A have three channels. So A needs a device that may allow

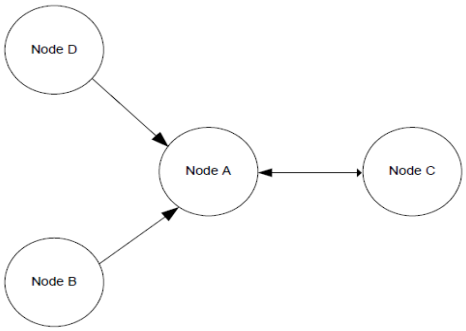


Figure 2.9: A Simple ANT+ Wireless Sensor Network

more channel configurations. Suppose node C communicates with Acknowledge (ACK) data type while the other nodes communicate with broadcast data type. The two ways to implement this network, are: Implementation using Independent channels or Implementation using Shared channels. More in depth technical details about the ANT+ network configuration and implementation can be found in the Appendix 11.4.2 in section .1.1 and .1.2.

2.8.4 ANT+ Development

For the developers there is an online guidance [139] from the ANT alliance; which provides the ANT Windows Library Package (AWLP) [137]. This contains a set of libraries to support ANT enabled application development for the Windows PC. The AWLP also contain application demos, depicting how to usage different ANT libraries during the development process. To develop computer applications for the ANT+ technology the controller component must be ANT+ enabled. This is possible by connecting an ANT chip or module which receives ANT messages from a specific ANT+ sensor, and decode this data according to its ANT+ device profile using the AWLP [137] which contains a set of libraries to support ANT enabled application development.

Make Enable Sensor Communication

Pre-installed ANT+ enabled devices are available in the market. However, for the other components which do not support ANT+ protocol, we can extend them using ANT/ANT+ adapters; so to make them enabled to communicate. Thus, for the wireless data transmission between the device and the PC as a gateway, we can use ANT USB Adapters. There USB adapters are in two flavours - i.e., ANT1 (USB1) and ANT2 (USB2). ANT1 has a limited number of channels (i.e., 4), whereas the newer ANT2 stick has more channels (i.e., 8). It's communication

aspects are addressed under IEEE 802.15.4 [221]. Different devices operate upon different channel types, message periods and radio frequencies (RF); and each of them needs to be configured accordingly [140]. The default RF frequency value is 66 and represents the network operating frequency of 2466 MHz. The channel message rate can range from 0.5 Hz to above 200 Hz. For this, we need to configure channel and USB settings with appropriate parameters to build up a network. The AWLP also contain the functionality to connect to ANT enabled USB sticks given the network settings. AWLP also manages the low level serial communication at the PHY layer. It also allows a programmer to interpret the ANT message frames, and optionally do logging of the raw messages during different communication sessions for the debug purpose.

For the development, it requires around 2 ANT enabled USB sticks [138] or an ANT development kit [139]. We need to install the USB stick drivers to make enable the serial communication with the ANT+ sensors [136]. It is to make sure that a proper driver must be installed with respect to the USB stick some one is using. The two types of USB sticks are: USB1 and USB2. Such information is normally printed upon the USB stick dongle itself. If someone is using the USB Interface Board (UIF) and an ANT module, one may need the USB1 drivers to install. The driver installation instructions are provided in the *ANT Development Kit User Manual* [137]. It is to note that, only one application can access an ANT USB stick at a time.

If someone is using Mac OS he/she may not have to install any drivers for the ANT USB sticks, rather the installation of the ANT MacOSX Library Package with Source Code would be enough [134]. For secure communication and authentication of the devices the library uses the network keys [135]. The network key consists upon 8-byte code that uniquely identifies a network. It is used as an access control mechanism for the security. Only channels with identical valid network keys may communicate with each other.

2.9 Summary

The start of smart home technology is rooted in the start of home automation in 1960. This is an application of pervasive computing to provide personalized services to the users using Ambient Intelligence. They have applications in the area of comfort, healthcare, energy efficiency and security. The architectures of Hospital Information Systems require middlewares to build centralized or decentralized smart home environments. The chapter discusses different projects in the area of healthcare.

Part II

Employed Technologies

Chapter 3

Middleware

The world is concurrent. Things in the world dont share data.
Things communicate with messages. Things fail.
-Joe Armstrong, father of Erlang

Some introduction is already provided in previous chapter sections (see 1.4.1). Middleware plays a role of a glueing mediator by staying between the OS, application program and the network to integrate heterogeneous resources. They offer different communication types, and high-level general services, such as message based, object communication based, continuous data processing or event based. We shall learn that, there are proprietary and open middlewares; with their applications in different domains.

3.1 Introduction

There are many different definitions for a middleware, such as: It is a software glue that helps integrate the heterogeneous components, or it is the slash in client/server, or it is a software glue between software and the network, or it is a layer between OS and distributed applications. Alonso *et. al.* [181] and Bernstein [66] added that they are the direct result of working with the infrastructure for application integration to have easy to build enterprise information systems. Middlewares provide an abstraction to the common programming types and an infrastructure to the distributed applications.

Bernstein [66], says that, a *middleware service* is a general-purpose service that sits between platforms and applications, as in figure 3.1. He adds that a platform is a set of low-level services and processing capabilities of a processor architecture and an OS's APIs, such as Intel x86, Win-32, Sun OS, IBM RS/6000 and Windows NT etc. Whereas a service is defined by the APIs and protocols it

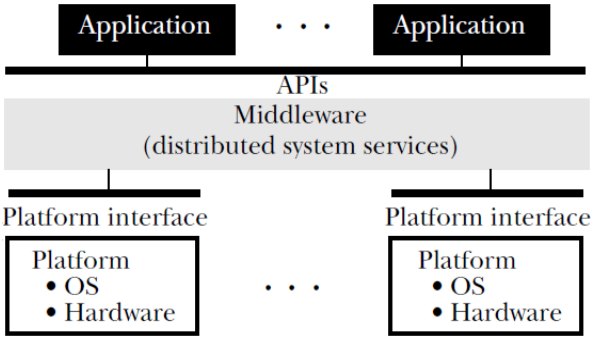


Figure 3.1: Middleware Service [66]

supports. However there may be different multiple implementations that conform to a service interface and the protocol specifications. In a technical precise way it is hard to define a middleware, like high-level system concepts. There are certain properties that if consider together, we can perceive that the component is not an application. Such components are generic across applications, platform independent, distributed and support different interface and protocol standards [66].

3.2 Middleware Communication Service Properties

To understand, lets consider a middleware as an additional service in a client-server scenario. For example, an Electronic Email System (EES), where a client composes an email and delivers it to the EES and expects that the message will be submitted to the recipient eventually. Similarly, the recipient's email agent connects to EES to see either any mail has arrived? If so, a mail is transferred to the recipient's mail agent so that it can be read by the recipient. This scenario depicts certain cases, such as: (i) either the mail was stored by EES locally?, (ii) does the sender continue its work immediately after submission to EES?

3.2.1 Persistent or Transient

In persistent stage, a message that has been submitted for transmission is stored by the communication middleware as long as it takes to deliver it to the receiver, for example, an EES. Whereas in the transient mode, a message is stored by the middleware as long as the sending and receiving application are executing [354].

3.2.2 Asynchronous or Synchronous

If a sender continue its work immediately after it has submitted the message, then this is in asynchronous mode. Whereas in synchronous mode a sender waits until a reply is received either from the middleware or from the receiver about its submission delivery status.

3.2.3 Discrete or Streaming

The distinction between discrete and streaming communication is highly important. All the examples some how fall in the discrete communication category, where both (or more) parties communicate by messages, and each message is a complete information unit. Whereas, a streaming mode involves multiple messages delivery, one after the other with in a certain sequence, having a timestamp.

This and the next sections are based on A. Tanenbaum's book *Distributed systems: principles and paradigms* [354]. There are several combination of persistence and synchronization, and the famous one is persistence-to-synchronization at request submission. This is a common for many Message Queue Systems (MQS), which we shall discuss in the next section 3.3, beside other high level middleware communication services, i.e. Remote Procedure Calls (RPCs), Stream-Oriented Communication, and Event-Based Architecture (EBA) middlewares.

3.3 High-Level Middleware Communication Services

Already discussed that, middleware communication protocols support high-level communication services, following we discuss them step by step:

3.3.1 Remote Procedure Call (RPC)

An RPC permits a process to call a procedure or an object on a remote machine as being local in a transparent manner, using a request-response paradigm. In 1984, Birrell and Nelson allowed different programs to call each other's procedures located on different computer machines, and they permit the two remote processes to communities as if they are in the same address space. The precised communication scenario is as: Suppose, a process on machine A calls a procedure on machine B. The calling process suspends, and the called produce executes on B. The information from the caller also transferred to the callee and the effect appears in the callee procedure's results. In this way the programmer is not aware of the argument passing.When a client packs its parameters it is called as marshalling; and when a remote procedure's stub converts the results back according to the

callee's address space it is called de-marshalling, as depicted in Figure 3.2. There

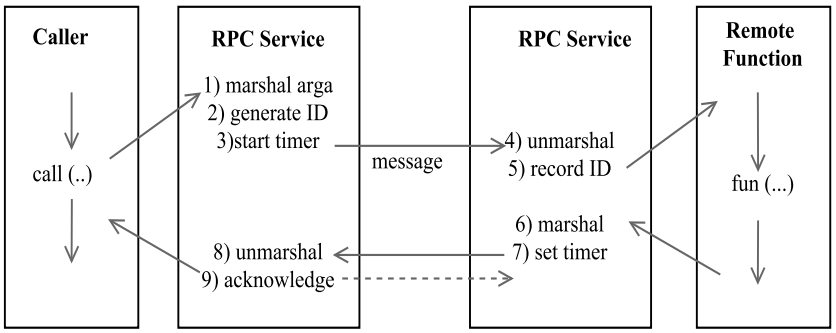


Figure 3.2: RPC request-response steps [188]

are some issues also like, the execution happens in different address spaces and each machine often has its own representation and data formats, e.g. endian notations, 1's or 2's complements, EBCDIC and ASCII codes. For the pointers and references this becomes even more complex, especially when they are pointing to an array. A solution is that the *client stub* must copy the whole array and send it to the remote procedure. For a transparent RPC the caller and the callee must agree to the same format or protocol. They should follow some specific steps, especially in case of sending a complex data structure.

3.3.2 Object-Oriented Middleware (OOM)

RPCs are not object oriented, or can not call methods of an instance of an object because they are not able to transfer or serialize an object directly to a remote method. Remote Method Invocation (RMI) is based upon object-oriented programming to enable communication between distributed objects, e.g. CORBA, Java RMI. Java RMI's allows to invoke methods of an object which exists in a different address space, or in another Java Virtual Machine (JVM). RMIs are basically object oriented RPCs, which uses object serialization for the transmission of the objects. This object serialization is the same concept which we have learned as marshalling in RPCs. Similarly it performs un-marshalling upon receiving results back as reply from the remote object. As the figure 3.3 [387] shows, the three layers which RMI systems have are: (i) stub/skeletin, client-side stubs (proxies) and server-side skeletons (dispatchers); (ii) remote reference layer, it invokes the mode of behaviour with semantics (e.g., unicast, multicast); and (iii) transport, responsible for managing and setting up the connection.

A client side RMI to a remote object's method first travels down, from the stub to the client-side transport, through the layers of RMI system, then up , from the remote-side transport to the remote side skeleton, to the server.

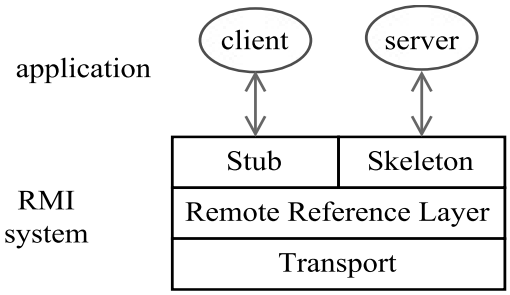


Figure 3.3: RMI System Architecture [387]

3.3.3 Message Oriented Communication Middleware

RPCs and RMIs bind the client and server to each other that both have to be active at the same time. If an RPC blocks for some time? then the client may have to wait. How to avoid binding of client and server, that the client could start working after submitting a call. The solution to this is *messaging*, that assume that both the parties could execute during the time of communication. This is possible by storing the messages in message queues.

Message Oriented Transient Communication

Mostly distributed applications are built on top of the message-oriented model offered by the transport layer, and for it, we first discuss messaging through transport-level sockets. For example, Berkeley sockets is an API for Internet sockets (across the network communication) and Unix domain sockets (same host based communication), used for Inter-Process Communication (IPC).

Interfaces we had for RPCs were not standardized and many of the software were not compatible. Therefore special attention was paid to have a standard interface for the transport layer; to permits the developers to write their software that could use the entire suite of protocols. The standardized interfaces will make it easier to port their application to different machines. A *socket* is a standard communication end point where an application can send and receive data. Message Passing Interfaces (MPI) [177] were introduced because sockets are not suitable to the high-speed networks used in server clusters. These standard API that could be useful in parallel applications, and can work in transient communication.

Message Oriented Persistent Communication

It is also known as Message Queues or Message-Oriented Middleware (MOM), which provide vast support for persistent asynchronous communication. Such a middleware holds an intermediate capability to store the messages, until their

delivery, hence do not require the sender/receiver to be active during message transmissions. Berkeley sockets and MPI are for very fast communication that matters for seconds and milliseconds, whereas MOM are typically targeted to support message transfers that are allowed to take minutes.

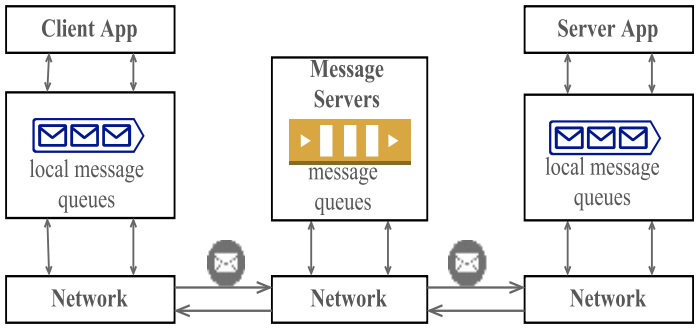


Figure 3.4: Message Oriented Middleware

It is important to understand the organization of the collection of queues distributed across multiple machines, and Figure 3.4 depicts the general architecture of it, by distinguishing the client and server components from the main middleware holding the large message queues with persistent storage. Beside it also depicts the network level addressing. Such an architecture is ideal for database integration for the enterprises. IBM WebSphere Message Queue (MQ) [191], Java Message Service (JMS) [182, 283] are the examples of it.

3.3.4 Stream-Oriented Middleware

The middlewares we have studied so far were based upon complete units of information, and without considering the time factor.There are many cases in which time is a crucial factor, such as audio video streaming, sensor data streaming, health measurements etc. What facilities does a distributed system must offer, to make enable the streaming data communication; such as described in [344]. To communicate time-dependent or sequential data a special media is required that must support continuous flow of data. Another important factor is the representation or format of the data. Different data types require different formats, e.g. GIF or PNG for images; AVI or PCM for audio; etc. The data items themselves also possess temporal relationships, e.g. video required a correct order of images. Same is the case with life data sensor measurements, e.g. heart rate.

Distributed systems provide data stream processing techniques, which allow the passage of information in a sequence. Such streams are both applicable to continuous as well as to discrete media. We shall learn the details in chapter 6.

3.3.5 Event-Based Middlewares (EBM)

An event can be defined as "a significant change in state" [261]. An EDA is mostly consists upon event emitters (or agents), event consumers (or sinks), and event channels. A basic architecture style for distributed applications is based upon generating events upon several factors, such as realising the state of the system, publishing data, or notifications to the subscribers. Such EDA support the development of pub-sub systems (see 1.4.1), where subscribers express their interests by subscribing to specific events, and publishers publish the data messages upon generating events. EBM provide event services to the publishers and the subscribers; and play the role of a broker, as depicted in the figure 3.5. Hence the difference to message queues is that, rather than distributing the messages to the queues, the event-broker (or message broker) route it to the subscriber.

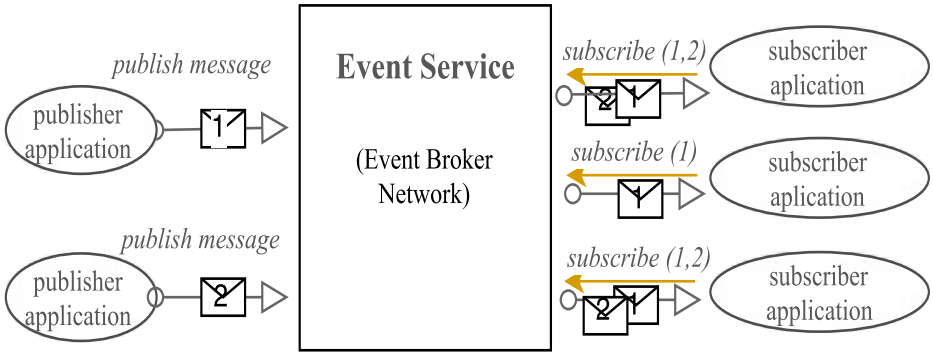


Figure 3.5: Event-Driven Middleware

3.4 Middleware Standards and SHE

Middlewares can be proprietary or open, as well as, domain specific or general purpose [58]. Many software companies prefer proprietary middleware; but open standards have more tendency; and we discuss some of them as: (i) *Open Services Gateway initiative*[1] – *OSGi :* This is an open standard and a service-oriented dynamic software component model for Java programming language, which does not exist in standalone JVM environment. This allows to install, un-install, start, stop and update the components or applications remotely without a reboot.; (ii) *Foundation for Intelligent Physical Agents*[2] – *FIPA:* This is an agent-based component model to provide dynamic support for agent management.; and (iii) *Web*

[1] http://www.osgi.org/Specifications/HomePage
[2] http://www.fipa.org/

Standards[3] – *WS:* These are for the Web applications development and world-wide interface standards, such as Web Services[4], REpresentational State Transfer – REST architectural style, or standards for the Web of Things (WoT).

In SHE the two responsibilities of middleware are: (i) interfacing with device drivers and (ii) to provide interoperability using standardized interfaces and protocols [58]. OSGi based general purpose platforms are: (i) Apache Felix[5], and (ii) Eclipse Equinox[6] ; similarly FIPA based are: (i) Java Agent DEvelopment Framework[7] (JADE) – for the development of intelligent Java agents, and (ii) Java Intelligent Agent Componentware[8] (JIAC). Examples of proprietary are [58]: (i) Demotic OSGi Gateway (DOG) [77], and (ii) LinkSmart, from EU HYDRA project [144].

3.5 Summary

Middlewares are important for the application or system's integration to resolve heterogeneity. They offer general purposes services that sit between platforms and applications, e.g. RPC, RMI, CORBA. During their processing operations the data is differently formatted; persistent or transient; and, discrete or streaming. They allow synchronous or asynchronous modes. Middlewares communicate by either messages (i.e., MOM), object methods (i.e., OOM) or events (i.e., EDA). Middlewares are proprietary and open with some standards. There are special domain oriented middlewares, such as OSGi, JADE etc.

[3] http://www.w3.org
[4] http://www.w3.org/standards/webofservices/
[5] http://felix.apache.org/
[6] http://www.eclipse.org/equinox
[7] http://jade.tilab.com/
[8] http://www.jiac.de/

Chapter 4

Windows Services

Everybody can be great because everybody can serve.
–Martin Luther King, Jr.

The new operating systems contain many sensor and environmental monitoring services and any user can use them freely, after configuring with the smart devices. This chapter gives an overview about the Windows Services; their categories, types, advantages, disadvantages and how to built them.

4.1 What is a Windows Service

A Windows service is a program that operates in the background and is managed by the *Service Control Manager* [23] on Windows NT based operating systems. This is similar in concept to a Unix daemon program [24]. Microsoft Windows services enable the programmers to develop long-running executable applications that run in their own Windows sessions [1]. Since these services operate within their own sessions and they do not interfere with other processes, therefore they are ideal for the long term processes, such as monitoring scenarios in the domain of healthcare. Such services mostly do not provide user interfaces, except than the operational capability like the Windows' built in "start" and "stop" functionality, etc.; or a few settings to control security, etc.

Such Windows services posses capabilities to be launched automatically upon Windows' bootup process; or on demand by another service due to service dependencies. A user can also run such a service within the security context of a specific user's account [1]. These features make them great for standalone long-running processes that "just work" and need not user's direct intervention, but these features are also very hard to debug and develop.

4.2 Pros and Cons for Windows Service

The advantages of using services over a Windows GUI application are as:

1. They are auto start without any user login.

2. Security - they can run under a specific user name.

3. Service Recovery Options, i.e., a crashed service can restart automatically.

4. Such services allow remote service control i.e., a remote user can start or stop a Windows service from a remote computer without logging on and clicking on any GUI interface. For example using Net Start [27]. For the form application a person have to login into the computer to start or stop a service, for many computers it is hard to do so.

5. Services may generate events to start other services.

6. Since services run in a separate Windows session (mostly, session 0) and, therefore can be shared between all users of a computer.

Following are the disadvantages of using Windows services:

1. Difficult to develop

2. Hard to debug, since they are attached to a running process.

3. We do not have a direct user interface. To have this one have to develop a separate interface for this purpose, beside the service component.

4.3 Types of Windows Service

Microsoft Operating System have different System service categories, such as Application Programming Interfaces (APIs), Services, Component Object Model (COM), Dynamic-Link Libraries (DLLs), Remote Desktop Services etc. [26].

Using Visual Studio .NET Framework, a programmer can create two types of services, i.e. (i) Win32OwnProcess, are the services that are the only service in a process, and (ii) Win32ShareProcess, are the services that share a process with another service. One can know the type by querying the *ServiceType* property.

4.4 Windows Service Development

A programmer can develop a Windows service using Microsoft Visual Studio[1] or Embarcadero Delphi [2]. For a program to be a Windows service, must handle the *start*, *stop*, and *pause* messages from the Service Control Manager. The Windows Resource Kit [30] for Windows NT 3.51 [3], Windows NT 4.0 [2] and Windows 2000 [5] provides tools to control the use and registration of services.

Services must be created using a .NET Framework-enabled project that creates an ".exe" file, which is built and inherited from the *ServiceBase* class. The Projects containing Windows services should have the installation components for the project itself and the services it contain [1].

4.5 Data Acquisition Windows Services

Data is acquired using various methods, such as from papers, distributed datastores, interactive voice response or Electronic Data Capture systems (EDC) etc. An EDC system collects data in electronic format to increase data accuracy and to minimize the time [62]. Microsoft windows 7, 8 and 10 come with many sensor data acquisition services[4]. Windows 7 includes native support for sensors and provides device APIs [22]. Windows 8 comes with pre-installed sensor monitoring service [3]. Windows 10 comes with many pre-installed monitoring services, such as *Sensor Data Service* [4].

4.6 Summary

Windows Services are very efficient for the long term monitoring processes. They do not require a user's login to get start, rather they are auto-startable. New operating systems are coming with pre-installed monitoring services. A manager can manage the services from a remote server without loging into the system.

[1] www.visualstudio.com
[2] www.embarcadero.com/products/delphi
[3] https://support.microsoft.com/en-us/kb/124814
[4] http://servicedefaults.com

Chapter 5

Model-View-Controller (MVC)

It is better to have a design pattern than to be fascinating.

For the usability applications need refined interfaces, which are possible by decoupling the logic, data and control. The Model-View-Controller is a design pattern, which is famous for developing applications to decompose system into subsystems to have flexibility.

5.1 Introduction

To provide effective actions and control of the software components to the human being, a system always needs special interactive interfaces. The study of interactivity lies in the domain of Human-Computer-Interaction (HCI) [216], for which the usage of design patterns and pattern languages is proved in [368], which are mostly for desktop interfaces. The Model-View-Controller (MVC) is a design pattern [86, 6], which is useful to architect interactive software systems [236]. These patterns provide flexibility to the programmer to decompose the system into subsystems that are relatively independent to each other, as a result a program designer can partition the user interface and the system's functionality.

The figure 5.1 depicts the MVC components and their interaction - i.e, the Model, the View and the Controller. The model is the central component, which holds the application logic and data. This is independent of the view. The view displays the information to the user, and it is possible to have more views. The Controller processes the user inputs and have the user's interface. This converts the inputs into commands for the view or model. Presentation-Abstraction-Controller (PAC) is also a design pattern [115] , which decouples the application's information from the user's interface [236]. PAC's presentation component is a composition of view and controller of MVC; The application's data becomes the

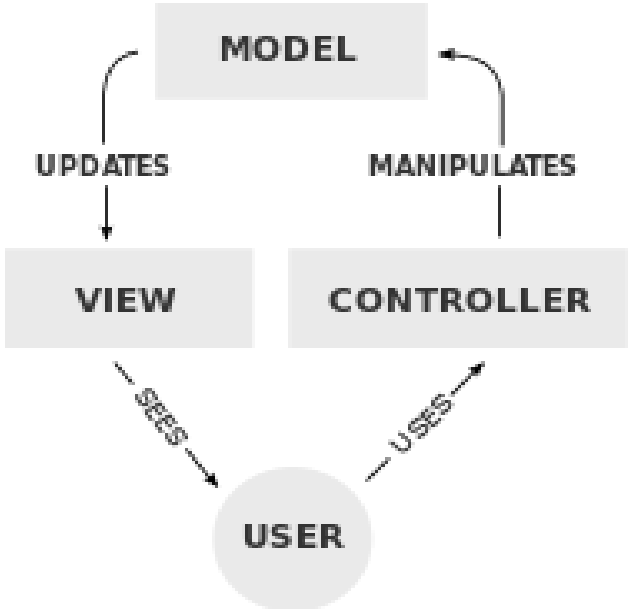


Figure 5.1: Model-View-Controller

abstraction component; and the control component handles the message cooperation between the decoupled presentation and abstraction components.

5.1.1 Web Application using MVC design pattern

MVC is applicable across all programming languages to decouple the modules very finely. However in the Web context MVC design pattern works very well; and many modern MVC-based frameworks are very popular, such as Ruby-on-Rails[1], Django[2] and ASP.NET MVC [6]. They can support on the fly-adaptive interface designs. For the MVC based real-time Web application development there exist frameworks, such as Node 6.3.6, Express 6.3.6.

5.1.2 Windows Service and MVC

The MVC design pattern patterns are not only for the Web, rather are applicable in any programming context, like the desktop programming. But clear separation between a controller and view is hard. There exist different variants of MVC; like Model-View-ViewModel as used in .Net WPF applications. While working with the monitoring applications interactivity is not required directly during data

[1] rubyonrails.org
[2] www.djangoproject.com

acquisition, like Windows Services which do not have direct views, but this is possible to have views.

5.1.3 MVC Issues

MVC based design patterns are having some independency assumption for the study of usability, which actually appears during the real interactions, which are mostly unlimited because of the many roles and many users and each one have different requirements and access control. MVC and PAC tries to approach the UI design as an independent subsystem, however this decoupling can be dangerous as the functionality is actually embedded into the application logic which might have effects upon the system usability [329]. Similarly during the design process, if the usability aspects are considered quite early, they might effect the system architecture [329].

5.2 Summary

MVC design patterns are famous for building flexible interactive applications to separate the user interfaces from the underlying application data. The model contains the core application functionality and its data. The Views display information to the user in different format, and the Controller handle user input.

Chapter 6

Real-Time Web and Data Streams

Reducing kilobytes of data to 2 bytes, and reducing latency from 150 ms to 50 ms is far more than marginal. In fact, these two factors alone are enough to make Web Sockets seriously interesting to Google.–Ian Hickson (HTML5 Spec Lead)

Now the computers are embedded in everyday life, due to IoT features, allowing an individual to interact with several distributed devices and systems simultaneously. This ubiquitous prevailing of embedded sytems has openned the doors for the real-time processing and management of the data and resources. Hence the old traditional batch processing modes are rarely used and are limited to some specific systems. This chapter is limited to the software issues only and shall discuss some of the technologies related to the real-time web applications; mainly in context to the Real-time Web and real-time data stream processing and management issues. HTML5 provides a new protocol for the Web,- i.e., WebSockets, which improves real time applications.

6.1 Introduction

The post PC-era has shown rapidly evolving new technologies, due to miniaturization goals, such as manifold reduction in the size and cost, hence observing an emergence growth in small and portable computers. In this post PC-era, the use of computer systems based upon real-time and embedded technologies has already touched everyday life and is growing at a rapid pace. Such real-time and embedded technologies previously was just restricted to specific applications in the domain of space, defence and environment monitoring [251]. According to an estimate, 70% of the processors production is for the real-time systems [323]. Such rapid technical growth has supported the real-time applications development.

As the micro-sensor technology is becoming cheap, the digital societies are adopting to manage their ordinary day to day physical equipments by getting them sensor-tagged. Such smart objects help them by transmitting real-time information about the environmental equipment after their identification. This kind of sensorization of the real world environments disclose the monitoring and control applications with bulky data processing with low latency requirements [345].

The M2M, IoT or Web-of-Things paradigms have opened the doors for the real-time embedded systems to get integrate with each other to build big global ecosystems. Therefore the real-time systems are now ubiquitous and have penetrated in a large number of application domains, such as market feeds, electronic trading in eBay or Wall Street, fraud detection in banks, security services, environment monitoring, e-health, digital medicine, smart homes and smart cities. The in-time process control and automation of the processes for the real-time scenarios, now have become indispensable part of the new generation pervasive systems.

The utilization of Internet as a platform to connect things has changed the information requirements of the society also, which has now shifted from just information usage i.e., Web 1.0 era; to information interaction i.e., Web 2.0 era; and to the recent information for pricing i.e., e-commerce systems [243]. Such global information exchange and electronic trade in volumes generate millions of messages per second and is expanding exponentially. On the other side the text based social networks, such as Twitter, YouTube, Facebook etc; generate meaningful volumes of information which require real-time data sharing. Alongside, in the healthcare domain a patient's health critical life data, obtained during the monitoring experiments, becomes more important and critical to be shared in the real-time with the stakeholders. Especially the extremely fast delivery of the events is mandatory for the critical situations, such as heart attack, fire alarm or a sudden fall.

As a result, one fundamental requirement of a distributed web based system is efficient real-time processing and large amount of information exchange over the Internet [118]. Traditional systems are no more sufficient for such applications which require high-volumes of data stream processing in real-time [345]. Different processing modes are discussed in section 1.4.1, and one of them is real-time processing. There are number of factors that may effect the information exchange mechanisms adversely. For example, the two or more parties taking part in the information exchange may possess different platforms, and could only understand the information in their own specific format [118]. Such situations require a data format which must be acceptable to both or all the parties. The limited internet resources, such as bandwidth, service availability etc., may also complicate the transmission process; especially when dealing with large amount of data; such as graphics, videos, voice streams or sensor data streams; or a number of clients simultaneously try to connect with the same server.

6.2 Real-Time

Real-time systems consider the execution of their functionality in some quantitative time, which can be measurable using a clock. In contrast to this, the qualitative concept of time is called as logical time or virtual time; which deals with the ordering of the events, like before, after, precedes, succeeds [251]. Therefore, a system is called as **real-time system**, when we need quantitative expression of time (i.e. real-time) to describe the behaviour of the system.

6.3 Real-Time Web

To understand real-time web we must deal it with general and technical perspectives, in the general terms the creation of huge volumes of information over the web rapidly has created problems to share, find and store relevant information. This is a networked Web using technologies and methodologies that enable the users to search and find information automatically online as soon as it is published by its authors; instead of using some software which polls a specific server to find updates [237]. The new social media has enabled the online users to query instantly occurring data using latest real-time web search technologies [237]. Examples of real-time web are Facebook's newsfeed, Twitter, social networking, online search, and news portals.

In the technical terms, there is significant importance of pushing data to the clients by the servers, without a client's request. Previously the updates when arrived on the servers were only shared with the clients upon a request originated by a client. However the server push technology has made this possible, that a server may share the updates to clients as they appear. So such communication needs a type of connection, between the client and the server, which may able to work in both directions and any party may start the communication. The communication may be continuous with the corresponding real-time server, during every user connection.

6.3.1 Simplex or Full-duplex communication

In simplex communication data travels in one direction at a time, whereas, full-duplex communication transmits in both sides. It is useful in situations when either sides want to start communication; or server wants to respond quickly [354].

6.3.2 Connection Oriented Options

Hypertext Transfer Protocol (HTTP)

The first document of HTTP protocol was released in 1991, which is a request-response protocol in which the connection is closed after a connection initiation by a client and response by the server [158]. For the real-time the connection needs to be permanent, since either side may transmit data any time. HTTP/1.0 was released in 1996; and after three years, HTTP/1.1 was released as a standard [158], with the introduction of HTTP persistent connections. This allowed several request to be made at the same time using single connection[279].

HTTP/2.0 is the next-generation protocol for transferring information on the web. It is initiated from Google, in 2012, and is based on SPDY [356], to tackle a number of performance gripes which the company had with the traditional HTTP, such as HTTP/1.x uses multiple connections to load resources in parallel. The objective is, why not to use a single connection? In HTTP/2, multiple bidirectional streams are multiplexed over a single TCP connection [356]. The new specification recommends to support 100 different streams over a single connection. Being a binary protocol, this will transmit non-human readable messages rather than just the text as in case of HTTP/1.x. It will allow server to push streams to the clients.

XHR

The XMLHttpRequest (XHR) is an API used for the browser-server communication based upon an independent channel [166]. The JavaScript is used to transfer custom data format (XML, HTML, plain text or other format) to and from a web server using HTTP transport layer. The browser fires the *onreadystatechange* callback event, each time it receives new data from the server and parses the data using the JavaScript. AJAX technology is based upon XHR, which uses JavaScript to send and receive data behind the scenes between a Web browser and a Web server instead of reloading the entire Web page.

Polling

To see either a communication party is still connected and want to communicate polling is used, in which a program or a device continuously checks the other programs or devices to see what state they are in [1]. This can be the simplest approach to provide updates to the clients from the servers. A server can instantly sends back a response, either containing new data or just an empty response if there was nothing to retrieve upon a poll request from a client. This has serious

[1] whatis.techtarget.com/definition/polling

flaws, such as how to set the interval to prevent empty responses. Polling at regular intervals, and sending data with the HTTP responses (as in case of piggybacking [116]) may provide advantages but they are not robust.

Long Polling

It is same like polling but it utilizes the keep-alive headers of HTTP/1.1 protocol, in which the connect is kept open after the client's Ajax-style request [116]. This works in all browsers which supports XHR, upon an asynchronous clients request, the server's response can contain encoded data (typically XML or JSON) or JavaScript to be executed by the client. Whereas, in script tag long polling a request can be made to work across sub-domains. Since the previous polling techniques can be used across different second-level domains (SLDs), due to browser security policies designed to prevent cross-site scripting attacks [159].

HTTP-streaming

HTTP-streaming is an old technique introduced by Netscape during 1992, even before the HTTP/1.0 standard [80]. Page streaming and service streaming are its two forms, which receive streaming contents from the server using a long-lived always-open TCP-connection. A common implementation of this is the so-called "forever frame", or iframe, that receives script-tags in an everlasting response from a server [326]. The Browsers execute code in script-tags when they read it.

Comet

In 1996, with the start of Netscape 2.0 [111], the two-way communication was possible by embedding Java applets into the browsers using raw TCP socket [350]. Event notification can be in any format, text or binary. In 1996-98, the very first browser-to-browser application was Tango Interactive [64], which was mostly used for distance learning. Long-polling and HTTP-streaming are often referred to as Comet or Comet Programming [319].

In **Comet** web application models a long-held HTTP request permits a web server to push data to the browser, regardless of any origination of request arrival from the browser[116]. Comet is a general term and semantically encapsulates many related techniques, which mainly reply upon the Javascripts. The other known names for Comet are several, such as Ajax Push [143, 197], Reverse Ajax [116], Two-way-web [248], HTTP Streaming [248], and HTTP server push [128] among others [276]. The first set of Comet implementations date back to 2000 [41], with its Pushlet [374] and Lightstreamer[2] projects. Pushlet was open-source

[2] lightstreamer.com

and based upon server-side Java Servlets, and a client-side JavaScript library.

In 2001, Chip Morningstar and Douglas Crockford designed a Java-based (J2SE) web server and a client respectively, which used two HTTP sockets to keep open two communications channels; and transferred the messages based upon the JSON format [273]. In 2006, Comet techniques were introduced through some social applications, such as (i) Meebo[3]: a multi-protocol web-based chat server which facilitates the end users to connect to Yahoo, Microsoft, Hotmail, AOL chat platforms through the browser simultaneously; (ii) Gmail: added the chat feature; (iii) JotSpot[4]: now this is actually Google sites.

Server Sent Events (SSE)

This takes advantage of the "text/event-stream" content type of HTTP/1.1 to push messages to the client [112]. While using an HTTP connection a browser receives automatic updates from a server. This is a one way communication channel from the server to the client, but the client has to connect first "subscribe" to the channel, then the server will send the new data through events when the data is available, hence it is similar to long polling. The SSE EventSource API is a technical component of HTML5, available for the programmers [5]. In this the connection kept open but ether side can close it. One can also provide time period to limit the connection opening time.

WebSockets

In December 2011, the WebSockets was proposed by IETF as an independent protocol standard, to solve the HTTP's lack of abilities for bi-directional communication between a server and a client. Such full-duplex communication channel operates through a single socket over the Web. It is just not another enhancement to the conventional HTTP communications; rather it posses much advance features, especially the real-time, event-driven web features [287]. However, this uses HTTP as a transport layer to take benefit from the existing infrastructure. "This enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that do not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>and long polling)" [157].

[3] techcrunch.com/2011/11/15/meebo-reaches-250-million-people
[4] sites.google.com
[5] w3.org/TR/eventsource

To establish a WebSocket connection, the WebSocket protocol uses the same ports as HTTP and HTTPS (80 and 443, respectively), which allows it to perform the initial handshake using the classical HTTP, but later it can open a bi-directional channel upon the client's request. Sending messages back and forth once the connection is up, is a lot more efficient and fast than what HTTP can provide. If one side leaves the other side gets knowledge about it. Data in request/response headers in HTTP may accumulate to hundreds of bytes [246], while WebSockets sends messages in frames with only two bytes overhead [286]. A frame can be sent both sides simultaneously eliminating the need for more than one connection. A programmer can use the WebSocket API to develop customized applications [187].

6.3.3 Drawbacks of HTTP Based Approaches

Polling, long polling, HTTP streaming, Comet, Server Sent Events are the different approaches to get real time communication based upon HTTP. Somehow they all work in the same manner, either by keeping the connection open or pushing data to the clients. Long polling is mostly use since proxies have least effects. One of the major issue is how to determine interval time. In order to achieve real-time, one might desire that the client should make a new request each time it receives the last response. This will create a lot of load on the server. Dealing with this would require some serious load balancing technology, leading to an expensive solution. A short interval will lead to many of empty responses, whereas a long interval will make the application least real-time. In long polling keeping requests open for a longer time, will provide the opportunity to reduce many unnecessary requests. Though if the server receives updates at a high rate, the connection will never be able to stay open. Each time the client tries to initiate long-polling, there is always some data waiting for it and the server responds at once [246]. This effect makes long-polling work just as regular polling at a short interval, hence long polling will become polling.

With HTTP-streaming and Server-Sent Events, the server is the initiator. One can configure SSEs to work almost same like long-polling, but it uses push technology, same like streaming. Having an open connection that sends a lot of data, gives rise to the problems like to have proxy-servers and firewalls [246]. Such proxies or firewalls will create a lot of latency to the client because of the buffering process using the HTTP. To solve this, many Comet-based streaming solutions again switched to long-polling if the server or the clients have to do lot of buffering. In case of forever frames the programmers have to handle the incoming scripts extensively. The EventSource API facilitates the developers to handle incoming events. We have pure event-handler middlewares, which guarantee not to write the extra code to clean up the incoming data. But again at the end, SSEs

are still HTTP dependendents, which have still many issues as discussed above.

HTTP was never designed for real-time

The HTTP/1.1 features, such as keep-alive, chunked encoding and persistent connections; support data transmission for real scenarios. The claim that HTTP supports the real-time with these features, is in fact true. Since the headers in HTTP requests/responses may consists upon hundreds of bytes [246]. Benefits of HTML5 Web Sockets are many, especially an incredibly dramatic reduction of unnecessary network traffic and latency by comparing it to traditional mechanisms, as discussed above. P. Lubbers et al. have compared polling to Web Sockets in [287], and the results are incredible. For example, "HTML5 Web Sockets can provide a 500:1 or depending on the size of the HTTP headers even a 1000:1 reduction in unnecessary HTTP header traffic and 3:1 reduction in latency" [246].

HTTP is unidirectional, whereas the WebSockets is a standard protocol which is developed only for the purpose of bi-directional communication [157]. Using HTTP several connections are required to make enable an application for real-time communication. For example, in case of the SSEs, a developer requires one connection to push an event to the client and at least one for the client to send the messages to the server [246].

6.3.4 Client-Server Interaction for Data

After understanding the connection oriented technologies, the next stage is how to receive data, what functionality to adopt? How the client and the server will interact with each other. The four basic functionalities are discussed below:

Simple Messages

If only a very small piece of data is to transfer from the server to the client, then a simple *onMessage* type of functionality will be enough. EventSource API and WebSocket API offer such onMessage functionality, and SockJS is one of the [6] available real-time solution with focuses on basic messaging and connectivity.

Publish/Subscribe pattern

We discussed the pub-sub systems in previous chapters in section 1.4.1, 3.3.5. Such systems are very popular in real-time data-driven environments. With pub-sub solution one normally subscribe to a Channel, Topic or Subject, such as in YouTube channels, or on facebook pages. As the data updates on the channel, the

[6] github.com/sockjs

subscriber also gets those updates accordingly. Otherwise the user also binds to certain specific events to receive updates related to those. Such systems are mostly based upon event driven architectures. Pusher[7] is a real-time pub-sub solution to build applications.

Data Synchronisation

In such a scenario the focus is mainly upon keeping all the application's data synchronized to each other. In data sync solutions the focus is upon manipulating the data structures and to get assurance that all the application must have the same version or copy of the data structure [237]. For example, Google Docs try to keep the document synchronized when multiple users are accessing it simultaneously. Firebase[8] is a real-time solution to build synchronized applications.

Remote Method Invocation (RMI)

As we discussed during the previous chapters sections, such as 3.3.1,3.3.2; RMI is same like RPC apart that the former's communicate is based upon objects. In pub-sub network calls are relatively clear, due to channels and topics; whereas RMI network call to the object methods are hidden. SignalR real-time framework possess these features, see section 6.3.5.

6.3.5 Real-Time Application Frameworks

There exist many real-time web application frameworks [237]. Some of those are mentioned in the previous section, following is a brief summary for some of the popular self hosted frameworks.

SignalR

SignalR[9], very recently became part of ASP.Net, is a real-time solution to build RMI oriented applications. This provides two separate levels of abstraction for real-time applications -i.e, Hubs and Persistent Connections. The low level persistent connections extends the WebSocket API [51]. The framework uses Hubs[52], which provide a way of defining objects on the server and the framework creates client-side JavaScript proxy objects to expose the server-side functionality [237]. It has clients for both web applications, Windows Phone and applications, ordinary Windows Store apps, and even Android platform.

[7] pusher.com
[8] firebase.com
[9] asp.net/signalr

Socket.IO

This is a library for Node framework (sec. 6.3.6) that with the goals facilitate real-time application development possible every platform, browser or device, focusing equally on reliability and speed [25]. This is care-free real-time, with a cross domain support and 100% in JavaScript. It extends WebSockets API by letting the developers to introduce their own events including the events supported by the API itself. A user can send an event to all the clients at the same time using a namespace in Socket.io. In contrast to this, by using a common WebSockets API, a user is forced to keep an array of all connections and loop through it to send messages to all clients. Because Socket.io is the best to send customized events (to target a set of clients) and uses an EDA (3.3.5), which facilitates the development of pub-sub model based application development. Such namespaces supports the development of pub-sub systems, and introduces loose coupling, since in this is the typical way in which coordination middlewares work [354].

Following socket.io code, which first performs handshake upon HTTP's port 80, and then by using the configuration loads the *node-tweet-stream* package to communicate in an event-driven mode. The *tw* object emits the *tweet* event upon receiving a *tweetr* event. In asynchronous programming the callback function (i.e., *function (tweet)*) will execute at the end, to return the value or exits [198].

```
var io = require('socket.io')(80);
var cfg = require('./config.json');
var tw = require('node-tweet-stream')(cfg);
tw.track('socket.io');
tw.track('javascript');
tw.on('tweetr', function(tweet){
  io.emit('tweet', tweet);
});
```

Web Socket Client Libraries

Since Socket.IO is based upon the Web Socket protocol, therefore for the client there are also available a list of libraries. There are many clients with different languages based technical platforms, such as JavaScript, Microsoft.Net, Java Virtual Machine (JVM), C++, etc. Following we describe a list of such libraries that may be used to write client applications for a WebSocket protocol based web server.

WebSocket-JS[10] is a JavaScript library in the HTML5 Web Socket implementation which is powered by Flash. *AS3WebSocket*[11] is an ActionScript 3 WebSocket client libarary. *Silverlight WebSocket client* is a Javascript wrap of Silverlight WebSocket implementation. *ArduinoWebsocketClient* is a WebSocket

[10] https://github.com/gimite/web-socket-js
[11] github.com/theturtle32/AS3WebSocket

client for Arduino [12]. *WebSocket-for-Python*[13] and *WebSocket-client*[14] are the two client libraries for the applications having business logic developed in python. Similarly *UnittWebSocket*[15], is an API that allows the developers to use the web sockets in the Java/Android projects; and *Java-WebSocket*[16], is a barebones WebSocket client and server developed in Java technology.

For the Microsoft .Net framework, we have WebSocket4Net[17] as the client library, which originated from SuperWebSocket[18] WebSocket Client. It supports WebSocket client implementation for many different kinds of runtime, such as Net 2.0, .Net 3.5, .Net 4.0, Mono, Silverlight, Window's Phone, Xamarin.Android and Xamarin.iOS. *System.Net.WebSockets*[19] namespace provides a managed implementation of the WebSocket API for the programmers. *Anaida*[20] - WebSocket Client/Adapter with 3 versions for Java, Silverlight and C#. Microsoft *MessageWebSocket class* makes enable the network communication by allowing reading and writing the data messages using a WebSocket [263]. *WebSocket Sharp*[21][22] is a library based upon the WebSocket protocol for the C# technology.

Atmosphere

This is a real-time Client Server Framework[23] for the Java servers based using JVM. This is the most innovative technology that uses a JavaScript client supports WebSockets with cross-browser fallback support to transport transparently [24]. It provides an API that is more low level than the Hubs API of SignalR, but a little higher than the persistent connections of the same library.

SockJS

This is a browser JavaScript library[25] for the developers to have objects that resemble the WebSocket interface for the low latency, full duplex, cross-domain communication channel establishment between the browser and the web server.

[12] github.com/krohling/ArduinoWebsocketClient
[13] github.com/Lawouach/WebSocket-for-Python/tree/master/ws4py/client
[14] pypi.python.org/pypi/websocket-client/
[15] code.google.com/archive/p/unitt/wikis/UnittWebSocket.wiki
[16] java-websocket.org
[17] websocket4net.codeplex.com
[18] superwebsocket.codeplex.com, www.nuget.org/packages/WebSocket4Net
[19] msdn.microsoft.com/en-us/library/hh159285.aspx
[20] anaida.codeplex.com
[21] sta.github.io/websocket-sharp
[22] github.com/sta/websocket-sharp
[23] github.com/Atmosphere/atmosphere
[24] async-io.org
[25] github.com/sockjs/sockjs-client

Figure 6.1: Real-Time Web Technologies [237]

SockJS provides cross-browser, WebSockets fallback-support, HTTP streaming, HTTP polling, JSON polling and EventSource support [237].

XSockets

XSockets[26] is an event driven library for .NET, and resembles with the Socket.IO library. On the server-side it uses an approach that resembles with the controllers of ASP.NET MVC.

Lightstreamer

This [27], is a project of an Italian company i.e, Weswit[28]. This framework is for the application development using real-time messaging. It is for the web, mobile, tablet, desktop, and IoT applications, hence it offers different APIs. It's server itself is written in Java, but through the use of an adapter based model, you can make server-side code with JavaScript, C# or Java. There is also a long list of client-side APIs[29] including a JavaScript client for web browsers.

[26] xsockets.net
[27] lightstreamer.com
[28] weswit.com
[29] lightstreamer.com/doc

6.3.6 Web App Development and Stacks

NodeJS (Node)

NodeJS[30] is a JavaScript runtime framework that is built upon Chrome's V8[31] engine, which is a Google's open source high-performance JavaScript engine. It is a command-line tool that can be run as a regular web server and lets run the JavaScript programs in side it. It has been written in the C++ technology and is used in Google's Chrome browser. V8 engine is compatible with many platforms, such as ECMAScript[32], IA-32, ARM or MIPS processors. This is especially good for the applications where one want to maintain a persistent connection from the browser back to the server. Other programming languages such as Ruby on Rails, Django would create immense load on the server, because each active client eats up one server process. When one uses Node, the server does not require to maintain separate threads for each open connection. This means that a browser-based chat application, developed in Node, will require almost no system resources to serve a great number of clients.

JavaScript is exceptionally well furnished to a callback-based concurrency model, and Node dominates here [343]. Node is lightweight and efficient due to its usage of event-driven architecture and non-blocking I/O model [369]. It can serialize and de-serialize the JSON native to both the client and the server in a proficient native manner. Socket.IO in combination with Node.js will reduce the real-time latency even further than what is possible with long polling in the traditional manner.

There are available a lot of Node packages to build the big ecosystem of Node applications using the *npm*[33], which is the largest ecosystem of open source libraries in the world. Such components can be added into the the processing middleware stack of any Node application, which run on each request made to the server. There exist many MVC pattern (ch. 5) frameworks for Node Java Script applications [223]. One can have as many numbers of such component middlewares that can process each request one by one in a serial manner. Any one of them can change the request, alter data, and then pass it to the *next()* middleware in the chain.

[30] nodejs.org
[31] developers.google.com/v8
[32] ecmascript.org
[33] www.npmjs.com

Express Framework

Express[34] is a fast web framework for the flexible Node based development, and provides a robust set of features for web and mobile applications based upon HTTP communication. It provides best performance with a thin layer of fundamental web application features, without hiding the Node features that anyone wants to use. Many frameworks are using this, such as LoopBack[35]: Highly-extensible, open-source Node framework for quickly dynamic end-to-end REST APIs development; Sails[36]: a MVC framework (ch. 5) for Node to develop applications; and MEAN[37]: JavaScript framework that simplifies rapid Web application development.

AngularJS

AngularJS[38] is to extend HTML vocabulary for creating customized web applications. It provides buit-in services on top of the XHR, and facilitates a developer to code in an easiest manner to handle the asynchronous communication and data manipulation.

Meteor

This is a set of new technologies to develop very easily high quality real-time web applications[39]. It is with Node.js, fullstack HTTP, Streaming HTTP, long-polling, WebSockets features [40]. There are many different Web Application Frameworks but none of those is similar to Meteor. Real-time is the heart of Meteor, and it uses SockJS. All code with Meteor is written in JavaScript, but interesting enough, it is not a Node framework [41].

6.4 Data Streams

A stream is a continuous flow of incoming data records, and such streams are vastly used in healthcare monitoring applications based on wired and WSNs, in social networks, and in the Internet of Things (IoT) [171]. Current trends like

[34] expressjs.com
[35] loopback.io
[36] sailsjs.org
[37] mean.io
[38] angularjs.org
[39] www.meteor.com
[40] guide.meteor.com/
[41] docs.meteor.com

IoT, comprise various new requirements, has opened new research areas for the scientists, one of those is handling of data streams, that is why Data Stream Management Systems (DSMS) are in hype. In healthcare data considering temporal aspects are vital.

There are three main differences while comparing DSMS with the DBMS, first they do not directly store the data persistently rather keep the data in the main memory for some time for autonomous predictions to respond to outlier values, such as fire alarm, emergency situations, such as in healthcare domain etc [186]. Therefore DSMS computation is generally *data driven*, i.e. to compute the results as the data is available. In such cases the computation logic always resides in the main memory in the form of rules or queries. Whereas DBMS approach is *query driven*, i.e. to compute the results using queries over permanently stored data. Therefore, because of *data driven* nature, the very first issue which DSMS must solve is to manage the changes in data arrival rate during a specific query lifetime.

Secondly, it is not possible to keep all the previous streams in the memory due to their massive nature. Therefore only a summary or synopsis is kept in the memory to answer the queries whereas the rest of the data is discarded [292].

Thirdly, although we can not control the order of the data arrival but it is critical to consider the order of the arrived data values, hence their temporal attributes are essential. In order to handle the unboundedness of the data, the fundamental mechanism used is that of *window* - which is used to define slices upon the continuous data to allow correct data flow in finite time [186].

6.4.1 Data Stream Processing

Traditionally, the real-time bulky data streams are dealt with traditional data processing infrastructure or custom coding techniques including Commercial-off-the-shelf (COTS) software and services; but those have appeared to be inflexible and have conviction for high costs [345]. At the same time, several existing technologies, such as memory DBMS and rule engines, are also being re-promoted after some adaptation to address stream processing challenges. In addition to this a new class of software infrastructure has become prominent as a solution to low-latency, stream processing applications i.e., Stream Processing Engines (SPE) for example, Aurora [91], TelegraphCQ [94] and STREAM [49].

6.4.2 Software Technologies for Stream Processing

Apart from the custom programming techniques, which highly attached to the use case logic and data format; there exist at least three different software technologies that have the potential to be used to solve the high volumes of streaming data with

low latency [345]. Following is brief description of those technologies; which are: (i) DBMS, (ii) Rule Engines, and (iii) Stream Processing Systems.

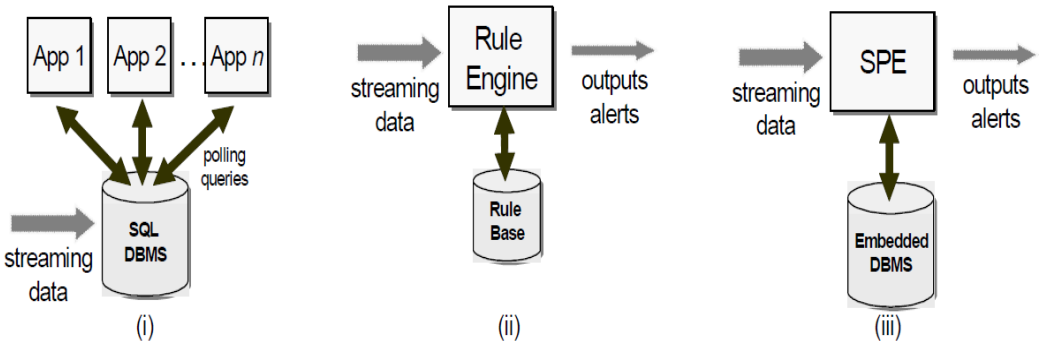


Figure 6.2: Basic architectures of (i) a database system, (ii) a rule engine, and (iii) a stream processing engine [345]

DBMS

They store and manage high volumes of data and provide results against the complex queries. In memory DBs do not store the data at the disk and are very efficient for data processing than the traditional DBMS. Figure 6.2 depicts how the streaming data is loaded into a DBMS by an application, or it is inserted directly. Applications can manipulate the data using the run-time arguments provided by the user. DBMS based stream processing is a type of *process-after-store*; and are passive since they wait for the application to send an SQL query to further do any thing [345]. Another option is to use triggers, but those are not portable and scalable.

Rule Engine

Rule engines are in common since 1970's, which were initially proposed by the artificial community. Then in 1980s, we got Prolog language to program logically by defining general-purpose rules. A rule is actually consists upon condition/action pairs, which are mostly expressed by "if-then" statements in the traditional programming languages. Such a kind of rule engine observes the stream data and checks it against the provided rules, as depicted in Figure 6.2(ii). It also generate events upon meeting some condition like this. A set of rules is called a rule base.

Stream Processing Engines (SPE)

These are the special programs designed to deal with the streaming data, and their basic architecture is depicted in Figure 6.2(iii). Without storing the data persistently, such SPEs apply SQL-style processing upon the in memory data to infer results. Such systems use special constructs or primitives to program the logic for the incoming streaming data.

Table 6.1: The capabilities of various processing systems

	DBMS	**Rule Engine**	**SPE**
Keep the data moving	No	Yes	Yes
SQL-style strem processing	No	No	Yes
Handle stream imperfections	Difficult	Possible	Possible
Predictable outcome	Difficult	Possible	Possible
Stored and Streamed data integration	No	No	Yes
Distribution and Scalability	Possible	Possible	Possible
Instantaneous response	Possible	Possible	Possible

To provide high-level guidance, Stonebraker et al. has presented eight rules in table 6.1, which a stream processing system must deal [345]. Such rules are same like we see for the DBMS[104, 103] and online analytical processing systems [**?**].

6.5 Summary

The chapter presents an overview upon the real-time system and web technologies; as well as the importance of sequenced data streams in the real-time scenarios. Real-time Web mainly involves how to push data to the client without its special request, which demands a special connection oriented mechanisms that must allow communication in both the directions. Such features are provided by some HTTP based connection technologies, such as HTTP persistent connection, Polling, Long Polling, HTTP streaming, Comet, Pushlet and Server Sent Event (SSE). HTTP's initial targets were not to tackle the real-time processing, therefore a special full-duplex persistent protocol was developed latter -i.e., Web Sockets. Real-time scenarios are intensive data-driven and data handling is an issue, different technologies are the candidate solutions, such as OnMessage, pub-sub systems, data synchronization and java based RMIs. To use these technologies application frameworks are available possessing different characteristics. Streaming data needs special processing technologies that consider the sequence factor in the arrival messages beside the prolific arrival of the data. DBMS, Rule engines and Stream processing Engines are the available mechanisms to handle the streaming data nature.

Chapter 7

Big Data

Without data, you're just another person with an opinion.
–W. Edwards Deming

Proliferation of structural, semi-structural and no-structural data, hence Big Data, has challenged the scalability, flexibility and processability of the traditional Relational Database Management Systems (RDBMS) and even Object-Oriented Database Management Systems (OODBMS). The next generation systems demand horizontal scaling by distributing data over autonomously addable nodes to the system. For schema flexibility, they also want to process and store different data formats. The solution is in NoSQL approaches, also called as IoT databases, which are considered vital nowadays. Monitoring applications transmit the data continuously after certain time intervals and data contains time as property. So, the key research aspect is to investigate scalability, schema flexibility and temporal aspects altogether.

7.1 Introduction

The emergence of Web 2.0 systems and millions of users have played a vital role to build up a global society, which continuously generate volumes of data. Where as, this data tsunami has threatened to overwhelm and interrupt the data management systems. Hence constant data growth, information storage, support and maintenance have become big challenges while using the traditional data management approaches like RDBMS or OODBMS. Scaling can be achieved in two ways, such as vertical or horizontal, where the former means adding up resources to a single node, whereas in the latter case we add more nodes to the system [260]. This flood of data has not only affected by its size, rather has also opened up many

challenges because of different data types, hence it demands more robust mechanisms to tackle different data formats. The Web, e-science solutions, sensor laboratories and industrial sector produce in abundance both structural, semi-structural and non-structural data [85, 37, 75]. The new generation systems do not like to fix their database schema to a single structure, rather they want their schema to evolve in parallel to an entity data type's adaptation, hence they want flexibility. Alongside, previous methods of centralized or distributed storage with static data impose constraints on addressing the real-time requirements, therefore we need to find robust temporal data storage mechanisms, which can support real-time integration of data volumes for efficient query processing while using less storage space, and NoSQL approaches are keys to these [141, 117].

Above discussed problems of data proliferation, data structure flexibility and data processing urge to find alternate data management mechanisms, hence NoSQL data management systems have appeared and is now becoming a standard to cope with Big Data problems. These new data management systems are being used by many companies, like Google, Amazon etc. The three primary types of NoSQL databases that can be categorized by their data models are: key-value stores, column-oriented databases, and document databases [190, 315].

7.1.1 Relational Database Management Systems (RDBMS)

RDBMS stores the data in the form of tables, where each table represents an entity or its relationship with another entity. A logical grouping or a set of attributes when combined together forms a table. A management system operates the table data according to a set of constraints which keep the data consistent and concurrent; for more details refer to [146].

Scaling: Relational database systems fail to scale the applications according to the incoming data traffic, because they are not much flexible to allow data and read/write operational distribution over many servers; whereas they can easily support vertical scalability by adding more resources to a single server, which do not solve the problems arose by data proliferation at a bigger level.

Different Data Formats: This is not a new problem, and roots back in the history of Object-Relational DBMS (ORDBMS), under the name of Object-Relational Impedance Mismatch [258, 219]. This mismatch is natural, when we try to model an object into a fixed relational structure. Similarly, the digital information with different structures, such as natural text, PDF, HTML and embedded systems data, is not simple to capture as entities and relationships [218]. If somehow we do this, it will not be easy to change afterwards, hence such mechanisms are rigid to schema alteration because they demand pre-schema definition techniques. Same is the case with ORDBMS, which also demand pre-schema definitions, use multi-joins, as well as the tables for the classes need to be defined in advance.

Performance: RDBMS use two-dimensional tables to represent data and use multi-join transactional queries for the consistency. Although they are matured and still good for many applications, but recent research issues show that they lack robust storage and performance especially while dealing with extensive data using multi-join queries [184, 241]. This problem highlights more when extensive processing is required to find hidden useful information in big data volumes. The data mining research is not the focus in the thesis and can be found in [303, 59, 183, 176, 321].

7.1.2 Object-Oriented DBMS (OODBMS)

For the OODBMS [281, 379, 146] which combine database capabilities with Object-Oriented Programming (OOP) language capabilities; and which store complex data and relationships between the data directly. There are other advantages also, such as (i) no Impedance Mismatch ; (ii) parent class contains all sub-objects, so no joins are required since the class hierarchies handle the relationships; (iii) No query language, since interaction with the database is done by transparently accessing objects; (iv) no primary keys; and a single data model can represent dynamic operations also beside the sub-classes and properties. But there are many disadvantages also, such as (i) no alteration of schema model independent of the applications, as they are tightly bind to each other; (ii) as one model mainly that holds all the data, so update to a single class will lead updates to the other classes also that interact with that class; (iii) no language independence, as they are typically tied to a specific language and are accessed through specific APIs; and (iv) in RDBMS we can run Ad-hoc queries which create new tables though joins, but the semantics of joining two tables is different than joining two classes. OODBMS are although much faster than RDBMS but the design of the system is very strict and any alteration to a single instance may lead the application to stop and wait. They may handle different formats of data but they don't handle volumes of continuous streaming data with scalability characteristics.

7.2 Data Streams and Timestamped Data

The data streams was discussed in section 6.4 in the previous chapter. The real-time monitoring applications generate continuous or sliced (as in our case, see section 9.5.1) data. In the latter case the discrete units of data elements arrives in a sequence, one after the other. This chapter will present a robust stream processing and storage approach that considers the temporal aspects during the periodic arrival of the JSON data elements. The next generation systems produce temporal data continuously in abundance, especially in monitoring scenarios. For example from the various domains (e.g. health care, banking, etc.) and sources (e.g. sen-

sors, social networks or mobile devices). This data is always in large quantity and its storage and processing has been an issue for the researchers [232, 171].

7.3 Limitations of DBMS

As discussed previously *data-driven computation*, *unbounded streams* and *timestamped data* are the three main issues that have arose while dealing with streaming data, such as during sensor data acquisition in monitoring scenarios. Alongside the problems already discussed in the introduction section (i.e. data proliferation, schema flexibility and data processing) and the problems appeared because of streaming and time-series scenarios have put serious limitations upon the legacy structural approaches of RDBMS. So, previous time-focused models, techniques and algorithms are no more applicable to solve these problems, hence leading to Big Data issues, such as *volume, velocity, variety and complexity* - and will be discussed more in the upcoming sections.

These novel research challenges demand investment in finding time-focused big data models, storage algorithms and strategies that could work in real-time scenarios. Cuzzocrea [117] investigates the temporal aspects of the Big Data management and present the state of the art in this domain. He also provides a list of future directions, such as *modeling, indexing, in-memory processing, security, storage* and *scalability* of temporal Big Data etc. Therefore as a pathway to progress, we require temporal big data management for real-time data with new challenges to tackle. Since Big Data is a recent forthcoming technology which has brought a number of advantages to the enterprise world, so it has become mandatory now to solve the challenges and the miscellaneous associated problems.

7.4 Big Data Management Frameworks (BDMF)

A Big Data management framework means the organization of the information according to the principles and practices that would yield high schema flexibility, scalability and processing of the huge volumes of variety of data, but for which traditional RDBMSs do not suite well and have become impractical. There is a need to devise new data models and technologies that can handle such Big Data. Recent research efforts have shown that BDMFs can be classified into three layers that consist of file systems, database technology, and programming models [190]. However in this thesis the is database technologies only in context to the healthcare domain with real-time and temporal perspective.

NoSQL also be interpreted as the abbreviation of “NOT ONLY SQL” or ”no SQL at all” [92], whereas this was first used by Carlo Strozzi in 1998 for his

RDBMS that did not offer an SQL interface [349]. NoSQL databases are often used for storing of the Big Data in non-relational and distributed manner, and do not provide ACID characteristics of RDBMS [303].

7.4.1 The *V*s of Big Data

The *V* of Big Data is paramountly refer, even in healthcare, to as mainly for *Volume, Variety, Velocity* and *Veracity* [303]. The first three *V*s have been introduced in [67], and the *V* for *Veracity* has been introduced by Dwaine Snow in his online blog *Thoughts on Databases and Data Management* [337].

Volume prolific data at scale creates issues ranging from storage to its processing.

Velocity real-time data, data streams — data stream analysis, data snapshots in the memory for quick responses, availability for access and delivery.

Variety many data formats (structured, unstructured, semi-structured, media).

Veracity deals with uncertain or imprecise data, its cleaning before the processing. Variety and Velocity go against it as both do not let clean the data.

7.4.2 NoSQL Database Categories

Based on the differences in the respective data models, NoSQL databases can be organized into following basic categories as: key-value stores, document databases, column-oriented databases and graph databases [190, 315, 179, 303].

Key-Value Stores

These are systems that store values against the index keys, as key-value pairs. The keys are unique to identify and request the data values from the collections. Such databases have emerged recently and are influenced heavily from Amazon's Dynamo key-value store database, where data is distributed and replicated across multiple servers [123]. The values in such databases are schema flexible and can be simple text strings or more complex structures like arrays [347, 277]. Its data model simplicity make possible to retrieve the information in a very fast manner, hence include real-time processing of Big Data, horizontal scalability across nodes in a cluster or data centers, reliability and high availability. Few key-value databases store data ordered on the key, while others do not. Whereas few keep entire data in memory and while others use it after writing it to the disk permanently with the tradeoff replying to the queries in real-time. Their scalability

and durability rely on two key mechanisms: partitioning and replication and object versioning [190].

It is like efficient lookups and reminds of simple abstractions as file systems or hash tables. Amazon's Dynamo and Voldemort[32], which is used by Linkedin, are two examples of systems that apply this data model successfully. Other databases that use this model of data category are as: Redis [20], Tokyo Cabinet [28] and Tokyo Tyrant [29], Memcached[1] and MemcacheDB[2], Basho Riak[3], Berkeley DB [17] and Scalaris [21]. Whereas Cassandra is a hybrid of key-value and column-oriented database models[4].

Column-Oriented Databases

Relational databases has their focus on rows in which they store the instances or the records and return rows or instances of an entity against a data retrieval query. Such rows posses unique keys against each instance for locating the information. Whereas column oriented databases store their data as columns instead of the rows and use index based unique keys over the columns for the data retrieval. More columns can be added to it easily and a column can be further restructured called super-column, where it contains nested (sub)columns (e.g., in Cassandra) [303]. Google's Bigtable played the lead inspirational role towards the development of this category of databases [95].

Google's Bigtable is a compressed, high performance, scalable, sparse, distributed multi-dimensional column oriented database built over a number of technologies in Google's infrastructure, such as *Google File System (GFS)* [170], a cluster management system, SSTable file format and Chubby [88]. For performance it can be indexed by rows as well as columns and a third dimension to index is the timestamp. Bigtable is designed to scale across thousands of system nodes and allows to add more nodes easily through automatic configuration.

This was the earliest popular column oriented database of it's type however latter many companies introduced other variants of it. For example Facebook's Cassandra [229] integrates the distributed system technologies of Dynamo and the data model from Bigtable. It distributes multi-dimensional structures across different nodes based upon four dimensions: rows, column families, columns, and super columns. Cassandra was open sourced in 2008, and then HBase[5] and Hyper-table Hypertable[6], have emerged to implement similar open source data

[1] http://memcached.org
[2] http://memcachedb.org
[3] http://basho.com/products
[4] http://cassandra.apache.org
[5] http://hbase.apache.org
[6] http://www.hypertable.com

models based upon Bigtable technology which is still proprietary.

Graph Databases

Graph databases, as a category of NoSQL technologies, represent data as a network of nodes connected with edges and are having properties of key-value pairs. Working on relationships, detecting patterns and finding paths are the best applications to be solved by representing them as graphs. Neo4j [16] and Allegro Graph[7] are two examples of such systems. Neo4j is an open source, an embedded, fully transactional, schema free and a Java persistence engine that supports high scalability. Whereas Allegro Graph is a Resource Description Framework (RDF) [307] triple store for linked data and companies like eBay, Walmart and Telenor are using it.

Document Databases

Document-Oriented NoSQL databases are the most general models for data management which use JSON (JavaScript Object Notation) or BSON (Binary JSON) format to represent and store the data structures as documents. These provide schema flexibility by allowing arbitrarily complex documents, i.e. sub-documents within sub-documents, documents as lists. A database comprises one or more collections, where each collection is a named group of documents. A document can be a simple or complex value, a set of attribute-value pairs, which can comprise simple values, lists, and even nested sub documents. Documents are schema-flexible [347, 277], as one can alter the schema at the run time hence providing flexibility to the programmers to save an object instances in different formats, thus supporting polymorphism at the database level [271]. Figure 7.1 illustrates two collections of documents for both students and course of the Academic Management System. It is evident that a collection can have different formats of documents in JSON format and they have hierarchies among themselves, such as courses has an attribute books which contains a list of sub-documents of different formats. These databases store and manage volumes of collections of textual documents (e.g. emails, web pages, text file, books), semi-structures, as well as no structure data and denormalized data that would require extensive usage of *null* values as in RDBMS [272]. Unlike key-value stores, the document databases support secondary indexes on sub-documents to allow fast searching. They allow horizontal data scaling over multiple servers called shards. MongoDB [9], CouchDB[8] and SimpleDB[9] are the most popular examples of these.

[7] http://allegrograph.com
[8] http://couchdb.apache.org
[9] http://aws.amazon.com/simpledb

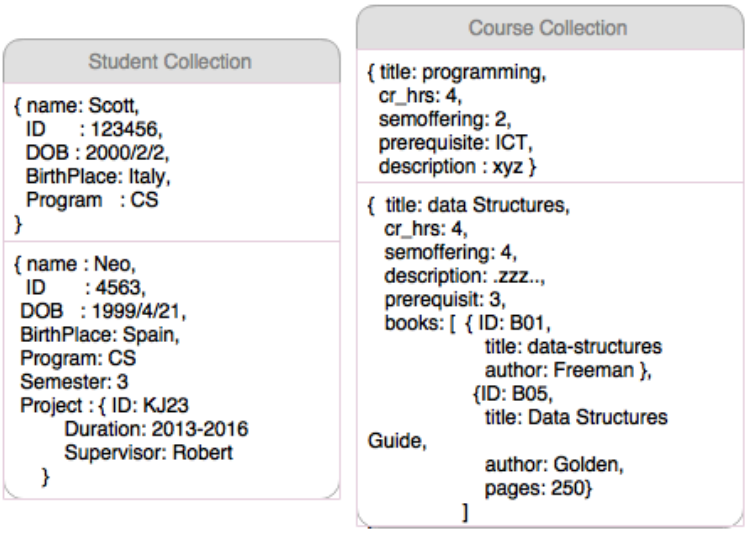


Figure 7.1: Example of a documents-oriented database for the Academic

Document Databases: MongoDB

MongoDB, created by 10gen in 2007, is a document oriented database for today applications which are not possible to develop using the traditional RDBMS [259]. This is an IoT database, which instead of tables (as in RDBMS) provides one or more *collection(s)* as main storage components consisted upon similar or different JSON or BSON based documents or sub-documents. Documents that tend to share some of the similar structure are organized as collections, which can be created any time without predefinitions. A document can simply be considered as a row or instance of an entity in RDBMS, but the difference is that, in MongoDB we can have instances within instances or documents with in documents, even lists or arrays of documents. The types for the attributes of a document can be of any basic data type, such as numbers, strings, arrays or even a sub-document.

MongoDB provides unique multiple storage engines within a single deployment and automatically manages the movement of data between storage engine technologies using native replication. MongoDB 3.2 consists upon four efficient storage engines as shown in Figure 7.2, all of which can coexist within a single MongoDB replica set [267]. The default WiredTiger storage engine provides concurrency control and native compression with best storage and performance efficiency. MongoDB allows both the combinations of in-memory engine for ultra low-latency operations with a disk-based engine for persistence altogether.

It allows to build large-scale, highly available, robust systems and enables different sensors and applications to store their data in a schema flexible manner [347, 277]. There is no database blockage, such as we encounter during *alter table*

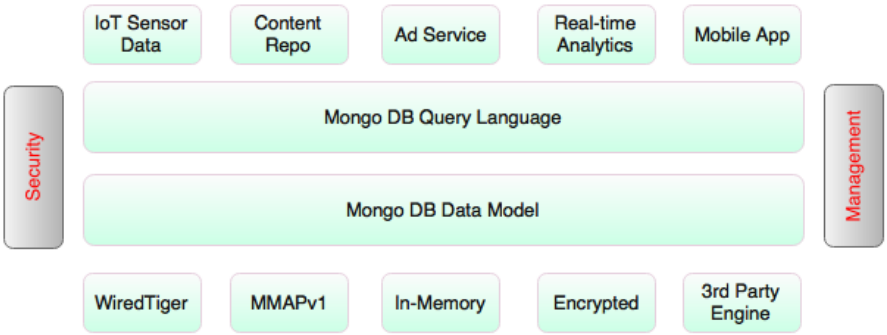


Figure 7.2: Flexible Storage Architecture, optimizing MongoDB for unique application demands [267]

Table 7.1: Understanding MongoDB terms with respect to RDBMS terms.

RDBMS	mongoDB
server	server
schema	database
table	collection
tuple	document
field	property

commands in RDBMS during schema migrations. It enables horizontal scalability because table joins are not as important as they are in the traditional RDBMS. MongoDB provides auto-sharding in which more replica server nodes can easily be added to a system. It is a very fast database and provides indexes not only on the primary attributes rather also on the secondary attributes within the sub-documents even.

In context to the data modeling point of view, next we discuss how NoSQL databases, such as document-oriented databases, allow the data structure flexibility by permitting the applications to have their own (one or more) multiple data models out of the database, hence to provide high data availability and scalability features; later we shall discuss very briefly about the data retrieval perspective.

MongoDB CRUD Operations

CRUD is an acronym for create, read, update, and delete operations against a database. MongoDB provides al these operations to manage data but in a different manner than we see in the case of SQL etc [268]. For example one have to specify explicitly and separately the collection name, query criteria, projection and the cursor modifier. As the MongoDB query shows below to find 5 names and

Figure 7.3: MongoDB find query[268]

addresses from the users collection where the age must be greater then 18. MongoDB allows to pass the query criteria and project as arguments, but initializing the arguments separately. The same above query is given below.

```
var query = { age: {$gt: 18}};
var projection = {name: 1, address: 1};
db.users.find(query, projection).limit(5);
```

7.5 Modeling Aspects

The representation of the concepts of data structures required by a database is called as a data model. A data model determines the logical structure of a database and basically decides in which manner the data can be stored, organized, and manipulated. Such data structures decide the data objects and their relationships and the semantic data rules they govern. In software engineering data modeling is a process to create data models for an information system. There are mainly three types of data model instances, i.e. *conceptual, logical* and *physical* [146]. Conceptual data modeling defines the semantics of an application domain, to elaborate the scope of the model, in order to identify the various entities and relationships needed in an application. Logical data model is similar to conceptual, but with more details, which defines the structure of a domain of information. It is also called as a schema model, as it describes the objects with their properties and relationships to determine the structure of the data. Whereas the physical data model concerns the physical storage of the data.

Although NoSQL approaches provide schema flexibility to the developers which allow them to store different formats of same entity's instances in a single storage collection, but for the rationality, sanity and demonstrating the storage structure they still have to follow some sort of format logically, at-least at the application level and then to store the data in the selected format(s). In this way one can define as many schema formats for a single entity and the same *collection* will allow the storage to all of them, given that a logical rationality has been maintained through out the definitions, as shown in figure 7.4. Applications treat each schema differently based upon different logics using advanced programming, such as object oriented principles in the form of polymorphism, classes inheritance or object

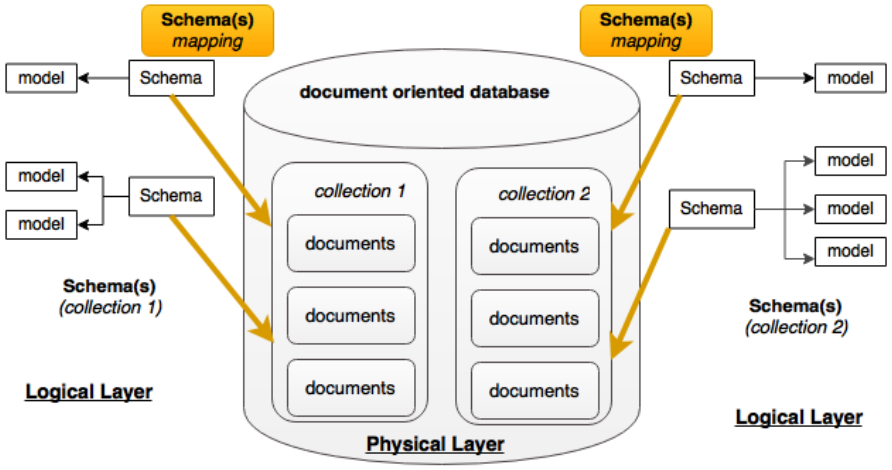


Figure 7.4: *collections*, *documents* and *sub-documents* in Document Oriented Databases can refer to one or more schema models

arrays etc. Hence developers eagerly define and model the schemas that yield robust characteristics for their problem domain in general. The initial research regarding NoSQL database modeling and insights is provided in [217, 297]. Bugiotti et al. deal with NoSQL modeling approaches in general [84]. How to have NoSQL related approaches inside the relational databases is discussed in [383]. For the initial thoughts about NoSQL modeling and its comparison between traditional relational models and NoSQL models is provided in [258]. A study about normalization and de normalization during the data modeling is done in [214].

Note that document-oriented data models allow sub-documents, hence hierarchical principles are their core features, which create place for the sub-entities to play role. A collection can refer to one or more models, as shown in figure 7.4. A schema can have a single model representing an entity or can have multiple models representing more entities (may be hierarchical). For example in figure 7.1, we model two course documents for a *course* entity. The document titled as "Programming" is quite simple, whereas the document titled as "data structures" has a tree structure where a sub-entity *books* is shown at the second level. Similarly, two structures is shown for the *student* collection, where the first with name as "Scott" is simple and the second with the name as "Neo" contains a hierarchy. Therefore the collections having different schemas are actually having different format logics and must be treated differently.

7.5.1 Schema Design: MongoDB and RDBMS

New technologies such as XML and NoSQL databases have put doubts on the usefulness of data models, but this is not the case because the significance of data models will always remain inevitable to understand and demonstrate the logical and storage structures of the data [218]. Regardless, NoSQL databases are schema flexible and support multiple schemas but in reality they actually support multiple variants of schema, which is actually some sort of structural representation, if we differentiate and treat them individually. Therefore for such databases the terms like schema flexible would be more appropriate than schema less.

Mainly most RDBMS are built around one particular data model, although it is possible for a RDBMS to provide more than one data model to an application. Such data models are fixed structure and require a database locking for a certain period of time upon migration processes, such as from one version to another version, adding more attributes to a table etc. RDBMS use schema models to generate the databases using the Data Definition Languages [146]. Database designers use the traditional approach of Entity Relationship Diagrams (ERD) to concentrate on the database structures and constraints during conceptual database design [97]. ERDs are used to identify the main entities and their attributes for which the information is required to store in the database. Besides this it is also required to identify the relationships and relationship types between the entities, which lead to a natural and logical grouping of the attributes into relations during the mapping process from ERD's to the relational schema [146].

7.5.2 Schema Design Rationality

Speaking in context to the RDBMS, during the mapping process, we still require some formal efforts to analyse why a relational schema, with a grouping of attributes, may be better than another. This *goodness* or measurement of the quality of a structural or relational schema design follows the procedures of checking data redundancy and data dependency features extensively [102]. Given an application domain and the technology mechanisms, we should investigate the available data modeling approaches in context to the given problem.

Speaking in general, during the modeling process for a problem domain, we must consider all the requirements and constraints related to it, such as temporal, streaming, distributed aspects, multiple messages formats and their attributes or properties. Such an investigation needs intensive care while modeling for the schema flexibility to allow the storage of multiple structural schemas, so that the one main goal of Big Data can be achieved i.e. *variety*. It is to note that schema flexibility, such as provided by MongoDB, can permit all vendor devices to store data simultaneously with the time-stamp regardless of different message formats.

In context to the IoT perspective, hence it supports storage independence of data related to any "things".

Goals of Normalization

Katsov in his online guide [217], discusses very general NoSQL data modeling techniques for all the NoSQL management frameworks. He says that denormalization and aggregation are the main fundamentals beside thinking about the application's addressing to the end user queries. Normalization is a process to organize data in a database by grouping attributes in a better way [146]. This is a systematic approach of decomposing tables to eliminate data redundancy and ensuring data dependencies; and includes creating tables and establishing relationships between them [264]. RDBMS do normalization because of three goals, i.e (i) To free the database of modification anomalies, i.e. to update only one row in one table against a wrong instance entry, (ii) to minimize the redesign of the database and (iii) to avoid bias towards any particular access patterns. For the third point MongoDB do not care at all, therefore with MongoDB we are not worry towards being bias a particular access pattern.

One of the big advantage of RDBMS is that they keep the data relational, normalized and enforce the foreign Keys for the data consistency. But in MongoDB it is only upto the programmer to keep such a consistency, therefore MongoDB do not provide such a guarantee. Whereas on the other side, MongoDB allow the embedding of the entire document and in this case do not have to care about even about the foreign key at all. With normalization RDBMS usually perform transactions based upon ACID principles, which MongoDB dot have to care about. However it allows atomic operations at a document level.

Normalization or Embedding-(How good is your model?)

Normalization was first introduced by E.F. Codd in the field of rational databases and is explained in many of the database related text, such as [103, 102]. In RDBMS the traditional practices are to properly normalize the database because it increases the data integrity and efficiency of the space utilization [146]. In denormalization we reject the splitting of the physical tables, and keep the data together in a single database table which have frequent access, to reduce the query processing time, as in this way we can avoid joins as well and the disk scanning at different location to answer the queries can be reduced. Following can be the database schemas in normalized forms; for the de-normalized student and course collections depicted in figure 7.1 respectively:

```
Student (ID, name, DOB, birthplace, program, semester, PID)
Project (PID, duration, supervisor)
and
Course (title, crhrs, SemOffering, prerequisite, desc, bookID)
```

```
Book (ID, title, authorID, pages)
Author  (authorID, name)
```

Kanade *et al.* [214] did a study for NoSQL databases with both normalized and denormalized forms using a similar dataset, and have found that the embedded MongoDB data model provides a much better efficiency as compared to normalized MongoDB data model. Therefore, Database designers have confronted that tight normalization degrades the system performance, hence they recommend to denormalize in many cases. Denormalization supports the data read performance, as the spinning disks have a very high latency (e.g. approx 1 ms); therefore the idea of keeping the data close and together help avoiding the disk read overheads and decreases the latency. Therefore, denormalization is a key feature of document databases and supports efficient data retrieval and optimize the data storage capacity by allowing dataset hierarchies.

7.5.3 Schema Modeling using Object Relational Mapping (ORM)

There exist a specific driver for each programming language, including few languages which are still not much popular, one can see here [269]. Recently a new trend have been seen which has provided facility to the application programmers in developing NoSQL database applications is the use of Object Relational Mapping (ORM); which related with the mapping of programming language objects to the persistent data storage in the database and then to the objects usage in the application [199]. In this way the data becomes more fluid and natural both for the application as well as the used programming language.

A beauty of MongoDB is that its documents are objects themselves, which performs this process even more easily, hence the MongoDB architecture lends itself very well to ORM as the documents. One example is mongo-Java-orm or "MJORM" a Java ORM for MongoDB [270, 174]. A plain old Java object (POJO) is an ordinary Java object, not bound by any special restriction or bogged down by framework extensions [209] [257]. One of the main advantage for using POJO is its decoupling from the infrastructure frameworks [209]. POJOs accelerate the development process by separating the persistent object properties and business logic; hence facilitating the programmers to concentrate upon one thing at one time between the business logic and persistence and transactions. Other variants of POJO are as: [13, 19, 14]. The .Net framework variant of POJO is POCO [18].

ORM based upon Mongoose

Although storing accessing and manipulating data in MongoDB is not difficult, but still application developers find it more convenient and flexible to have a

mapping layer between the application and the NoSQL database, as discussed above. Such a service is provided by an application layer middleware, i.e. Mongoose [15], [252], which is written in JavaScript and is designed to work in an asynchronous environment; and helps in validation, type casting, object mapping, query building and building business logic. It delivers a direct, schema-based solution to model an application in MongoDB database. We can think of that it is consist upon the concepts that are not part of MongoDB and are out of the box for a database management system.

7.5.4 Time Series data Modeling in MongoDB

As streaming data contains timestamp values within each sequenced message element. MongoDB handles timestamped data in a very robust manner and many organizations are using it [295]. Some proprietary services are also available and popular in the community, like MongoDB Management Service (MMS) which models and displays the time-series data as a dashboard [8]. This facilitates a developer to develop visualization and alerts on different database metrics.

Schema Design for Time Series : A Case Study

One can design as many schemas as possible for MongoDB; as NOSQL databases are flexible for this purpose. But, we learn in section 7.5.3, that being rational, to is beneficial to have a mapping schema out of the box; thus facilitating the programmers in developing scalable distributed systems. This section discusses a blog, describing a method to model time-series data in MongoDB [295] , [10].

Before start designing for a NoSQL database, first let us first observe, how a RDBMS handles a time-series data? This is quite straight forward, i.e. by storing each event as a row within a table. For example, a monitoring system stores a temperature sensor measurement in relational format, as shown in table 7.2: But,

Table 7.2: TimeSeries data storage in a traditional RDBMS.

timestamp	temperature
2016-04-10T22:04:23.000Z	24.1
2016-04-10T22:04:24.000Z	24.1
2016-04-10T22:04:25.000Z	24.0

this **doc-per-event** approach saves a document for each row in MongoDB.

```
{ timestamp: ISODate("2016-04-10T22:04:23.000Z"),
  type: "temperature",  value: 24.1},
{ timestamp: ISODate("2016-04-10T22:04:24.000Z"),
  type: "temperature",  value: 24.1},
{ timestamp: ISODate("2016-04-10T22:04:25.000Z"),
  type: "temperature",  value: 24.0}
```

This approach does not take advantage from the document embedding, so there is a lot of latency in reading the data; for example around 3600 reads for 1 min data, if the sensor transmits one message per second [10]. An alternative is, to have a single **document-per-minute** and to save the temperature value against each second. Such an approach is better and more optimized in context to the storage as well as data retrieval. The approach is shown below.

```
{ timestamp_minute: ISODate("2016-04-10T22:04:23.000Z"),
  type:"temperature", values:{0:25.1, 23:24.1, 24:24.1, 59:22}}
```

This model contains sub-documents as values, which are embedded in the main document. To store one value per second (against a minute), we can simply represent each second as separate fields between 0-to-59. Such kind of approaches effects upon the efficiency of the systems, in context to the number of inserts and updates. For example, for one hour read 60 total documents need to scan; and some systems copy the entire data to a new location against an update, which is not robust when the data is in large quantity. If the database also has to perform indexing after the copying process, it will affect adversely. A feature of MongoDB is that it can manage in place field-level updates as long as the size of the document does not grow more than a certain limit (i.e, 16 megabytes), which may also contain sub-documents [12]. However GridFS API of MongoDB provides flexibility to store a document more than the maximum size [11]. Conclusively, because field level updates are more efficient, so the document-embedding is better than the relational design. Writes will be faster as: for updates (one per second) than for inserts (one per minute) [295]. Because instead of writing a full document at a new place we will actually request a much smaller update that can be modeled as described bellow for the temperature collection:

```
db.temperature.update({
  timestamp_minute: ISODate("2016-04-10T22:04:23.000Z"),
  type: temperature}, {$set:{values.23: 24.1 } } )
```

A more compact approach is to store per second data at hourly level, i.e. **doc-per-hour**. It has two variants: (i) by second, and (ii) by minute. In the former case seconds are stored from 0-to-3599 for an hour. This approach cause extra workload during update operations, because to update the last second 3599 steps are required. To avoid document movements pre allocation of the structure space is beneficial. However the later case stores per-second data at the hourly level but with nesting documents for each minute. This approach is also update driven, but to update the last second it requires 59+59 steps. The following MongoDB query depicts the later model and the update query to modify the last second.

```
{ timestamp_hour: ISODate("2016-04-10T22:04:23.000Z"),
  type: temperature, values: {
    0: { 0:25.1, , 23:24.1,  24:24.1, , 59:22 },
```

```
    ,
    58: { 0:25.1, , 23:22,  24:22, ,  59:22 },
    59: { 0:25.1, , 23:24,  24:24, ,  59:24.2},  }}
```

To update the last second:

```
db.temperature.update( {
    timestamp_minute: ISODate("2016-04-10T22:04:23.000Z"),
    type: temperature }, {$set: {values.59.59 :24.5 }})
```

7.6 Big Data Analysis

It is to note that distributed NoSQL systems, such as MongoDB etc., are mainly for the large volumes of data storage or data integration purposes, rather than the data analysis purpose; so complex multi join queries are not supported by them [214]. As as solution to this is the utilization of the MapReduce [122, 258]; which compiles SQL-like queries to a sequence of jobs and consolidates the results at the end of each job; such a feature has been introduced by SQLMR in handling tera to petabyte of data in cloud data management [189]. For the data comparison and analysis between different collections we have available other different technologies also, such as aggregation framework [7], MapReduce [122, 258], Hadoop [303, 190], Tez[10] and Hive[11] etc. These processing techniques are with the goals of descriptive and predictive analysis of the data, which are valuable for the business intelligence purposes to provide valuable services [321, 59].

7.7 Summary

Volumes of unstructured data has challenged the scalability, flexibility and processability of RDBMS; and NoSQL DBMS have filled this gap, as they can handle volumes, of varieties of continuous data efficiently. Key-value stores, document-oriented, column-oriented and graph are the categories of NoSQL databases. MongoDB is a document-oriented database, which doesn't require schemas; but programmers need them within the application for the business logic. Such schemas are placed out of the DBMS; and the challenge of their timestamped based modeling can be solved with the help of different approaches, such as: (i) save timestamp with each event; (ii) save per minute; and (ii) save per hour. MapReduce, Hadoop etc. are used to analyse such data.

[10] http://tez.apache.org
[11] http://hive.apache.org

Part III

Conceptualization

Chapter 8

Towards a Real-Time Distributed Healthcare Architecture

I hear and I forget I see and I remember. I do and I understand.
-Confucius (550 BC-479 BC)

The chapter discusses and highlight the requirements related to the phases involved during a healthcare oriented distributed monitoring process. We shall suppose a general problem scenario, in context to the SHE to solve it to have a general architecture solution using M2M interoperable technology. To get high technical requirements and the level of complexity involved, to build a flexible and scalable distributed architecture, for ANT+ protocol based monitoring environment will be explored using a prototyping approach. The prototype is planned to a answer the research questions in a step by step manner.

SHE, being an area of Gerontechnology and AmI is an integration of ICT and smart technologies with distributed ubiquitous features (see 1.3.1). The research scientists are with the goals to adopt internet enabled new networking technologies with real-time features to provide a QoL to the society. Their solutions should have more flexible, interoperable and scalable architectures; but this also opens theoretical and technical research challenges, as stated in sections 1.6 and 1.6.2.

The technical demands range from the IoT vision for "Things" or Smart Objects' identification and integration - to - the global Internet based communication oriented platform; to build the big ecosystems. To meet new research demands and challenges of the emerging societies, HCM systems demand flexible and scalable distributed architectures; with real-time data-driven processing features. Such systems demand multiple autonomous distributed components, ranging from the remote monitoring application using sensor devices to the advanced

sensor data management and analytical processing systems. In addition, their distributed components consist upon many sub components with diverse functional and technical features. The distributed components within a healthcare system must communicate with each other through message passing over a robust communication medium. We suppose and state a scenarios to make the things understandable to the readers and researchers, therefore a general use-case is presented below. This will provide us a visionary scope and help us to learn how we design and develop a general solution, having a general architecture, methodology and framework to handle M2M sensor data acquisition and storage. We shall highlight the technical requirements along the problem complexity, to build our flexible and scalable distributed healthcare architecture in a robust manner.

8.1 Healthcare Use Case

A retired office worker, named *Dave* at the age of 65 years, lives in a three room apartment with a small garden. His only son and his son's wife work in a nearby town. Dave has got first respiratory disorder last month. He got his treatment from the doctors using an ambulance service and had stayed in the hospital for few days. Now he doesn't want to reside more in the hospital nor he can visit his physician daily. So he wants to stay with his family and also to continue the treatment at home. His doctor has provided him the medication but also asked him to measure the heart beat and blood pressure continuously and to be Keep In Touch (KIT). The doctors asked him to note his daily activities, such as total distance covered on foot, number of calories burned and steps taken daily. Since he moved to his home, he needs robust and continuous HCM mechanisms. For work his family members go out of home daily. He and his family members have decided to consult a software company (who has an alliance with the hospital and doctors), which has taken the responsibility to provide him with a robust innovative ICT-based monitoring system to be KIT with others. Such a system will not only measure the required life sign vitals but will also keep the family members, as well as the doctors, updated about his living condition by capturing and transmitting the body life parameters on per minute bases, at the minimum.

8.1.1 Use Case exploration and being practical

Following we explore the technical aspects of the above given software problem in the domain of Ambient Assisted Living with the goals: to be KIT, to be healthy, to be productive and to age well at home in order to have a QoL. There are around 22 million people effected with heart failure [357]. Careful heart monitoring, regular exercise and better daily activity routine strengthen the cardiovascular system,

improve blood circulation and normalize the blood pressure [196]. A detailed study is also presented in [302]. Weight, blood pressure, steps taken, distance covered and calories burned are some fundamental parameters; those if managed carefully, will help in maintaining a healthy body [274, 336]. Such parameters contribute to achieve, healthy routine, timely check-up, diagnostic decisions and treatment; hence at a large scale this leads towards a healthy society.

The software company decided to provide *Dave* with the first prototype of the product, which will do HCM and shall keep others update about patient's situation. This prototyping methodology will present a limited initial model to get further requirements 1.5.4. The main goal, the software company has to achieve as a whole, is stated in the introductory chapter as the main research question, in Section 1.6.1.

Given the above goal, the company realizes that the very first objective, to develop a Healthcare system, is to identify the the main building blocks, to investigate the available mechanisms and to realise the required software and hardware components. Because the lack of interoperability between devices has made the application developers much inflexible, as we read in Section 1.4.1. Beside this low energy consuming communication mechanisms (2.7.3, 2.7.4), miniaturization, device profile oriented M2M characteristics (2.8) are considerable features. Therefore selecting appropriate sensor technology is important. Discussion related to this task is provided in hypothesis formation Section 1.6.2.

After choosing the devices, the next stage to deal with is, to make enable the sensors as functional units within a network; and this requires the integration of the devices with a specific given configuration and deployment settings. Considering interoperability, the devices' communication and integration are the three tasks, with which the company encountered with during the initial stages. As also asked in the following research questions (1.6.1):

HC1: What kind of Smart Objects (SO) or wireless sensors would be appropriate for a given AAL problem?

HC2: How to communicate with the sensor technology?

HC3: How to integrate the sensors to build a monitoring network?

As an answer to these, the company did an overview of the available different sensor technologies, as discussed in section 2.7, and decided that at an initial stage they have to deal with the WSN that can operate within short range of 25-50 meters (m); and they also don't want to have frequent battery replacements. They need integration of the devices because only in this way they would able to provide more services. But for this there must be a standard communication technology between the devices also, other wise they will have to add a layer

between the devices that may work as a translator/gateway for the commands and data, which is a highly cumbersome process to do. The company also realized that, different sensor devices using different communication format and standards may cause them to develop different kinds of management, or communication features. May be one for each of the specifically chosen device type. This will also create problem to their programmers to deal with different APIs, protocols and data formats; hence also difficulty in making software updates upon any change in the device specifications etc. As a result more cost and efforts are required, so the company should choose an efficient strategy that may yield optimum results.

8.1.2 Sensor Hardware and Communication Protocol

A comparison between different competitive WSN technologies is given in Section 2.2. Company's system analyst found number of smart devices (2.7), after an extensive market search. He/She found that different vendors now provide interoperable communicating devices and support M2M interactions (1.4.3).

As an answer to the first two questions HC1 and HC2, the analyst considered the company's preferences about smart object devices, and found the required characteristics are present in the ANT+ sensor technology (2.8), which suits very well to build BANs and PANs to monitor a patient's environment. due to its different advantages, such as: M2M based interoperability, miniaturization, wearability, portability, support for vast network topologies and a long list of available devices with device profiles as options to chose. Its most promising factor is the support for long time battery utilization as it consumes very less energy. Using these embedded wireless solutions ones can target both healthcare as well as fitness applications. In this way their built ANT+ ecosystem will continually able to deliver the peak of innovation to the market. The exact strength of the ANT+ technology, is the reliability of the ANT+ interoperable standard and the possibility of the collaboration with global industry vendors.

The company identified that to provide the minimum set of monitoring services to the user, they have to monitor the patient in three different perspectives, i.e. the heart beat, the temperature of the room, and the activities of the patient. According to the activities point of view, they decided to concentrate upon the following factors: steps taken, the distance covered, duration of walk, strides, speed of the walk and the accumulated values of distance and steps. They found that the industry leaders has build ANT+ enabled sensor device solutions that can work for them [367]. By using ANT+ devices their developed monitoring system may quickly evolve into an ecosystem and will allow the integration of other connectable solutions from leading vendors. Consequently, *Heart rate* [132], *Environment* [131] and *Stride Based Speed and Distance Monitor* [133] are the device profiles for heart rate, temperature and foodpod or step count, running or walking

and speed provided by ANT+ Alliance [367]. In this way one can cover the three monitoring perspectives. The set of sensors along the device profile icons, which the company selected initially to work with, are displayed in the figure 8.1. The brief description is given below:

Figure 8.1: ANT+ Selected Sensors [367]

WTEK's [378] HS-2+ [377] : It monitors a user's heart beat with the help of an innovative heart rate embedded sensor. It has a thickness of only 8 mm and a diameter of 4 cm. This is also compatible with the *Garmin, Timex, Suunto* and *Mio* display watches. It contains a winged hook system using which one can clip the sensor device to any wrist or sweatband. One can activate the WTEK optical heart rate sensor by pressing the button, and the green LED turns on/off thus preserving battery life.

Garmin's Foot pod [365]: This is a small sensor easy to attach to the shoelaces or fit in the mid-sole pocket of the shoe. The small replaceable watch battery's power is enough for more than one year. It also measure the running cadence to help optimize the training and work. In contrast to other pedometers, this sensor uses an advanced MEMS inertial-sensor technology to detect and analyse a user's movement and is sensitive to the variations in the stride-length to achieve 98 percent accuracy for speed and distance.

Garmin's tempe Wireless Temperature Sensor [366]: This sensor collects the ambient temperature and transmits the data wirelessly to a users compatible device, such as a display device, ANT+ enabled computer or mobile. It can easily be attachable with the pack, jacket or shoe. It accurately tracks the the temperature and stores the minimum and maximum temperature values (in the previous 24 hours) also beside the current temperature.

8.2 Prototype 1: Healthcare Monitoring Application

They targeted the heart patients in general, and planned to develop a monitoring application having a flexible architecture for a desktop system, which will

automatically configure the ANT+ sensors and shall display the transmitted information in real-time to the user. By taking advantage of the ANT-FIT [364] and ANT-FS [360] technology (2.8), the system will encode and store the health information about the user locally; and will share the data later with the remote server for the permanent storage and data analysis.

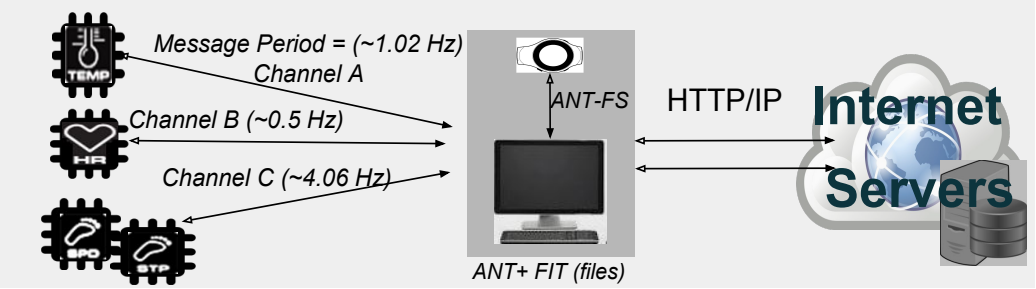


Figure 8.2: Sensor nodes with the specific communication channels to build the basic WSN, operating on 2.4 GHz frequency [C2]

Using the development process for the Windows platform, as we discussed briefly in previous chapters in section 2.8.4, the company's next task was to build a BAN using the selected ANT+ wearable technology sensors described previously. Such a network is depicted in the figure 8.2. The unique BAN operates on 2.4 GHz and each sensor device plays the role of slave receiver only after establishing a secure communication channel. The technical details, how to implement an ANT+ WSN are provided in the appendix section .1.1. Each sensor operates at different channel configuration settings as depicted. The data collecting device, such as a personal computer or a display device as shown in the figure 8.2, collects and stores the data in FIT format files; which are available for sharing with the host server over the Internet. The system functionality is as follows:

Heart rate is the speed of the heart measured by the number of heartbeats per unit of time typically beats per minute (bpm). The system displays the speed of the heart or number of heartbeats, operating time and the previous heart rate of the user. It displays the current environment temperature in Celsius (C°) or Fahrenheit (F°), minimum and maximum temperature since the device had started operation. The foot pod sensor acquires information about the user's foot strides, cadence, distance covered (in meters), calories consumed, accumulated distance and strides count since the last battery change, and transmits the data messages to the PC gateway.

The desktop interface of the implemented prototype is depicted in the figure 8.3, which displays the processed sensor data to the user in real-time. The interface contains five portions, as: (I) menu bar on the top, (ii) three panels, each for patient, configuration and connection with the server., (iii) sensor data display

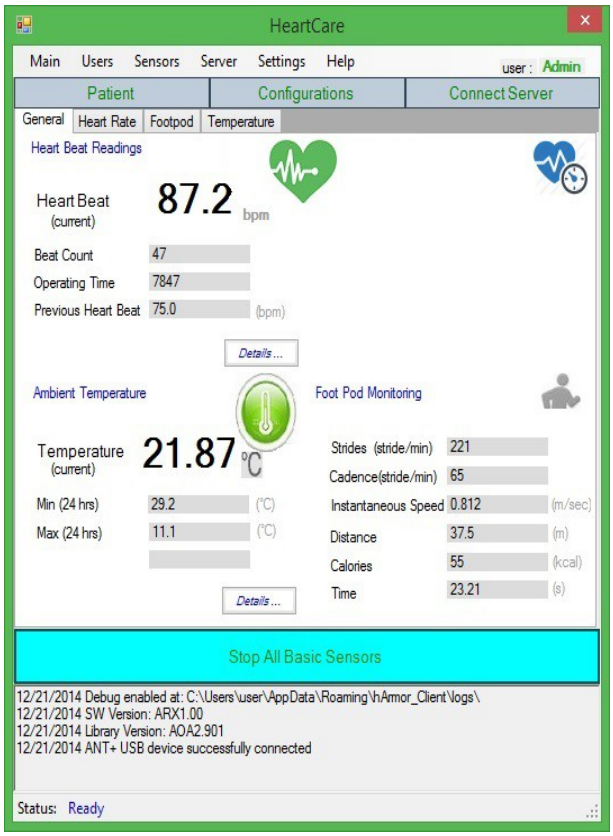


Figure 8.3: Heart care monitoring application [C2]

portion, a total of four tab controlled panels; one specific panel for general data display and then a specific panel for each sensor device; and this also contains four portions for the real-time values display in textual format; (iv) the cyan coloured big button start/stop all the sensors; and at the last (v) a log panel showing the working of the system at the bottom. At the bottom on the left side the status of the system is shown.

After performing authentication the system displays the current user and updates the interface according to his rights. It also updates the application status based on the ANT+ protocol status. The different statuses are: *Closed, Ready, ANT-FS, Broadcast and Returned to Broadcast.* System monitors ANT channels while displaying such statuses. It uses a parametrized approach to start up the application and to start sensor devices. These parameters, such as device configuration and application settings are stored in an XML file. A user can set the sensors to start automatically or can also start them all together manually. He/She can start/stop an individual sensor. The application only opens the required ANT+ communication channels, however the remaining channels stay close.

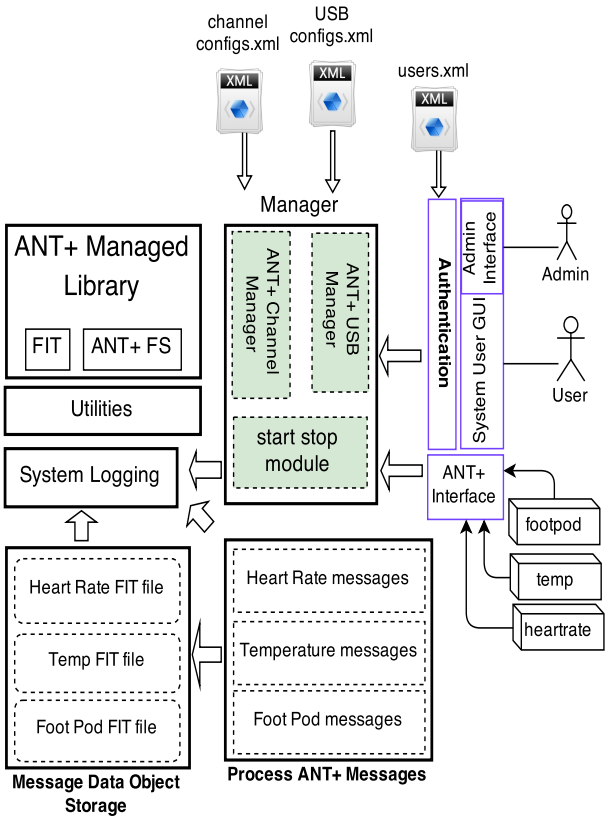


Figure 8.4: System Modular Architecture [C2]

8.2.1 Prototype Architecture

The company realised that to provide a robust system to the user they need to identify the main components of the system, so that each individual module must be dealt individually and would be available for the reuse purposes in the long software engineering process. This task, as stated in the introductory chapter in section 1.6.1, corresponds with the following research question.

HC5 What components are required to develop a basic healthcare system?

System components

As an answer to the research question HC5, the architecture of the prototype, is shown in figure 8.4, which describes the different modules and high-level logical components with their relationships. The users interact with the system through the interface and can take advantage from the system services after secure authentication. Both normal and admin users have different levels of access rights.

The system holds a logging module also, which records the debugging warnings and errors in the textual files. The application also displays the log information within a small panel, at the bottom of the interface, as shown in the figure 8.3. The application possess a separate Utilities module that caries small functions, such as time conversions or bit interpretations etc, available for all the modules. The system uses ANT+ managed libraries which also hold other libraries, such as ANT File Share (ANT-FS) and Dynastream's The Flexible and Interoperable Data Transfer (FIT) format extensions which we discussed in the previous chapters in section 2.8. Following sub sections describe the system components and will map the implemented or required tasks with the research questions pointed out during the introductory chapter in section 1.6.

Making Communication Enable

The company identified that to communicate with the sensors they need to make enable the computer for this purpose, the relevant information has been discussed in section 2.8.4 and the task is also stated in the introductory chapter in section 1.6, pointed out by the following research question.

HC2 How to communicate with the sensors?

Each sensor communicates with the system through ANT+ enabled USB stick interface, which opens a channel on the ANT+ enabled USB stick interface device having 8 channels in total. The Manager module manages the ANT+ Universal Serial Bus (USB) stick and its associated different channels for the communication. This Manager module accepts the USB configurations from a structure format (an XML file) to configure and control an ANT+ enabled USB interface device [C2]. The module also provides an interface to start and stop individual communication channels.

Different sensors vary in configuration settings and communication message formats. Therefore, for each sensor device, the USB interface establishes a separate ANT channel with appropriate message rate, Radio Frequency (RF), device type and ANT channel ID. A user can start and stop all or an individual sensor through the interface. The system supports auto start and stop feature for all the sensors. The system can load individual configuration parameters from an XML file and can also use the default settings to establish an ANT channel with a sensor.

8.2.2 Data Interpretation

The company identified that there are many reasons that a sensor data must be processed in real-time before the final delivery or storage; such as interpretation to enhance the data semantics, translation of the data within a given situation to

capture a more abstract contextual concept, data analysis to build relationship with the environment, data processing to shift the load from the server side to the data collecting side and interchange formatting of the data for more meaningful data wrapper objects for the transferral. Examples of such data processing and transferring features can be seen in [172], [327]. Such data processing must always be, sensor protocol specific, sensor device specific or use case dependent. Therefore this aspect needs to be discussed with respect to two dimensions; (i) data processing and (ii) physical location of processing [C2].

To deal the monitoring problem using different sensors in a general manner, they need a general message interpreter, instead of a specific message interpreter for each sensor device. Following research question points out the same task (1.6).

HC4 How to process and interpret different sensor messages?

Therefore, it is important to distinguish and interpret each communication message at real time before the storage. ANT+ nodes send a lot of information in the default 8 bytes payload, where first byte (ranges between 0-255) is used for identification [359]. This protocol supports more than 60 different message types and same is the case with the generated events. The system, on the runtime, has to check and understand message related information, a kind of meta data, such as device type, message data pages and message types of each upcoming message. Although ANT+ managed library provides the functionality to differentiate the messages based on their identification, but different sensors may generate same message ids, such as broadcast or acknowledge messages etc. The problem becomes more complex when the system needs to communicate with more than one device through a single interface. Also the software can not keep a separate module for message interpretation and processing for each different profile [C2, J1].

The programmer solves this problem by having a single static module for all the ANT+ messages i.e., *Process ANT+ Messages* as depicted in figure 8.4. This module further contains three smaller functions to distinguish and interpret an individual message based on a sensor device profile. ANT+ sensors send data in the form of data pages. Firstly, page definition is received which is followed by the page data itself [C2, J1]. For each sensor device, the application manages both these message types and also interprets individual message bytes. As mentioned in section 2.8, the four data types, supported by an ANT protocol are: *broadcast, acknowledged, burst*, and *advanced burst* data [140]. Beside this the protocol also supports the extended message format of 12 bytes, as in section 2.8. The displayed data is for the broadcast type, whereas the module handles the other types also. For more realistic results, it is better to perform testing to check and compare the results with another custom devices.

Secondly, for the processing location, the company pointed out two alternatives: (i) to process at the data acquisition end, where the sensors might reside,

and (ii) to send unprocessed data to the server, but this will create the load and will slow it down. Adding interpretation logic to the server will make the server inflexible in context to minor updates; or any addition, with respect to any sensor type message format, will also cause the server rigid and inflexible. Consequently, after processing these messages the application stores the extracted data in relevant temporary objects. The discussion about the permanent data storage is given in the next upcoming sections.

8.2.3 Interoperability

The monitoring system developed by the company provides data interpretation interoperability. Interoperability 1.4.1, the ability of systems and devices to work together, is one major research goal in ICT domain. ANT+ device profile based M2M interactions are the key to support and build vast ecosystems [367, 140]. The other devices which also adhear to the same data format standard will also work with the implemented system regardless of the manufacturing vendor. Therefore, the heart rate sensor used in this system can be interchanged for any other ANT+ heart rate sensor, from any other manufacturer.

8.2.4 Data Format, Transmission and Storage

Since, the target of the company is to provide remote healthcare by integrating all the stakeholders, such as doctors, hospitals, etc. The company also wants to provide predictive and analytical services to the user, so that a patient may able to see his health related weekly monthly or yearly reports; and he may also receive automatic events or messages. The software company has planned to provide such simple services initially. As well as, he can receive prescriptions from his general physicians and nurses. For this purpose the company needs to store the data to a permanent repository at a remote location. Such remote server would hold the data processing and analytical logic to perform data mining and other data-driven operations. For the data storage purpose, the company identified that, they need to send the data from the user's location to the remote server location. To transfer the data to the server, the company needs to decide about the transmission mode and data format. Beside this the schema of the target database is required to define according to the sensor data characteristics.

Before going into the designing and development requirement details, about each of these phases, the company decided to take advantage from the ANT+ FIT technology (please see section 2.8.1). ANT+ FIT technology facilitates a system to store hour's and day's of data in a compressed FIT format, hence this enables the sensor data sharing at the remote server as a batch processing job. Different

processing modes have been discussed in section 1.4.1. Therefore, for the understanding, we distinguish the storage of the data at the local display device from the remote server storage and refer the former as temporary storage in the compressed FIT format. Whereas the remote server storage will be the permanent storage, managed by a database system, for the detailed data analysis tasks. Following is a brief discussion about these highlighted tasks, which are also identified by the research question HC6, in section 1.6.1, and is stated as follows.

HC6 How to acquire, represent, transmit and store any kind of sensor data?

Data Representation

Here with data representation means the format of the data in which the system will store and transmit the data to the server. While modelling monitoring environments and thus, there is always a requirement for data delivery or exchange over a communication channel. For this, a monitoring system has to take account of many non-functional requirements, such as (i) heterogeneous systems' interoperability; (ii) flexibility, to permit easy modifications; (iii) lightweight data, to permit efficient transmission; (iv) truthfulness, to be consistent with the real world domain; and (v) readability, to allow data evaluation by its domain experts [173]. Therefore, the systems require a storage structure that would be interoperable, compact, flexible and extensible. This is what, ANT+ protocol provides, with the help of its FIT protocol extension (section 2.8.1). However lightweight and readability features are not supported by the ANT+ FIT protocols, as the former need an object transformation for the target platform, where as the latter case does not apply to the compressed data types since those are in encrypted machine language formats. Beside this, FIT formatted compressed files are mainly with the target to save memory and data transmission throughput, since compressed data is meant to be transferred quickly than the uncompressed one.

The system developed by the company takes advantage of this data storage capability feature and stores the sensor data in FIT formatted files in the local data collecting device. The system possesses a message data storage module, as shown in the figure 8.4, which stores individual sensor's data locally in the form of FIT format temporary. A set of FIT File Types were created as described in the 2.8.1, such as Device, Settings and Activity. For the target platform's compatibility and interoperability of the FIT files, the company generated the code using the FIT SDK 2.8.1. These files will be available for download by any device which have the right to access and data decoding capability based on FIT format. Any FIT-compliant device can interpret an FIT file from any other FIT-compliant device. Such a scenario has been depicted also in the figures 2.8 and 8.2.

Data Transmission

After the real-time data acquisition within a smart environment, one of the requirements of a successful monitoring system is the transferral of the data to the remote server, over a communication channel. Data acquisition systems posses distributed storage components that are data consumers, which contain the data processing capabilities. Since the company decided to store the data locally in the FIT format, therefore the transferral in such a scenario would be a batch processing mode. There are different alternatives to send those files to a remote server, however the best in such a case is to use the simple File Transfer Protocol (FTP) to transfer the data FIT files to the server. Then later to decode the FIT files according to the FIT principles as stated already.

Data Storage and Analysis

It is the physical storage of the data in an organized way. The main factors which effect upon selecting a good database management system are the data structure or database schema; storage format; scalability; object and instance relationships; and techniques of data gathering. An ANT+ protocol deals with many device profile types with different formats. To support the FIT file interoperability across the system boundaries, the targeted platform can decode the FIT files by generating and developing the target application module using the ANT FIT SDK [363].

However, the company decided to use the already available remote data storage and analysis servers with the FIT decoding capabilities, such as [331, 342, 284, 165, 348]. Such servers accept different FIT formatted files and after parsing them, save the user's physiological data (i.e., foot steps, activities and heart rate data) in their preferred format. The users can see the online analysis reports very quickly and can track their activities and health using such portals. Such reports are also available to the doctors and trainers who can give feedbacks to the users.

A major difference is that, remote servers are with the targets to attract more group of people, hence the people involved with exercise, training, sports, fitness and athletic activities, draw more attention of such companies, as a result elderly and disabled people are mainly ignored and fail to draw attentions. Such ignorance urges to conduct research and to invest time and effort to develop robust systems for the elderly. There are open source software available (For example, Golden Cheetah [284] contaning a rich set of analysis tools, targeting the young society especially athletics. Such software accept FIT files to decode, store and analyse the user's data, to support them in their health activities. There is possibility to download and customise such software, for the elderly and disabled people.

8.3 Discussion

Following are the pros-and-cons, discussed in context to the above described prototype and the research questions stated in the introductory chapter. The relevant research questions are stated with the features of the system to understand the scope, but it does not mean that the feature is complete and robust. The purpose of this relevance is to make clear the research domain and the requirement, so that the company may develop robust and highly technical solutions to solve the patients and elderly people problems.

8.3.1 Advantage Features

There are many advantages of the developed prototype, which help the patient and provide good grounds for the software company towards solving health related problems at a global scale. Following discussion describe some of those features, and only those research questions are which are not covered so far.

BAN based Ambient Environment: The developed prototype helps the user by showing real-time physiological parameters locally using a pervasive smart object technology by developing WSNs within a user environment. Monitoring the heart beat along the distance covered by the foot steps, are the vital signs which keep an elderly person aware of his physical activities. The monitoring application supports the user in maintaining healthy physical activities to improve his immune system. An immune system is a system of many biological structures and processes within a human body that protects against diseases. It also stores the temperature, activities and heart beat data in a compressed format for later use. This system functionality covers HC1, HC2 and DS2 research questions (1.6).

M2M based Interoperability: The system takes advantage from the M2M technology of ANT+ sensors to integrate the devices, hence facilitating the programmer to ignore device heterogeneity problems, such as different drivers, vendors, APIs etc. This functionality will overlap its research domain with the answer set goals of the HC1 & HC2 and NG1 research questions.

Modular Architecture: The system contains different modules and high-level logical components with their relationships. Such components are reusable after some customization for later purposes. Such functionality partially fulfills the requirements for the HC5 research question.

Data interpretation: The prototype contains an *ANT+ Message Processing* module which is responsible for the processing and interpretation of sensor data

and control messages, hence depicts the realization of the HC4 question 1.6.

Batch Processing: There are different processing modes (1.4.1) for the transmission of the data; and the prototype uses a batch processing technique, in which the data is transmitted as a set of instances.This fulfils partially the transmission part of the HC6 research question, which is as: How to acquire, represent, transmit and store any kind of sensor data?

Distributed Data Storage: This facilitates permanent data storage at a remote server in relational format, and fulfils HC6 question partially.

Data Interchange Format: During the transmission of the data one of the first issues is always, how to deal with the data format during the transmission process? It is important because, the receiving party may not understand the format of the sender, hence both must agree upon a common format. This is provided through the FIT interoperable format usage, hence satisfying the partial requirements highlighted by the research questions HC5 & DS2.

Additional Features: Logging, parametrized approach, utilities, sensor automatic configuration and start-stop are the additional valuable features.

8.3.2 Imperfections in the First Prototype

Although the system posses many valuable features but there are certain pitfalls which are unavoidable and are considered as vital in developing distributed system for the AAL environments, such as real-time data processing rather than batch processing to the remote server; horizontal and vertical scalability, different data format storage at the remote server; streaming data and temporal aspects within the sensor data. Following discussion provides an overview of such features as well as also presents the related research questions.

Real-time data processing: Since the system prototype stores the data in FIT files and then allow the server to accept and download those files for permanent storage. Such a feature is although valuable for health conscious people who are athletes and are young. However elderly people and especially the patients need more quick care, which much be applicable in real-time scenarios. Therefore the current system lacks this feature, and demands a communication middleware which processes the data more efficiently. Different processing modes have been discussed in section 1.4.1. Such features are also among the objectives depicted in the following research questions.
DS3 : Given the application domain, what kind of middleware will best integrate

the different components of the system and how it will handle the flow of sensor or control data within those distributed components?
DS4 : What communication means would be useful for the sensor data sharing with the context data server?

Scalability: The scalability, as discussed in 7.1, is with the goals to solve the problems at a broad global level. The system implemented by the company as first prototype is very less scalable as it supports only the vertical scalability in which one may add more resources to a single server, whereas the horizontal scalability is missing deliberately, in which one may add more nodes to the system. Hence the current prototype do not provide high scalability, and the answer of *A3* question is still unknown, which is: What kind of architecture can support high flexibility and scalability to develop a distributed AAL system?

Flexibility: AAL system architectures must be flexible enough to allow the addition of more software components into the system from different perspectives, such as addition of components at the data acquisition, transformation, transmission and storage layers. The prototype architecture is also inflexible in context to adding more interfaces for the users, since it provides only a single static desktop interface. An AAL architecture must provide flexibility to the developers to add as many interfaces to the users, since in Ambient Environments, the patients may be disable and elderly therefore addition of personalized interfaces is highly desirable. Such interface flexibility is discussed in chapter 5. Such a requirement is highlighted in the following research question A2. The current prototype do not answer this question, nor it answer the question A3, stated above.
A2: What software architectural patterns help us to engineer a software for any user with universal accessibility, at least over the web?

If the current prototype allows the data transmission of any ANT+ sensor device using the FIT compact format, on the other side such an approach is too static and highly inefficient, as discussed earlier. At the data storage layer an architecture must support the addition of general purpose middlewares. Such characteristics are also the objectives, which are depicted in the following research question.
DS3: Given the application domain, what kind of middleware will best integrate the different components of the system and how it will handle the flow of sensor or control data within those distributed components?

Different formats of Data: In context to the permanent storage at the remote server, the system does not allow storage of different formats against same profile. Although, in ANT+ protocol the basic attributes regarding a specific device profile are always same. But some vendor devices provide few additional valuable

features against the same profile. Since in smart ecosystems devices enter and leave a network too quickly and in an automatic manner. Even application programmers want sometimes to add more attributes for the business logic into the system storage structures, but the traditional relational means do not allow them to do that on the run time, as a result they have to stop the server to change the schema. Such legacy approaches are operationally highly inefficient. Therefore, a permanent data storage must support the addition of such new arguments on the run time. The current prototype does not allow the schema storage flexibility, which is a required feature in new generation applications, as discussed in section 1.6.2 as a research question in the following manner.
NG2: Given the new trends how we can provide a general storage to any device?

Beside this, new generation systems, especially the remote sensor monitoring distributed systems generation all types of data, such as structure, semi-structure and no-structure. The current system is not flexible to provide robust storage to such data types, which opens new research trends in this domain.

Critical Situation Event Delivery: AAL environments capture user data and may need to generate situation critical events, such as sudden fall, fire alarm etc. The current system is not efficient because it does not send the data events in real-time to the remote server and relies upon the batch processing of the data which is performed upon a user's request or periodically i.e., daily, weekly etc. Such requirements is highlighted by the research question, as discussed in section 1.6.2, in the following manner.
NG6: How to provide communication for user critical services in real-time?

Streaming and Temporal Data: Monitoring scenarios transmit the data continuously upon certain intervals of time and temporal factor is the main property of the data. Therefore the key point is to research the above directions in context to the management of streaming and temporal data altogether, such is requirement is highlighted by the research question (see section 1.6.2), NG4 is as.
NG4: How to handle timestamp streaming data coming from monitoring devices?

Platform is a desktop: The current prototype provides a desktop application to a user, which may be difficult to use for the elderly. Such applications are however more likely to be useful in the hospital environments rather than at a user's home. Home users need monitoring services which possess auto start and remote configurable features. Such services can be one of the operating system components; and use minimal system resources to perform long time tasks which require little to no input from the users. The related material to these mechanisms have been discussed in the Windows Services chapter 4. The research questions which

highlight the importance of this valuable features are stated as:
HC5: What components are required to develop a basic healthcare system?
DS1: What are the distributed system characteristics and how to have those in an AAL system?

User's accessibility The current prototype tightly attaches the GUI with the control and application logic, which makes it inflexible and inefficient. There are different mechanisms available, which provide interactive interfaces to the users depending upon their unique personalized requirements. Separating the Interface logic from the control and application logic not only facilitates a programmer rather also provides opportunities to add general purpose personalized interfaces. Such approaches have been discussed in the MVC chapter 5. The following research question highlights the importance of this valuable features.

A3: What software architectural patterns help us to engineer a software for any user with universal accessibility, at least over the web?

8.4 Summary

Using prototyping, with the purpose to get further requirements, the company has developed a HCM system to facilitate an elderly person; using the ANT+ M2M interoperable wireless sensor technology. The initial prototype is planned based upon the research questions and hypothesis. Although the product isnt complete, yet it provides many valuable features, such as M2M based BAN having device interoperability, modular architecture, data interpretation, FIT based structured storage and data sharing using batch jobs. Whereas architectural inflexibility, less scalability, inefficient data transmission, static structure format, streaming and temporal data handling are some of the drawbacks. Resultantly a better approach is required to find solutions having more efficient and robust mechanisms.

Part IV

A Flexible and Scalable Real-Time Distributed Healthcare Architecture and Temporal NoSQL Storage

Chapter 9

Flexible and Scalable Distributed Architecture

May be it's not always about fixing something broken,
May be it's about starting over and creating something better.

The purpose of this chapter is to solve the drawbacks, which we identified during the previous chapter. The target is to to design and develop a flexible and scalable architecture for real-time ANT+ sensor data acquisition and schema flexible storage of the ANT+ sensor data. At the end the chapter will present the relevant architectural framework as a guide for architects and developers for their future developments.

In a distributed environment, we require to develop robust general solutions to address the common data-driven concerns; ranging from sensor data acquisition to the data sharing and storage. Due to the absence of a shared memory, all communication in distributed systems is based on sending and receiving messages. The high-level targets to achieve are: better methodology, flexible architecture, context data space, middleware based integration and architecture framework for future reference. The different phases encountered during the data flow are: sensor communication, data processing and interpretation, data interchange format, data transmission and data persistency.

To offer a single-system view in the heterogeneous computer networks, distributed systems are often organized by a layer of software, called as middleware (1.4.1, ch. 3). A communication middleware is an application that logically lives (mostly) in the application layer, but which contains many general-purpose protocols. Sensor based monitoring applications are highly data-driven and need special communication middlewares that can handle real-time data having temporal properties. Database Middleware (DBMw), Message Oriented Middleware

(MOM), Transaction Processing (TPM) and Application Integration Middlewares (AIM) are few types [354]. Middlewares provide high-level communication services (3.3), among those, stream oriented and event based services suits best to environments having streaming data (3.3.4) and event notifications (3.3.5).

The system developed by the company solves a patient's problem at an initial level, by providing a monitoring application, which captures the user's physiological and life vital data using the WSNs. But the first prototype has drawbacks (8.3.2); resultantly the company decided to develop the second prototype for the patient with advanced features. ANT+ sensor communication, building BAN by configuring and integrating M2M ANT+ sensor devices, data acquisition and interpretation are the features which the company has already developed during the initial phase. Therefore, the company decided to reuse these features in the second prototype development. But this time, beside dealing with the above stated pitfalls, the new system architecture will provide the desired horizontal and vertical scalability and flexibility both to the developers and the users. The research questions AAF1 points towards (1.6.1), this same main requirement of architecture building to address. The question is stated as:

AAF1 How the common technical concerns can be represented as an abstract software architecture for an efficient HCM system?

Such a system will facilitate the integration of heterogeneous distributed components in a manner that the stakeholders would have a feeling that they are actually interacting with a single system in real-time manner. The company will integrate the distributed components using a glueing factor (i.e., middleware (chap. 3)), which will organize and bind the heterogeneous system components. The relevant research questions which identify same required tasks are listed below.

DS1 What are the distributed system characteristics and how to have those in an AAL system?

DS3 Given the application domain, what kind of middleware will best integrate the different components of the system and how it will handle the flow of sensor or control data within those distributed components?

9.1 Basic Architecture

A software architecture outlines the high-level components, their relationships and how to create them. This is to understand an environment of a system beside targeting the main goals, such as the system must be scalable and flexible both horizontally as well as vertically. It can process, translate, distribute and manage

sensor data in the context-aware tele-monitoring system. Such context data component should allow the integration of services and applications as an addition. As in Figure 9.1, the three sensor acquisition abstract level basic stages are: *acquire, transfer* and *store* sensor data. Whereas, the acquire and transmit stages can be autonomous components with their own local storage repositories and easy accessible interfaces to offer their resources. From the previous experience, the company knows that there is no shared memory between the data acquisition and the context data storage components, hence the whole correspondence among the distributed components will be based upon sending and receiving messages and events in real-time.

Figure 9.1: Three main steps of data acquisition

In context to the above drawn basic architecture, the following paragraphs give a light overview of the main sections, which are discussed in this chapter, such as: data acquisition as a Windows service, data transmission in real-time, data transmission format and permanent context data storage server. In order to identify the requirements an analysis discussion has been provided regarding the above stated three stages. The corresponding research questions are also highlighted according to the respective concepts during the textual description of the system features. The research question HC6 identifies the required tasks in the same research directions, and is stated as follows:

HC6: How to acquire, represent, transmit and store any kind of sensor data?

Although there are various methods available to acquire sensor data, but having a built-in service as a part of the operating system will have many advantages, as discussed in section 4.2. A daemon service, as a solo process runs in the background and acquires user's environment data, to transform the data and to send it to the data context server in real-time. Using a background auto startable always running process is a robust approach to handle monitoring problems (9.2).

The company realized that they need a special mechanism which overlap on both distributed components (i.e, *acquire* & *store*) to bind them despite of their heterogeneity and should also reside in between those components as a transporter and bridge. They identified that the autonomous distributed component must provide easy accessible resources through the open interfaces based upon a standard. The bridge may attach the distributed components in a manner that a user at any end of the system has feelings that either he/she is infact interacting with a single

system, and he/she does not know that which parts of the system are being used and how are those actually distributed over the network. The resulting system would be scalable, adaptive and flexible in context to the addition of hardware and software components. Such are the characteristic goals of any distributed system (sec.1.4.1), and this shares the answer set goals with above stated *DS1* research question. The company identified that their targeted distributed system should reflect the properties of an information and pervasive computing system types [354]).

As stated already that the heterogeneous distributed components will communicate by sending and receiving messages, but such a process must be very slow if based upon batch processing, as can been seen in the first prototype (sec. 8.3.2). Healthcare systems may identify very critical situations which need quick transmission of event data between the parties. There are many other factors that must be considered while selecting a transmission middleware. The details about real-time transmission will be discussed later in the section 9.3.

Traditional relational means do not support the different varieties of same instances to store in a single storage table, whereas the sensor based monitoring systems demand more flexible and scalable storage mechanisms. Chapter 7 discusses such details, especially its sections 7.2, 7.3 and 7.4. Therefore the company decided to use the NoSQL databases for the permanent storage of the patient's healthcare data. Following are the details of the additive features.

9.2 Acquisition (Windows Service)

The company decided to have an always running background process, which may acquire sensor data to send it to the server, after wrapping it in some light weight transferable messages or objects. Using this technique, the user does not have to make login to his computer to start this service, since such a service just need a restart of the PC to start automatically. The company also realized that using such a technique, they can manage the service remotely (Chap. 4).

The company was aware of the fact that the already developed features are reusable, which are: ANT device manager, responsible for sensor communication and building BAN by using XML based configuration settings to integrate smart objects (sec. 8.2.1); Process ANT+ messages (sec. 8.2.2); Utilities along the libraries. Hence all these features, will be the basic part of this service. Therefore they removed the GUI part carefully and inherited the Manager module (containing the controller logic) from the *ServiceBase* class, which is the child of the *Component* object class in the Microsoft .Net framework. The data interpretation, logging, utilities, device configuration and BAN will remain the basic parts of this service. The technical details for the service will be provided in the section 9.5.1.

The development of such a Windows service with the mentioned features, is also a better answer to the following research question.

HC5: What components are required to develop a basic healthcare system?

Before transferral the service processes the sensor messages and interprets the different devices' data uniformly (sec. 8.2.2); and it retains the data in the main memory in the form of objects containing the device profile properties. Later these objects will be transferred to the context server. The detailed discussion about these processes and features are provided in the next sections of this chapter.

9.3 Transmission (in Real-Time)

Since the previous prototype was not efficient and was static and inflexible in context to many aspects. Batch processing is more suitable to the young society, whose focus is upon fitness, sports and fresh health; and they can bear the information delivery delay; however elderly people are more serious to seek care with a focus not only to keep them fit and healthy, but also to provide proactive and primitive services in an ubiquitous manner. Such proactive services are not possible to deliver by processing the monitoring data as batch jobs. Therefore, to attain the additional efficient features the company realised that one of the basic requirements, for a successful web-based monitoring system for the elderly, is quick efficient processing and data transferral over a communication channel using an appropriate format. Data acquisition systems possess distributed components, where data is delivered to their consumers; which can be an application or a context data server, as in this case.

A real-time healthcare system is a soft real-time system; therefore, some latency is allowed [335]. Fast real-time delivery of urgent situation related events (i.e., a sudden fall, a heart attack, etc.) trigger appropriate automatic steps (i.e., calling a friend or a hospital emergency service), which are to save the lives of the individuals, hence they need to be handled using fast communication measures. Also a real-time healthcare system, based on the Web technologies, involve a large amount of sensor data exchange, as well as bi-directional event notifications between the system components. Besides this, as discussed previously, the components' heterogeneity, use case requirements, and limited resources make the data delivery process complicated; especially in the case when many clients try to connect to the same server simultaneously, or multiple data sources are accessed over the Internet via different computing nodes.

The company identified that they need to develop a communication mechanism that may transfer the patient's physiological data in real-time to the context

server. Such communication mechanism understands the interfaces of the distributed components and serves the both ends by exchanging lightweight control and monitoring data in a transparent manner; that the stakeholders at the remote components enjoy the real-time behaviour and could interact with each other any time. This involves a lot of factors to consider, such as what kind of middleware to chose having communication protocols (sec. 3.2), do they need object mapping as discussed in section 3.3.2. Since the WWW is involved, then what kind of client-server model to develop [354]. Distributed components run independent processes, then to reduce dependency (i.e., to decrease coupling) and to promote generalization, how to introduce coordination among them [354]. Following subsection discusses the different characteristics of the middleware perspective along the real-time communication goals.

9.3.1 Middleware Perspectives

Middleware communication protocols support high-level communication services, for example, that allow a process to call a procedure or invoke an object on a remote machine in a highly transparent way [354]. Similarly, there are high-level communication services to set and synchronize data streams to transfer real-time data, such as sensor data, or a video stream. In distributed systems, the general criteria or agenda for a real-time web environment based client-server interprocess communication is consist upon certain set of phases, such as *persistent or transient, synchronous or asynchronous, discrete or streaming communication, speed, connection-orientation, single or bi-direction, persistent connection, interface* and *data interchange format*. Following we discuss these communication alternatives for the middleware perspective, to find an answer to the **DS3** research question, stated above.

Persistent or Transient

As the company's goal was efficient delivery of the data in real-time, therefore using persistent middleware will cause extra delay, since it will store the data during the delivery process. So they decided to use the transient mode of the middleware in which, both sender and receiver are active and are executing entities. An example of this is, Message Queuing Service (MQS) middleware, which saves data persistently during the delivery process but cause high latency, such as in the email message delivery [354]. Also see section 3.2.1.

Synchronous or Asynchronous

The company does not want to block the transmitter until the request is known to be accepted. Therefore, for the healthcare real-time scenario, they need a transmitter which does not wait for responses and continue its work; hence, in this scenario, the transmission they chose was *asynchronous*. Synchronous or asynchronous middlewares are discussed in section 3.2.2.

Discrete or Streaming Communication

The company made a distinction between discrete or streaming communication; and in the former case, the distributed components communicate with messages and each message form a complete unit of information. In the latter situation, the streaming communication deals with carrying multiple messages at the same time, one after the other; and the messages refer to each other by the order they are sent [354], also see section 3.2.3. Therefore, in the context of the given problem, the company decided to send the sensor data as a complete discrete unit of information in sequence of time intervals, since monitoring scenarios are data-driven and highly prolific in nature (sec. 6.4, 3.3.4). They need to send the pieces of information as continuous data streams, otherwise the real meanings of the sensor data, holding timestamp values, will be lost.

Speed

Latency is a significant issue in networked systems and even milliseconds (ms) become important while dealing with industrial or healthcare problems. The usage of User Datagram Protocol (UDP) for transmission is considered very suitable because of its speed, due to its light data packets and no data delivery guarantee overhead. In healthcare, when a system has to deliver in real-time over the Internet, speed is very important, as well as the guarantee of the data delivery. But previous research shows that HTTP was not designed for real-time (sec. 6.3.3). The company realised that, they have to seek alternatives for internet-based real-time ANT+ sensor data delivery with speed and other features (discussed above) related to the distributed systems. A list of alternatives are discussed in chapter 6.

Connection-orientation or Connectionless

In connection-oriented communication a communication session or a semi-permanent connection is established before any useful data transmission; however in contrast to this, in connectionless transmissions the data may be delivered out of the order and independent to each other, over different suitable paths [161]. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are two divergent,

and are core members of the Internet protocol suite, where the former guarantees the transmission and is connection oriented but slower than the later one [354].

Developing for the distributed systems, in the domain of healthcare, the company deals with real-time critical and emergency situations. They do not require a middleware that may lose the data packets or a significant event message during the transmission process; especially during the transmission of critical event messages regarding life vital sign alerts or fire alarms, etc. The company realised that although there are available other alternate mechanisms, to assure the delivery of the data transmission; but a connection-oriented communication would be preferable in this case. There are many HTTP based connection oriented options, such as polling, long polling, Comet, etc. (sec. 6.3.2). But, HTTP based recurring connection establishing approaches are not robust (sec. 6.3.3). Whereas, while dealing with the video streaming or voice data the company may need more bandwidth and efficiency, than a faster communication medium would be preferable, in such a case a connectionless communication may be suitable.

Bi-directional Communication

The company knows that they have to send the data in both the directions; and for this, they need a full-duplex communication (sec. 6.3.1). They had the options, such as Polling (sec. 6.3.2), Push technology, Comet (sec. 6.3.2), Server Sent Events (sec. 6.3.2) and Web Socket technology (sec. 6.3.2). They found that HTTP was not actually developed for the bi-directional real-time communication, however a better option to chose is to use WebSockets (sec. 6.3.3).

Persistent Connection

The company identified that, making a new TCP connection every time for each request will add unnecessary latencies; and this will create an inefficient utilization of the network and host resources (sec. 6.3.3). So, the best option is to have persistent connections; and for this, they have different options, such as HTTP keep-alive, which will use a single TCP connection to transmit and receive multiple HTTP request/response messages 6.3.2. There are many advantages of having this technique, such as a decrease in the latency of the messages; fewer opening and closing of the TCP connections; the server may get a notification about when a client leaves; hence this creates a possibility to notice, when either side leaves; and the possibility of either side can start the communication, etc. Regardless of these, the company found that HTTP will not suit well due to the drawbacks discussed in section 6.3.3, especially in context to the permanent connections. A better option is to use WebSockets, which provides persistent connections and stays at an abstract level to allow traffic for different streams (sec. 6.3.2, 6.3.3).

Interface

Since the system will be a distributed Web system [354], where the components will interact over the Web, therefore providing universally accessible interfaces are essential. Such universal interfaces are highly desirable to bind the heterogeneous components to promote interoperability. An interface is a collection of procedures; which a client can call to use it's functionality and to pass the value parameters as arguments. A system can provide access to its components, without exposing any further detail, through the procedural/functional interfaces. Company wanted a middleware that may provide an interface between applications or parts of applications, and may allow the data transmission back and forth (ch. 3).

For the remote access, there are different techniques available to use, to expose a software component's API; such as Interface Definition Languages (IDL) - as mostly used in RPCs (sec. 3.3.2); the RESTful Service Description Language (RSDL) - as mostly used for HTTP-based web applications (i.e., typically REST web services [296]; and the Web Service Description Language (WSDL) - as mostly used in web services [100]. But RPCs bind the remote components to much, that they become tightly coupled (sec. 3.3.1). The purpose of the REST-compliant web services is to provide interfaces for manipulation of the representations of web resources, using a uniform set of stateless operations [110].

Data Interchange Format

While modelling pervasive real-time environments and thus, there is need for data delivery or data exchange over a communication channel; a complex system has to take account of many non functional requirements, such as (i) heterogeneous systems' interoperability; (ii) flexibility: to permit easy modifications; (iii) lightweight data: to permit efficient transmissions; (iv) truthfulness: to be consistent with the real world domain; and (v) readability: to allow data evaluation by its domain experts [173]. Data interchange format is similar to the concept of data wrapping; in the latter case, the data is associated with extra security and management features, without changing the underlying message contents; and to encapsulate the context data into an appropriate format. Whereas the former is to convert the context data into appropriate consistent and readable format which permits understanding and evaluation to its domain experts. Both these concepts are complementary to each other and are used as similar. Indeed, it involves both data abstraction as well as a transfer syntax for data exchange.

During the data exchange stages, the source schema based data structures are transformed into the target schema based data structures; so that, the targeted data may represent an accurate meaning the source data [127]. Data exchange process is very close in concept with the data integration process; where in the former case,

the data is actually restructured (i.e, with the possible loss of some data contents). There are many ways, to transform the data from one structural representation to another one; and dozens of source and target schema representations are possible, even within a single domain.

Therefore, to keep transmission rate and performance as the main goals, the company requires a data exchange language; which defines a set of data encoding rules, for both human-readable and machine-readable formats. In fact, this is in generally suitable with interoperation over the Internet; as well as, between the heterogeneous platforms. For example, XML enables the creation and exchange of domain-specific messages and is very popular over the internet [82]. However, JSON objects are more lightweight, serializable and AJAX friendly; hence, they are more suitable for faster transmissions [81]. The company decided to define JSON object's models holding properties for the device types. Nurseitov *et al.* [280] present a good comparison between the two technologies.

9.3.2 Client/Server Real-time Functionality

After deciding about the connections, speed and other factors, the next steps for the company was, to decide how the data will be received and published at the server end? How the client and the server will interact with each other? What real-time functionality the system will provide in the long run? Mainly the data will be JSON light weight objects, as discussed in the previous sub sections, therefore the best is to provide a processing functionality for relatively simple pieces of data between the server and the client. For this purpose EventSource and the WebSocket APIs provide *onMessage* functionality (sec. 6.3.4). Where as the other options are synchronization, RMI etc.

Moreover, one task for the company was to decide how the procedural resources may also get functional and executable upon receiving specific events, therefore the company decided to send and receive all the data using the event notifications. Such events will carry the messages data along them within an EDA in real-time. The following research question has the same research directions.

NG6 How to provide communication for user critical services in real-time?

9.3.3 Real-Time Application Framework

The next stage for the company was to chose a web application framework for the real-time application communication. There are frameworks available that support customized real-time communication base Web application development, such as SignalR, Socket.IO, Atmosphere, SocketJS, XSockets, LightStreamer (sec. 6.3.5). The company found that since WebSockets have all the characteristics

which suit the targeted healthcare use case; because it supports full-duplex asynchronous persistently connected streaming communication. By using this kind of framework they may also communicate through the events and can send and receive JSON data over the Internet in real-time. This facilitates to use the JavaScript technology so that the application could support XHR, which is infact the AJAX technology for the asynchronous communication (sec. 6.3.2). Consequently, looking on all these options the company prefered to use Socket.IO framework 6.3.5, which is best to have WebSockets and to send customized named events to a set of clients using an EDA. If they use other WebSocket APIs they might have to maintain a list of the clients within an array and then to loop over it to send a message or event to each client. But Socket.IO provides this feature by default; and it can easily be used as a middleware at the application layer with many frameworks, such as Node.js (sec. 6.3.6).

9.3.4 Real-Time Data Stream Processing

One of the requirement mentioned in the research question NG4, in the Section 1.6.1, is as follows:

NG4 How to handle timestamp streaming data coming from monitoring devices?

This requirement highlights the processing of streaming data, because AAL environments are having monitoring scenarios; similarly the company has in this use case. These pervasive environments transmit wire or wireless data in a continuous manner, which posses timestamp information with them. The real bottle neck is in understanding the requirements how to process such a prolific data and in which format to store it in repositories of choice. Different device types and message formats make the matter even more complicated, as in the case of ANT+ sensors (sec. 8.2.2). Message oriented communication needs to be managed in a different manner with the notion of continuous stream data or sequencing factor within the sensor data payload. Now the issue is, what kind of middleware does a distributed system must offer to make enable the real-time streaming data communication in the JSON format (sec. 3.3.4). This middleware property needs to be tackled separately from those discussed above during section 9.3.1.

Since data transmits in an asynchronous manner, the JSON data items transfer one after the other, but without any restriction of the completion time (sec. 3.3.4). In contrast to this, synchronous processing modes do not switch to the second data unit until the first one completes. In case of live streams (i.e., video or voice) the company has less control upon the stream tuning, as in the case of discrete units of JSON data messages. The temporal relationships between the data messages, is a highly important factor. They need a Data Stream Management System (DSMS), which considers the temporal relationship factor along the sequence of items; as

well as the format and size of the data items (sec. 6.4). To manage the sensor data, there are three basic factors. First is to manage the data arrival rate, second is to keep a synopsis of the data in the memory instead of the whole massive data stream, and the third is to define a *window* or *slice* on the data stream for the correct manipulation and management of data units in finite time.

The company decides to develop their own custom stream processing module for the real-time data, according to the above stated three main principles. Their motive is to extend the processing and analytical logic later in the long run, so to have an own Stream Processing Engine (SPE), as mentioned in Section 6.4.2. They planned to do customized development for this part of the HCM application.This is to note that the choice of repository or storage structure also affects the custom stream processing techniques. Besides this in many scenarios, especially with the optimization goals, the storage schema is also an influential factor in choosing the technique for processing the data. The continuous flow and the storage schema format of the data are both interrelated factors within the required media that processes the timestamped data. These factors have made the custom techniques to be inflexible and have conviction for high costs (sec. 6.4.1).

9.4 Storage (NoSQL based)

The third component within a basic AAL distributed architecture is the data storage component, as depicted in figure 9.1, which is the physical storage of the data in an organized way. This is autonomous and distributed in nature with its own memory, so that all the real-time monitoring healthcare applications may able to defuse their prolific data into this autonomous data space for the data persistency and analysis purpose. Real-time monitoring systems generate *Volumes* of data and need flexible and scalable storage and processing measurements, see Sections 7.3 and 7.4.1. Due to the diversity in topics and overlapping with the Big Data research domain, as we noticed during the previous chapter's Section 1.4 and in chapter 7, therefore the next chapter 10 is fully reserved for the discussion of this component along the temporal modelling based schema storage techniques. The research questions NG2, NG3 & NG4 (sec. 1.6.1), point towards the same requirements and shall be discussed in the next chapter 10 in detail with the answers. Storage schema and data stream processing algorithm are beneficial for the data-driven applications to relate the real-time aspects with the storage mechanisms, anticipating the temporal factors during the data persistency process. Following is a very brief discussion about them.

9.4.1 Storage Schema

The company knows that an ANT+ protocol deals with many device profile types and supports device interoperability. Thus, they cannot restrict the storage collections to a circumcised circle of specific device formats. They need a flexible database that may allow storage for any schema or format. In other words, the same entity's instance could be storable in more than one format and a single database relation would be able to support these features. Such is the case with the *Variety* of data in the Big Data aspects, see Section 7.4.1.

9.4.2 Stream Processing Algorithm for Storage

Section 9.3.4 discusses the data-driven nature of the monitoring applications, which generate data items rapidly in a continuous sequence of time intervals. The real bottleneck is in understanding the requirements how to process such a prolific sequential data and in which format to store it in repositories of choice. The research question NG4 identifies the requirements in the same direction (sec. 1.6.1).

Therefore, the ANT+ sensor's streaming data, in the form of JSON objects, need a unique approach for acceptance and storage at the distributed context server. As stated above, in Section 9.3.4, that the company decides to develop their own custom stream processing module, which contains a set of storage routines in the form of algorithms. Each specific storage routine corresponds to a specific device profile type. In an EDA, a particular procedural routine executes only upon receiving a specific event. Using such storage routines, the context data server accepts the JSON data and stores it in a document-oriented database. No doubt that document-oriented databases are schema flexible and allow storage of data in different structures (sec. 7.4.2). Yet to give sanity the company decides to apply a pre-schema approach which is very effective in context to rationality without losing the schema flexibility advantage.

9.5 Methodology

Handling different types of diseases and health issues in AAL domain, at any of the organizational level, such as individual, regional, national or an international level; is getting hard and expensive worldwide. Same is the case in the current use case problem, in which for the real-time measurements the company has to tackle a lot of obstacles, ranging from the ordinary simple device configuration to the complex real-time event notifications between the different distributed autonomous system components and the stakeholders. With the attainment of health management, exercise, and physical monitoring, a patient can prevent and cure his

diseases and can improve his health conditions; and the key to this is to be KIT with the patient or the old person using the ICT means. This section discusses all the phases, altogether, to present a methodology as a general approach towards solving same set of problems. Such a methodology is a guide for the system analysts and programmers working to provide healthcare measurements with flexible and scalable architectures using the state of the art key technologies, such as smart homes, real-time Web, Big Data and IoT.

During the previous chapters we discussed various technical obstacles, especially during the problem description and the research questions identification (sec. 1.6). The previous chapters' discussion regarding the use case and it's accomplishment; and the current chapter's improvements have shown that the problem is targeted to handle in general technically; so that the other patients and ICT stakeholders may also get benefits from this approach. The defined use case (sec. 8.1), is solved by providing a distributed system having a middleware which overlaps along the different components and involves many data flow stages; mainly in context to the communication aspects, such as sensor communication, information rendering, data interchange format, data transmission and real-time data storage. Such data flow phases have proved as to be the technical targets for the company. By achieving such tasks and dealing in general, the company presents and implements a healthcare system which is useful for the remote HCM, and it's architecture, methodology and framework will be useful in developing remote monitoring services using WSNs for the ANT+ enabled sensors and the real-time transmission for the NoSQL storage. Figure 9.2 depicts the high-level distributed components and their interactions w.r.t the proposed methodology that takes into account the rapid prototyping of ANT+ sensor data acquisition, delivery and storage for the healthcare systems.

Using a birds eye view, the system is composed of two main components, i.e. *Windows Data Pusher Service* and the remote *NodeJS Context Server*. Both these distributed heterogeneous autonomous components are binded together with the help of a communication middleware that uses WebSockets protocol for the real-time sensor data transmission, as shown in the figure 9.2. The *Windows Data Pusher Service* consists upon two main sub components i.e. ANTController and ANTDataPusher modules. ANTController has two main sub components i.e., Device Manager and Message Event Processing. The former is responsible for the WSN management and the configuration of the ANT+ sensor devices. The latter is responsible for the sensor's messages data real-time interpretation and extracting the semantics from the sensor payload. ANTPusher holds the ANT Data objects, ANT Models and the Data Emmiter communication service. Following we discuss all these in more detail with respect to the figure 9.3.

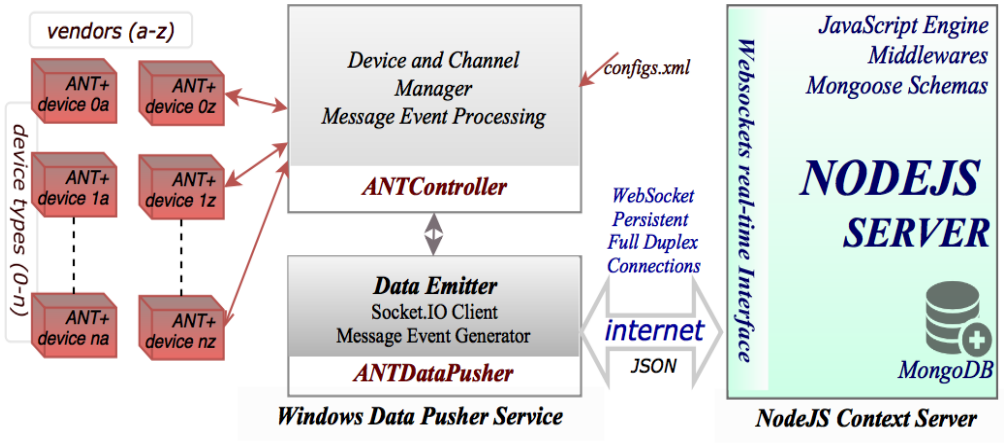


Figure 9.2: Methodology architecture for real-time ANT+ sensor data acquisition and storage

9.5.1 Windows Data Pusher Service

Starting from the left hand side, we have all the ANT+ device profile types ranging from 0-to-n and their vendors ranging from a-to-z [367]. By taking advantage of the ANT+ interoperability to build big ecosystems (sec. 2.6), the ANTManager configures any ANT+ wearable sensor device from any vendor given the setting parameters (sec. 2.8). It uses an ANT2 stick gateway, to make enable the PC to allow integration of the several sensors (sec. 2.8.4, 8.2.1). ANT Managed library, from Dynastream, helps to communicate to acquire data from wireless sensors through the serial USB stick (sec. 2.8).

To build an ANT WSN topology with the help of ANT Managed Library, the ANTController module contains two sub components: USB Manager and Channel Manager. The manager opens different communication channels for the sensor devices and uses predefined parameters (i.e., in the default case obtained from an XML file) for the configuration settings of the USB and the channels, (sec. 8.2.1 and Appendix .1.1). Each channel may have a different network number, message rate, device number, device type (as slave or master), RF frequency, transmission type, etc; for more details see Appendix .1.1.

The ANTController has a static sub-component (i.e, Message Event Processing) to process all the event messages of the sensor device channel or the USB stick. It distinguishes the messages based on their IDs and device types. Since, each device type targets a specific use case, such as a heart rate; hence it will hold different data semantics, therefore, this static module must have a specific message interpretation functionality, as in the current case that the company requires a set of specific sub-interpreters each for the heart rate, foot pod and temperature

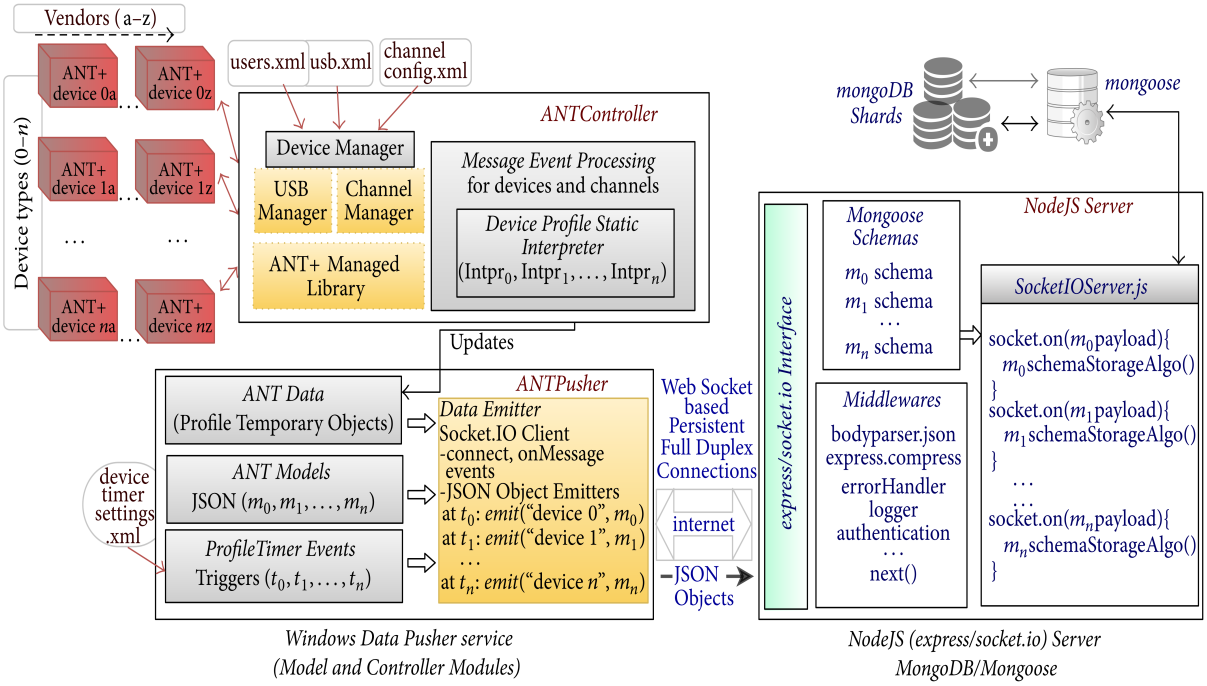


Figure 9.3: Methodology Architecture

sensor. For each individual device type, the message interpreter should be responsible for the extraction of the relevant data semantics and characteristics from the event message data; viewing the specific use case, whom details has been defined in the device profile documents provided by the ANT Alliance [359].

After data interpretation, the ANTController module extracts and stores each interpreter's specific data in the ANT Data objects, which hold attributes relevant to a corresponding device profile. For example, an heart rate object will have the attributes of its profile type. During processing of the messages an interpreter simply populates the extracted information into ANT Data objects, which are available for access any time to any service component. Thus the key is, the module's internal organization provides a separate ANT Data object for each device profile. Until now the *Window's Data Pusher Service*'s first part, related to the ANTController, has fulfilled; now the next is the description about the real-time data transmission, which is an essential feature of the communication middleware. Before going into more details, let us have very brief description about the functionality oriented interaction between the client and the server, as follows.

The data-driven scenarios are highly prolific in nature along a variety of data formats containing temporal properties, thus needs to be sent in real-time as a sequence of data items (sec. 9.3.4). It is also important to define a *window* or

slice on the data stream for the correct manipulation and management of data units in finite time (sec. 9.3.4). In this case, the data units will be the JSON objects holding the devices' data. An Event-Driven Architecture (EDA) is very flexible and scalable to extend a system for the development of publish-subscribe patterned applications; because these allow loose coupling between the client-server systems, (sec. 3.3.5). They provide lot of advantages in broadcasting and targeting a specific group of people to send the data; For example, one is sending an alert event to a patient's friends as well as to the doctors.

Therefore the Windows service contains the ANTPusher module, which transmits the data to the context server in real-time, after acquiring the sensor data and translating it into lightweight transferable JSON objects. The JSON formatted data is transmitted after managed sequence of time intervals using an EDA.

The ANTPusher component holds a general JSON modelling system, which additionally carry a separate JSON model w.r.t each device profile. Alongside the profile properties, a JSON model may have additional attributes, such as start time, end time, timestamps or identification. Furthermore, a single model can be subdivided into two models depending upon the requirements; such as all ANT+ sensors send their device information once after every 65 message events [133]. Therefore, we can differentiate this part of data from the specific JSON model and can send it separately upon another later time interval.

The sub-component DataEmitter uses a WebSocket protocol client to establish a permanent full-duplex, event-driven based, single communication connection with the remote distributed component after an HTTP-based handshake authentication, see sections 6.3.2 and 9.3.1. DataEmitter transmits the sensor data payload in the form of JSON message, as well as a device-specific *event*. Before the message sending starts, this emitter module populates the lightweight JSON model with the data taken from the corresponding ANT Data objects.

To handle the periodic streaming data of the sensors the software company decided to use discrete units of data elements in the form of JSON objects (sec. 9.3.1), which will be sent after defined periods of time using timmers. A timer plays the role of defining the size of the *window* or *slice* over the data stream, which is infact the time interval or duration after which the JSON data is transmitted. A specific timer generates events to trigger and notify the ANTPusher component to transmit the data after defined regular intervals. Each device profile can have a specific timer depending upon its message rate. A developer can specify the message rates in an XML file and provides those to the service component. Therefore, by following this method one can develop a sensor data acquisition service for any ANT+ device category.

9.5.2 NodeJS Context Server

The second main distributed component is *NodeJS Context Server*, which is a JavaScript runtime framework and is famous because of its vast list of add-in packages to build the big ecosystems (sec. 6.3.6). Its maintainability to support persistent connections, from the browser back to the server, make it distinguish from the rest of the programming languages, since those use a special thread for each new process, where as Node framework uses only single thread for the connection to deal all the clients. This component is responsible for the NoSQL based storage of the JSON sensor data payload which arrives in near real-time and emits from the ANTPusher component along a specific *event*. Since data-driven applications hold the properties of three V's, i.e. *Volume*, *Variety* and *Velocity* (sec. 7.4.1), therefore this component must provide flexibility and scalability to the sensor data storage, (sec. 9.4). To provide universal remote access to the clients, doctors and remote services, this remote context server component has an interface for the WebSockets with the characteristics of full-duplex persistent connection communication, (sec. 9.3.1). To have the WebSockets, the company decides to use the *Socket.IO* framework (sec. 9.3.3), because this is the best to send customized events (to target a set of clients) and uses an EDA, which facilitates the development of publish-subscribe applications by supporting loose coupling between the distributed components.

As stated previously that, the company needs a DBMS that must be flexible, in context to adding more software components both at a client and a server end (sec. 3.3.5, chap. 5); and scalable, in context to adding nodes to the system but only adding resources to a single node as this is in the case of horizontal scaling.The system must be flexible to provide storage to multiple different structural instances of a same device profile object, (sec. 1.4.3). For such a complex situation, the company decides that the best is to use a Document-Oriented Database, because of its schema flexible, scalable, and data handling features. There are many such non relational databases available; however mongoDB is considered as the best due to its flexible schema support, scalable, replication, and sharding features (sec. 7.4.2). For example, MongoDB can store different formats of the heart rate sensor data instance in a single collection and can provide efficient access to it.

To have a sanity layer before the schema-less data storage in the MongoDB, the company opts utilizing a pre schema approach which is very effective in context to rationality. For this, they use a middleware i.e, Mongoose; which is a fully developed Object-Relational Mapping (ORM) library for the NodeJS and MongoDB based environments, for details please refer to Section 7.5.3. This will support their programmers to develop MongoDB applications in a more flexible and rapid manner, hence the data will become more fluid and natural both for the programming language and the application itself. For a specific device profile, while

using Mongoose, a programmer can define as many schemas as he/she wants, according to the requirements. A programmer can validate, type-cast, map an object, build a query and build the business logic in a more flexible and proactive way; and can later use these features during the application development seamlessly.

The MongoDB allows the storage of data according to all defined schemas (i.e., using Mongoose) in any single (or more) collection, (sec. 9.4.1). However as discussed during the Section 9.3.4 that such data-driven environments needs stream processing capabilities to process and store the data in the desired format. The programmer wrote the device profile specific routines based upon Mongoose schema and processing logic. Next chapter Section 10.4, provides full details about the schema and the stream processing. Therefore, to store the JSON encoded sensor payload, based upon a Mongoose schema, this component contains a collection of device specific procedures which are responsible for the data manipulation and are also accessible through *Socket.IO* interface upon receiving an *event* from the Data Emitter. Such web resources (i.e., stream processing routines in this case) and event mapping (sec. 9.3.2,9.4.2).

These are set of programming algorithms responsible for a schema model based storage containing the temporal properties. To allow bidirectional event-driven communication, to have access to the data management procedures, to perform the tasks in a non-blocking way, this component must provide such a flexible loosely coupled services. Node.js is a JavaScript-based open-source event processing engine which performs operations in an asynchronous fashion (sec. 6.3.6). Due to its high-performance network communication and programming environment and the valuable features (sec. 9.3.1), the company can achieve the required functionality for the complex AAL environment to provide healthcare services to the elderly. The pervasive computing at the client side drive data to the *NodeJS Context Server* for the scalable and schema flexible storage. Next chapter will discuss how the schema flexibility will be achieved beside the customized stream processing of the sensor data for optimized storage.

9.6 Technical Implementation

An ANT network transmits two types of events i.e, ANT Device events and ANT Channel events, where the former is responsible for the processing of the *ANT_Response* object, containing the data details about a device event. Whereas the latter is responsible for the processing of the events which a channel generates, for example a channel for the heart rate sensor. The software company has to process all such kind of events, as they also contain the sensor payload data in their *ANT_Response* object. To have the control, message processing and transmitting features as a Windows service, they are inherited from the Windows

ServiceBase class. The technical details about the ANTController as a service are provided in Section 9.2 already in the same chapter. Similarly, ANTPusher (containing ANTEmitter) module is also inherited from the ServiceBase class. Hence the ANTController and ANTPusher are the two sub services of the *Windows Data Pusher Service* as a whole. Chapter 4 discusses the details about how to do Windows service development, along their advantages and disadvantages.

A prototype is developed and tested successfully, containing the above-discussed components and real-time flexible features. The three types of ANT+ device profile sensors are heart rate, temperature, and foot pod (sec. 8.1.2). The Device Manager module of ANTController component automatically loads the respective configuration settings from an XML file when the *Windows Data Pusher Service* starts. The XML file is provided in the Appendix .1.2. The system starts accepting data from the sensors simultaneously and parses the messages to populate the specific ANT Data objects.

The ANTController and ANTPusher service components are developed using the .Net framework 4.0/4.5 based upon the MVC architectural pattern. MVCs are flexible to add views, models or the control logic (ch. 5). A daemon service mostly does not have the display component because it runs as a background process. However, the prototype is developed with a target to have a flexible and integrated architecture which allow space for the additional features; therefore, there are different techniques available to add views as an additional feature to the system.

The pervasive monitoring applications are highly data intensive and produce volumes highly data intensive and produce volumes of sensor data streams, which need special measurements to send and receive data in continuous or discrete streams (see 9.3.4, 9.3.1). To send as a sequence of data items the prototype uses a total of four timers; which notify the DataEmitter module to emit the JSON data with device specific events. The three timers have been used to send sensor specific data periodically, so they are dependent on the data message rate of the devices; however, one timer is used to send device-specific information, for all the three sensor devices, but once after every 510 minutes (min). The reason for doing this is because the ANT+ sensor devices do not transmit device information so rapidly as compared to the other functional data pages (sec. 2.8), therefore transmitting/saving device information after a long interval is quite natural. The ANTPusher module fills the heart rate, temperature, and foot pod JSON models with the data payload and emits them upon receiving a triggering notification from the respective timers. Roughly speaking, almost each device's data is emitted after every 3-4 seconds (s); but these are not fixed configuration settings since the system is flexible and allows configuration changes to a programmer. For the transmission of heart rate data the DataEmitter module sends *'heartrateMin'* event and for the heart rate sensor's device information it emits *'heartrateHr' event*.

Different problems were been faced during the development of the prototype,

such as how to use WebSockets between a .Net platform containing *Windows ANT Data Pusher Service* and *NodeJS Context Server* server. To solve this issue, for the windows platform, the *SocketIO4Net.Client* library is used; which provides a .NET 4.0 C# client for the *Socket.IO* [254].

To accept WebSocket events and to utilize a full-duplex persistent connection in a non-blocking manner, the *NodeJS Context Server* is developed using Node.js framework (sec 6.3.6). For each expected event from the *Windows Data Pusher Service*, the *NodeJS Context Server* defines a listener written in JavaScript. For example, it defines `socket.on('heartrateMin', function (payload)){}` function, which listen for the `heartrateMin` event generated from the Windows service. In the listener function, the `function (payload)` is the callback function which executes only when the callee function finishes all its execution. In JavaScript asynchronous programming, a developer may get the return value in this parameter. As the listener function further may call any function, hence the current methodology suggests to call the corresponding storage routines (or the storage algorithms) to store or manage the data in MongoDB collections.

The server calls the corresponding algorithms, which is based upon the Mongoose schema, to check and store the data in MongoDB collections. Three different Mongoose schemas are defined for the three selected sensors representing three device profiles. For each device profile six different storage algorithms have written for their JSON data handling. The three of them relate to store devices' information in the corresponding collection. The communication middleware calls these respective procedures upon receiving the events through the *Socket.IO* interface. The storage routines consider the temporal factors and stores the data based upon the compact time series storage technique presented in Section 7.5.4. Chapter 7 also presents a case study to understand the advantages of normalization and denormalization for the storage approach in Section 7.5.4. The next chapter explains the MongoDB storage module using the Mongoose object mapping features based upon the stream oriented storage technique in more detail.

The distributed component uses ExpressJS (sec 6.3.6) framework for the MVC based Web development (ch. 5) to allow HTTP communication, especially handshake authentication to the Socket.IO base event communication (sec. 6.3.5). The context server also contain a stack of middleware processors which can be added of any quantity using the *npm*[1](sec. 6.3.6). These middleware components run against each request to perform general tasks, such as authentication, parsing, logging data or error handling, as depicted in the figure 9.3. Any one of these autonomous middlewares can change the request, alter data, and then pass it to the *next()* middleware in the chain.

[1] www.npmjs.com

9.7 Framework Design of the Product

This section presents the design of an architectural framework for the given problem to solve it, to support our work and the chosen approach. We shall use the concepts, defined and discussed in Chapter 1, Section 1.4.2. The framework will also be a guide for the future architects and developers who want to build real-time flexible and scalable applications. They may use this skeletal structure and its underlying artefacts to model or view their targeted architectures. The frame will keep their designs within the boundaries of the framework hypothesis space. The research question ***AAF2*** (sec. 1.6.1) identifies the tasks in the same directions, and is stated as: *What framework skeletal and its underlying artefacts, would support architecting a new real-time SHE to meet the current technical challenges?*

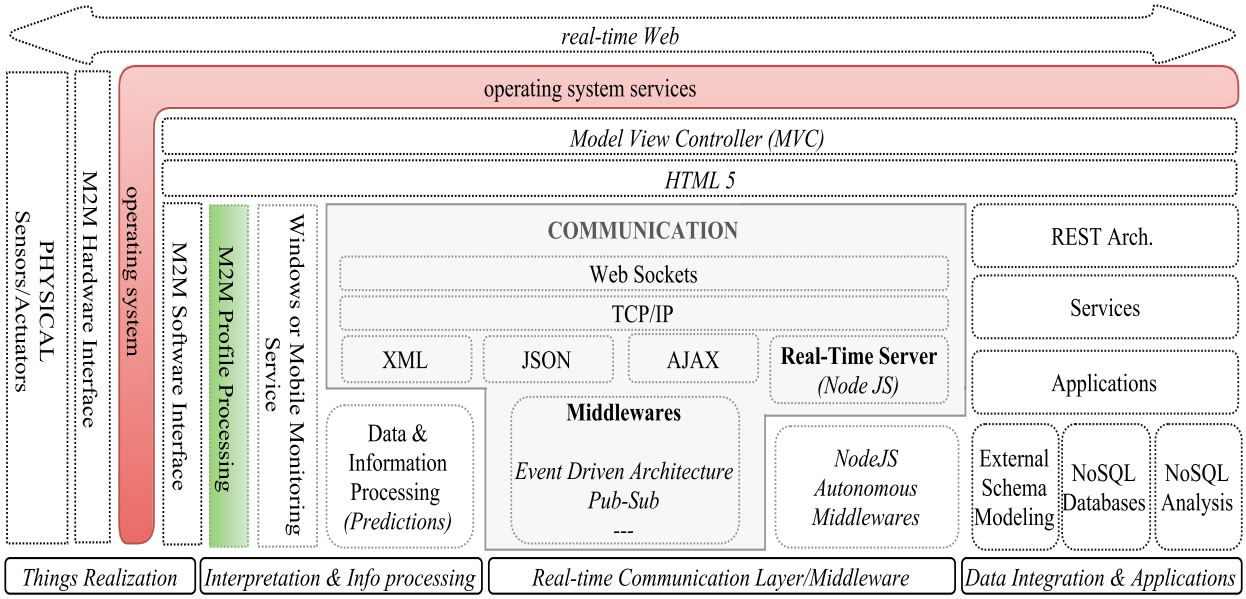


Figure 9.4: Abstract Architecture Framework

Our system is designed of modular components (Fig. 9.2, 9.3), to support reuse and ease of maintenance. We have already discussed the methodology and implementation views of the architecture (9.5, 9.6), here we shall discuss the functional abstract view of it using a layered approach. We shall highlight all the used artefacts, so far in the prototype; as well as other possibles artefacts of the real-time SHE architectures. The framework is shown in the Figure 9.4, and is consists upon four cooperating layers. Starting from the left, the *Things Realization* layer contains the hardware within the environment (8.1.2). This includes all the M2M

hardware components, including sensors, actuators, computers etc. The M2M software interface will allow the communication with the hardware over the OS platform (1.4.3). The *Interpretation and Information Processing* layer will consist upon the Windows Service (9.2) having data acquisition and message interpretation capabilities according to the M2M hardware profiles (2.8,8.1.2). Although not present in the current prototype, but data and information processing component is shown in the framework due to their importance to make predictions using mining algorithms used for decision making [352].

The *Real-Time Communication* layer will be available to all the layers, and this may have different artefacts as its core components. WebSockets (6.3.2) as a full-duplex permanent communication connection, EDA (3.3.5) as a middleware type, JSON as an interchange format and Node.Js (6.3.6) as a real-time server will be the core components of this layer. The system may also use XML and AJAX artefacts to build the communication middleware. An indispensable part of this layer will be the Node.js autonomous middlewares to provide distributed special services to the real-time server, such as parsing requests, authentication etc. The fourth is the *Data Integration and Application* layer responsible for many things ranging from the data storage to the development of distributed applications and remote services. The storage layer will use the NoSQL databases (7.4.2) for the temporal oriented (7.2) storage of sensor data using ORM mapping (7.5.3) techniques and data integration algorithms (Chap. 10). Whereas the development of the real-time distributed application and service provisioning will be possible by using different Web application and service development mechanisms, some are covered in 6.3.6. Such provisioning of applications and services will use NoSQL analysis tools (7.6) to access and analyse the sensor data in the NoSQL databases. Availability of resources through RESTful interface will make the distributed accessibility of the storage and service functionality.

An important artefact of this framework is to provide a design pattern that will overlap on all the layers and will be beneficial in developing SHE considering with the focus of separating the control from the underlying logic and interfaces, i.e. MVC design pattern (5). HTML5 is a significant component of this framework to design applications and services at any layer of the framework. The framework presented above contains the artefacts and design pattern practices, and gives thinking views to architects, by dividing the architectural description into layers to develop real-time flexible and scalable SHE applications.

9.8 Summary

This chapter solves the problem of the patients for a real-time scenario; and for this proposes a *Windows Service* approach which acquires ANT+ sensor data

within a BAN based smart environment. This interprets the sensor data simultaneously, to transmit it periodically to the context-data server in a non-blocking full-duplex event driven manner over the Internet. The communication middleware is based upon the *Socket.IO* framework which performs HTTP based handshake for authentication before establishing a permanent bi-directional connection. The remote context server is developed using Node.js framework which is highly efficient in context to resource utilization. The storage server accepts the lightweight JSON data and is capable of storing it in MongoDB in any structure in an asynchronous manner. MongoDB holds document-oriented capabilities to provide optimize, efficient, scalable and flexible storage. A pre-schema object mapping approach facilitates to build MongoDB based applications by bridging the gap between the storage repository and the programming language. The chapter gives details about the methodology user, technical architecture of the system; as well as an architectural framework guide for the future applications for the new architects fostering for real-time flexible and scalable applications.

Chapter 10

Modelling Temporal Aspects in NoSQL (MongoDB)

Time and space are not conditions of existance, time and space is a model for thinking.
–Albert Einstein

Monitoring scenarios transmit after time intervals and timestamp is important, so we should consider schema flexibility and temporal data integration altogether. This chapter explains how the *Node Server* processes the periodically arriving JSON messages with the help of custom storage routines. Such functions use Mongoose based schema models to map object to MongoDB. These routines get execution upon receiving events. An algorithm for the flexible evolution of the schema model for MongoDB is presented. Storage modelling approach considers temporal aspects and evolves during the data integration. The approach is beneficial in context to storage space and efficiency.

Monitoring applications produce different formats of data in abundance and transmit continuous data streams (sec. 7.2); which are difficult to manage for the traditionally RDBMS (sec. 7.3). During the chapter 7 we discussed that NoSQL databases are IoT databases and they can provide storage to any size and format of data, for which the relational storage mechanisms have given in, especially in the data-driven monitoring scenarios which transmit data in large *volume*, having *variety* of formats and with fast *velocity* (sec. 7.4.1). The new generation systems do not like to fix their database schema, rather they want multiple schemas for the same object type. They want their schema to evolve in parallel to an entity type's adaptation during data integration, hence they want flexibility (sec. 7.1).

The main reasons which help in selecting a good DBMS are: the data structure

or database schema, storage format, scalability, techniques of data gathering, and the custom real-time stream processing or third party management system. The company planed to promote these factors for flexibility and scalability; by utilizing the NoSQL approaches (sec. 7.4). Therefore, the company selects MongoDB, a document-oriented database, which instead of tables (as in RDBMS) provides one or more collection(s) as main storage components consisted upon similar or different JSON or BSON based documents or sub-documents (sec. 7.4.2).

The JavaScript based *Node Context Server* accepts the data and stores it in MongoDB by executing the ANT+ device profile related storage algorithm upon receiving the event notification from *Socket.IO* (sec. 9.5.2). Figure 10.1 depicts another view of the general real-time distributed system web architecture for the developed prototype during the second phase (chap. 9, Fig. 9.3).

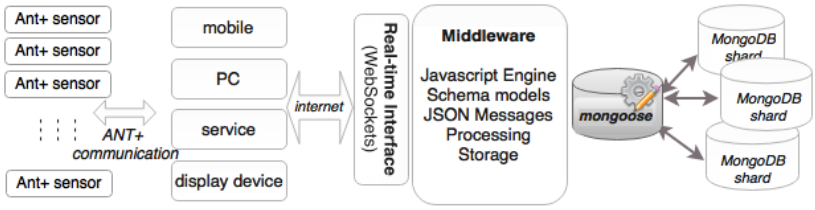


Figure 10.1: System, Mongoose and MongoDB relationship architecture WebSockets based Real-time interface with middlewares for JSON message processing, storage and evolution according to the schema model(s) defined in Node (JavaScript engine) using Mongoose

10.1 Schema Design

A data model represents the concepts of data structure required by a database (sec. 7.5). The NoSQL databases although provide schema flexibility and allows storage of different formats of same entity, but for the understandability a programmer has to follow some sort of format at-least at the application level (sec. 7.5). Multiple schemas are supported by document databases, if required each of them can be defined using the traditional modelling approaches, as discussed in Section 7.5.1. The techniques available for this purpose are entity identification, relationships, relationship types and ERDs (sec. 7.5.1). These lead to the natural grouping of attributes, i.e. relations. But relations are very much normalized, which is not the requirements in the hierarchical world of document-oriented databases (sec. 7.5.2). Denormalization is a feature of document databases, therefore the company needs to find a schema model that would allow the storage of the ANT+ sensors in MongoDB by keeping a balance between document embed-

ding or not-embedding. Such a schema designing rationality has been discussed in Section 7.5.2 and a case study is also provided in Section 7.5.4. For the object relational mapping and schema designing, the company selected to use *Mongoose*, as shown in the figures 10.1 and Section 9.3, which allows a programmer to define formats/schema out of the MongoDB, and uses object mapping technology to produce persistent storage against the program language object (sec. 7.5.3).

10.2 MongoDB Temporal Data Schema Design : ANT+ Healthcare Case Study

Time dimension is quite explicit in temporal Big Data and the problems of effectively and flexibly capturing and modelling temporal factors require further research orientations, thus modelling temporal aspects and schema evolutions is a well know issue to handle [117] (sec. 7.2). The company wants to define one or more MongoDB time series data models, that would allow the storage of real-time messages arriving from ANT+ enabled sensors (sec. 8.1.2). The next sections discuss the software engineering process of data modelling, in document oriented databases for time-series sensor data, in comparison with the RDBMS with respect to the healthcare use case.

10.2.1 Collections Identification v/s Entities and Entities Relationships Identification

The company decides to allocate a separate MongoDB *collection* to each sensor device category and defines its schema for JSON based storage using the Mongoose middleware. The storage server accepts two type of JSON messages: (i) per–minute timestamped holding sensor data; and (ii) per–hour timestamped message holding device specific information and some other healthcare data depending upon the use case. For the 3 sensors (sec. 8.1.2), they treat both message types differently during processing and storage depending upon the schema modelling.

What to model?

A big question to every one during document-oriented schema flexible modelling is what to model for their application, that also reside outside the database but gives a mirror to the program language objects to map for the persistency. The *ANT+ message protocol and usage* document guides us how to work with sensor nodes and to identify the device messages at an abstract level in general [140], but for more detailed understanding of the message data semantics related to each

sensor device type we have to follow the message format principles described in *ANT+ Device Profile* documents, such as *Environment* profile for temperature sensor [131], *Stride Based Speed and Distance Monitor* profile for footpod sensor [133] and *Heart Rate Monitor* profile for heartrate sensor [132]. Therefore if a profile indicates about transmitting a message it means that all the vendor devices will transmit it, hence they can be treated in general for all the sensors of that type. ANT+ devices transmit their monitoring data in the form of *data messages* which are specific to a profile type. Therefore if profile type can be considered as an entity then such data messages will be actually the attributes of it. Following descriptions expose the attributes of the chosen sensors.

Temperature: It transmits current, minimum (in previous 24 hours) and maximum (in previous 24 hours) temperature in Centigrade (C°). With each measurement it also transmits the transmission information, timestamp and the number of event counts - which increments with each measurement.

Footpod: A footpod measures speed, cadence, distance, duration, strides accumulated and distance accumulated in meters/second, strides/minute, meters, seconds, number of strides and meters as measurement units respectively.

Heart Rate Monitor It measures and transmits the heart rate in beats per minutes (bpm), with heart beat variability, event count and timestamp.

There are messages which most devices transmit but those are not specific to a profile standard. Such common messages are called as *common data pages* or *background messages* [359]. For example devices transmit their information such as battery status, message rate, manufacturer or hardware information etc.

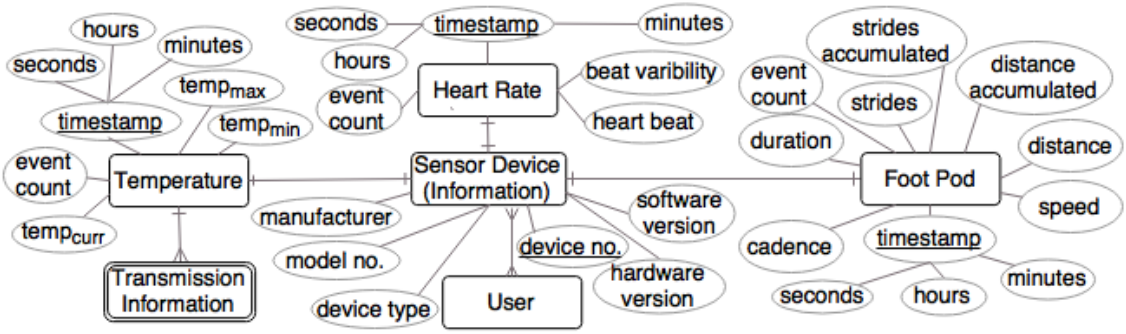


Figure 10.2: Entities and their attributes with respect to the discusssed use case

The three sensors are actually the instances of entity device, but should be treated separately because they do not share attributes; and also their data, background messages and timestamp values may be different to each other, as dis-

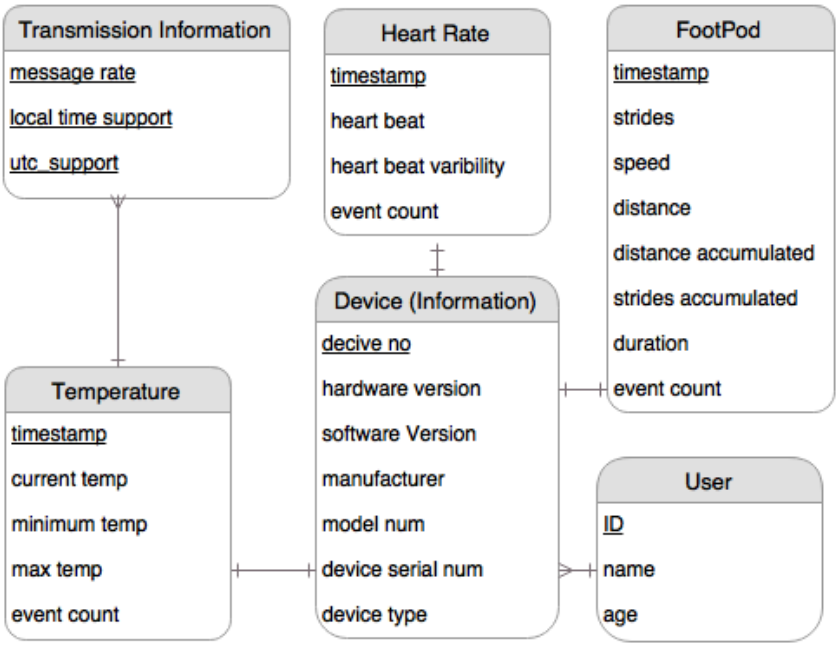


Figure 10.3: Data model with entities and their attributes for the use case

cussed above. However all ANT+ sensors may have same data structure regarding the device information, therefore it can be treated as a separate entity with reference to the other sensors. It is to note that a sensor will always have a unique serial number, manufacturer id and model number, although the devices can share manufacturer ids, device types etc. A user can use more than one sensor at a time. Based upon these guidelines an entity relationship diagram is depicted in figure 10.2 showing the possible attributes, relationships and the primary keys; then a relational model has been drawn in figure 10.3 in 1 Normal Form (1NF) [146].

Discussion w.r.t RDBMS: A relational model although supports sensor data storage but it has many flaws, such as many joins are required while querying any sensor's data for a patient at a particular time. The model is not in 2nd and 3rd Normal Form, so there are dependencies among the attributes too. There will be many null values because sensors will not transmit all the attribute values altogether. To resolve this issue the final result will have only three columns in each of the sensor table i.e. *SensorTable (timestamp, value, event count)*. It is almost same as saving every event message, but this will not be robust both for SQL as well as for NoSQL because there will be insertion and update issues too; beside the computation processing issues, such as averages, aggregates etc.

Discussion w.r.t MongoDB: Considering each entity as a possible document collection will require joins between them, which document-oriented databases

do not support natively. However this is possible, either within the application or by using data processing analysis tools such as aggregation framework [7], MapReduce [122, 258], Hadoop [303, 190] etc. Document-oriented databases do not have problems with the null values because of schema flexibility, however interestingly, in such a case the final result will be the same as having every event storage in a separate document structure, which is not efficient in context to processing and storage (sec.7.5.1. Therefore, the company decided to seek the advantages of hierarchical document structures.

Timestamped based cardinalities

A message rate is measured in cycles per second and the unit for frequency is Hertz (Hz). The data messages of the three device profiles with respect to their allowable message rates are discussed below. The discussion calculates that how many instances of a certain sensor *data messages* will appear against a time value, such as seconds, minutes and hours; and Table 10.1 summarizes the discussion.

Table 10.1: Data messages cardinalities with respect to allowable message rates

device profile	message rate	seconds	minutes	hours
Temperature	0.5 Hz	0	30	1,800
	4.0 Hz	4	240	14,400
Foot Pod	2.01 Hz	2	120	7,236
	4.03 Hz	4	241	14,508
Heart Rate	1.02 Hz	1	61	3,672
	2.03 Hz	2	121	7,308
	4.06 Hz	4	244	14,616

Temperature: This environmental device broadcasts only at minimum 0.5 Hz or the maximum 4.0 Hz message periods therefore once per 2 seconds or once per 0.25 seconds respectively.

Footpod: A device can receive data at the full rate (4.03 Hz) or at half of this rate, therefore once per 0.248 seconds or once per 0.498 seconds respectively.

Heart Rate Monitor: It transmits at the full rate (4.06 Hz) or at one half or one quarter of this rate; therefore the data can be received four times per second, twice per second, or once per second.

Common messages or *background messages* appear after 64 *data messages* for each profile type [359]. In context to the time it depends upon the message rate, for example, heartrate with a message rate of 2.03 Hz will transmit almost 2 messages per second, therefore in this case a *background message* will appear

after 30 seconds approximately; whereas with the minimum message rate (i.e., 1.02 Hz) the server will never receive two background messages in a same minute. This may not required to store every received message, hence it is better to store it either after every 10, 15 or 20 minutes or only once per hour.

How good is the model? Normalization or Embedding

Denormalization is a key feature of document databases and supports efficient data retrieval and storage optimization. Designing a better model is always the main goal, and their is always a trade off between normalization and denormalization during the modelling process (sec. 7.5.2). The Section 7.5.4 presents three different approaches with examples to model time-series data in MongoDB, i.e. document-per-event, document-per-minute and document-per-hour. All these three approaches are different in context to document embedding, which results in different storage space and number of queries, as discussed in Section 7.5.4.

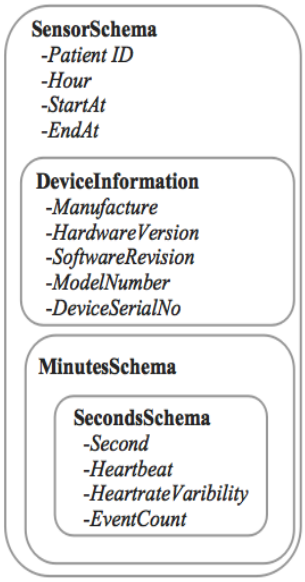
Because data stream management systems (DSMS) are prune to the arrival rate and need adjustments upon any change in the rate. For this prototype the least message rates of the sensors are used for the simplification purposes. The company used the **doc-per-hour** approach with the nested minute documents carrying 0-59 second documents. This approach requires one insert initially for the hour, and then only updates are required for each new second. To update the last second it requires maximum 59+59 steps (sec. 7.5.4). Next section presents the heart rate Mongoose model, which is based upon the doc-per-hour schema approach.

10.3 Heart Rate : Mongoose Schema (Doc-Per-Hour)

Since the above three version of the embedded document storage differentiate the data into hour, minute and seconds, therefore before going to present the mongoose schema, the first step is to separate the hourly, minute and second level data from the heart rate sensor payload. As stated already, *background messages* mostly relate with the hardware information of the sensor device itself, and apear only after 65 other data messages (sec. 10.2.1, 2.8) [140], therefore keeping them at the hourly level would be fine for the prototype. These messages include data related to hardware version, software version, manufacturer, model number and device number, as depicted in the use case ERD and relationship model diagrams 10.2. Beside this since hour-level document will be a separate document therefore it will also keep a reference to the end user, as well as the start and end timestamps.

The minutes' document will keep seconds' sub-document inside it. Each second's sub-document will have the heart rate data, i.e heart beat, variability and event count. The ERD in figure 10.2 and data model in figure 10.3 depict

these attributes. The Figure 10.4(a) shows an in general denormalized model for the heart rate sensor, in which the main SchemaSensor (for hours) contains the SchemaMinute (for minutes), which further contains SchemaSecond as the second's sub-document schema. Alongside the Figure 10.4(b) shows an in general Mongoose schema pseudocode for the denormalized schema. We can observe that most of the values will be at the seconds level, i.e, Heart beat, heart rate variability and event count as the measurements.



(a) Denormalized Schema

```
var SchemaSecond = new mongoose.Schema({
                second : Number,
             heartbeat : Number,
          hrvaribility : Number,
            eventCount : Number});

var SchemaMinute = new mongoose.Schema({
    minute:Number, seconds:SchemaSecond});

var SchemaSensor = new mongoose.Schema({
    patientID:String,
    deviceInfo:{ manufacture:String,
                 hardwareRevision:String,
                 softwareRevision:String,
                 modelnum:String,
                 deviceserial:String },
       hour : Date,
    startAt : Date, endedAt:Date,
    minutes : SchemaMinute
});
```

(b) Mongoose Schema Code

Figure 10.4: The denormalized schema and the Mongoose schema for HRM

10.4 Mongoose Schema Base Temporal Data Storage and Evolution

The company to define a finite set of sequenced steps, to store a JSON message instance related to an ANT+ profile type. Such steps must perform logically correct updates to a database, leaving it consistent and concurrent after the manipulations. It is significant to mention that, the storage algorithms should be capable enough to hold all the possible schema, and such schema oriented algorithm must be valid and agreeing to a delivered/selected specific profile type. Following is the description of the storage algorithm which accepts JSON based sensor data messages to store and evolve the chosen schema model for MongoDB. This is based upon Mongoose, which is the middleware used to define time-series schema for rationality and demonstrating the storage structure. The algorithm will use the object

of the Mongoose schema model to perform the inserts and the updates to the document database, and such manipulations must be performed in a manner that the new data integration would result into a storage formation as defined and desired in the model. For the above stated Mongoose schema i.e,SchemaSensor ,the set of tasks which the *Node Context Server* will execute once for each new specific sensor event, for the JSON formatted sensor payload, is provided in the flowchart depicted in the Figure 10.5. This flowchart shows the logic of the algorithm and emphasises the steps and their interactions in a controlled manner from one action to another. This also depicts how the data flow will take place within the Doc-Per-Hour Schema based upon the Mongoose object mapping middleware. The Figure 10.5 is followed with JavaScript code listing, which is the implementation of the data integration and schema evolver algorithm defined in the flowchart.

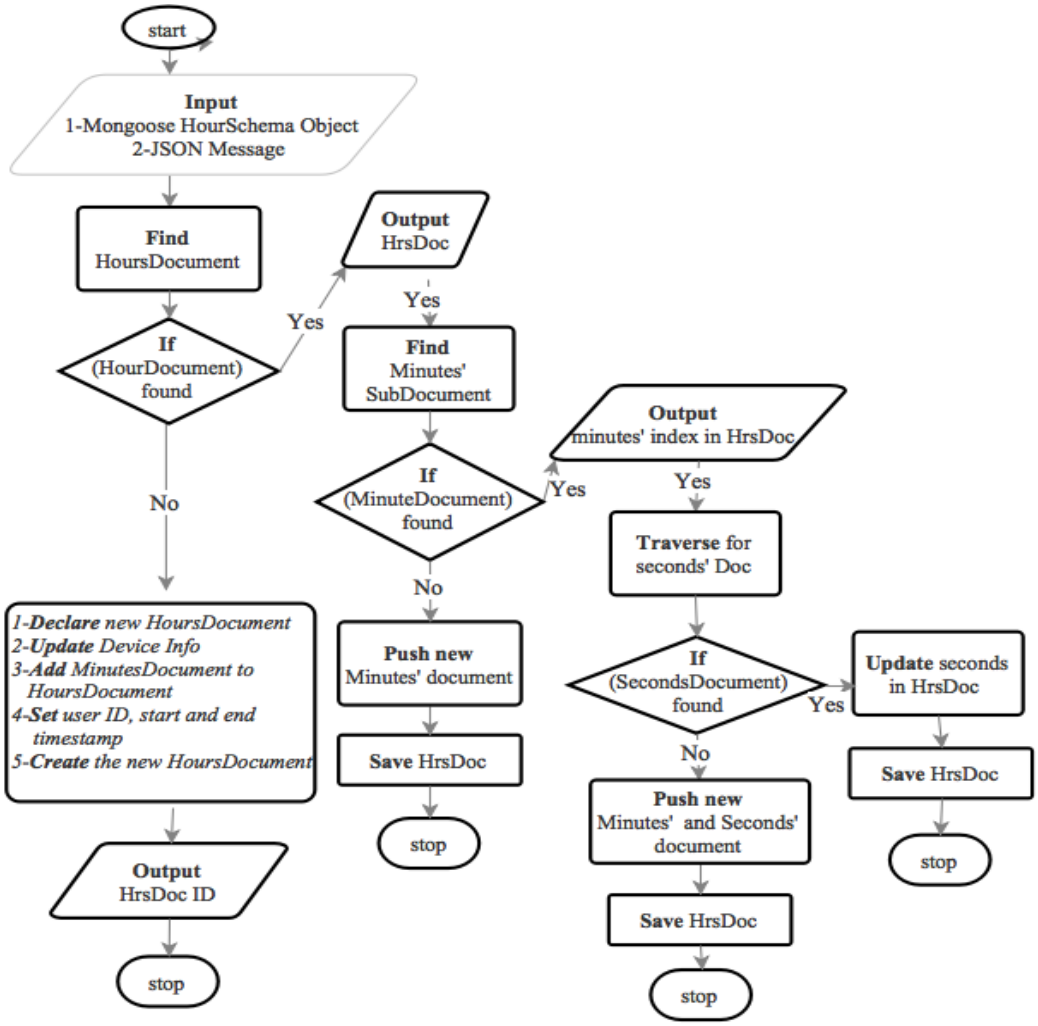


Figure 10.5: The flowchart of the Mongoose based schema evolver algorithm

```
1 require 'mongoose'
2 SchemaMongoose = require('sensorschema.js')
3 function SensorMinutes(reqObj)
4     args = parse reqObj for JSON
5     SchemaMongoose.findOne(query4Hour, hourDoc)
6     if (hourDoc){
7               SchemaMongoose.findOne( query4Minutes, minuteHRM)
8               if (minuteDoc)
9                   traverse for second's document
10                  if secondDoc does not exist add it, otherwise return
11                  else if second exist
12                        secdocs.push(sensor second data)
13                        save hourDoc and exit
14              else
15                  get hourHRM->minutes as array in minuteArray
16                  minuteArray.push( sensor minutes data)
17                  hourDoc.markModified('minutes')
18                  save hourDoc and exit
19   else //if hour-doc does not exist create new schema document
20            newHourDoc = new HRMMongoose();
21            if (args is null)
22                newHourDoc.device = null
23            else
24                newHourDoc.device = {
25                hwrevision  : args.HWRevision, swrevision:args.SWRevision,
26                manufacture : args.ManufacID,  modelnum  :args.ModelNum,
27                deviceserial: args.DeviceSerial }}
28       newHourDoc.minutes = {minute: minNum,
29                seconds  : [{secnd : datetime.seconds(), sensor seconds data
30         set patient and time info in newHourDoc
31         SchemaMongoose.create(newHourDoc, function (err){}
```

The sensor schema in JavaScript will be imported first and will initialize the main schema variable i.e., SchemaMongoose (line 1-2). The SchemaMongoose object will be the mapper between the language constructs and the persistency of data in the MongoDB, and will help also in evolving the database according to the desired temporal schema logic. It first queries the database for the hours-level document based upon the time-interval provided in the request object (i.e, reqObj in line 5). The Section 7.4.2 provides very brief information about MongoDB queries, which are different than the traditional SQL/PL-SQL environment. In the same manner the query4Hour (in line 5) will have the query parameter and the projection parameter. For example the following code constructs the query parameter to fetch the document where the hrmHour is equal to dateHours and patient is userid.

```
var userid = store.get(`curruser');
var query ={ "hrmhour" : dateHours, "patient" : userid};
```

In Asynchronous JavaScript programming the functions return the results in the last argument i.e., the callback; therefore the result will be returned in the hourDoc object. The listing is self explanatory and is having correspondence with the flowchart. If the hourDoc is found in line 6, the algorithm will go on searching for the minutes document, as in line 7; otherwise will move to line 19 for a new declaration and creation of the hours document.

10.4.1 Issues

In MongoDB schemas are sometimes prune to update problems related with the deep nested arrays of JSON objects. For example, the "SchemaSecond" dimension contains deep nested arrays of JSON objects (fig. 10.4). When this is periodically updated upon receiving values, the algorithm evolution sometimes result in an error. The author tried to investigate the problem and found that this is because of the limits in the MongoDB engine in context to updating the deep nested schemas having arrays. It does not support several positional operator (i.e, "$") based updates to the nested arrays[1]. The problem is discussed in an online thread[2], and has solutions in either to wait for a fix or to avoid deep nested schemas.

10.5 Summary

This chapter presents the usage of Mongoose middleware to represent document oriented hierarchical schemas for the temporal modelling of the ANT+ sensor data. The Node Context Server uses these schemas to run a set of ANT device profile related specific tasks to be executed upon specific event related to the profile. The algorithm is executed in a manner that it automatically builds upon the hierarchical structure of the sensor data based upon the temporal properties. There are many possible schema variants, however this chapter discusses the usage of three main types of temporal modelling document schemas, i.e. (i) document-per-event, (ii) document-per-minute, and (iii) document-per-hour. The normalization and denormalization of the document hierarchy decides the quality of a schema with respect to number of reads, updates and storage space utilization.

[1] https://jira.mongodb.org/browse/SERVER-831

[2] stackoverflow.com/questions/14855246/multiple-use-of-the-positional-operator-to-update-nested-arrays

Part V

Summary and Conclusion

Chapter 11

Summary and Conclusion

Although the role of Information and Communication Technology (ICT) is in almost every field, but healthcare sector demands a special attention due to a significant shift in the elderly population. Almost every individual society member want to be fit and healthy by optimizing his daily routine and body life parameters. The cardinal is to improve the quality of life (QoL) of the society, by being Keep In Touch (KIT) with them. RHCM mechanisms are there to build smart environments to meet such new healthcare and fitness challenges. Such smart environments use sensory objects and have distributed components, which are integrated with each other to provide pervasive features. Internet of Things (IoT) supports the development of such environments though providing Internet as a platform and promoting interoperability among the sensor devices. IoT supports for interoperability is also through enabling M2M communication, to build big ecosystems. Whereas the goals of the distributed system are to build autonomous components which work and cooperate in a manner, that any end user may feel the whole system as a single component during its usage at any end. For this they use middlewares, to integrate information and heterogeneous components.

Abundant use of sensor objects in HCM applications is almost mandatory, but they transmit large number of messages continuously. HCM environments are data-driven, where the data defines the control flow of the system; and generate volumes of different formats of control and textual data in a sequential manner. This large *volumes* of data has a *varieties* of formats and being transmitted with fast *velocities*, that the traditional data acquisition, processing, transmission and storage mechanisms has given in their support to develop next generation SHE based systems. Hence to meet these new challenges robust, real-time, flexible and scalable methods, architectures and frameworks are required the required goals. To achieve these goals, different tasks were carried out related to each identified research domain.

This Ph.D. research work contains a study of SHE and development of a healthcare prototype, which was made using a M2M interoperable technology (i.e., ANT+) for a real-time remote healthcare and fitness use case. To explore further requirements, the research work uses prototyping as a pragmatic technique and builds a smart real-time distributed environment. This pragmatic research was with the aims of sketching an architecture, deriving a methodology guide and designing an architectural framework skeletal, which reflects the properties of a HCM SHE within a distributed WSN environment having pervasive characteristics. Along side, during the research work we investigated the temporal data modelling and storage of real-time sensor data. The developed distributed system presents an *architecture* and its *methodology* which uses the state of the art new generation technologies, such as middleware, real-time Web, IoT and Big Data.

A prototype was developed having flexible and scalable architecture characteristics as well as real-time communication that proves to be reliable in the transmission and storage of healthcare measurement data. The architecture has three main components, i.e- (i) *Windows Daemon Service* – This is remotely controllable and autonomously startable service to make the system operational for work, as for this no login is required. This automatically configures the M2M ANT+ sensors using XML encoded settings and builds an ANT+ based BAN for the real-time data acquisition and device profile based messages interpretation. Such a message interpreter is unique and parses the semantics of all the sensors of a particular device type, such as temperature, heart rate etc. This first stores the sensor data into temporary objects and then later transforms the data into light weight JSON objects to transmit them to the remote server in real-time for schema flexible temporal storage; (ii) *Middleware* – This is to bind the remote heterogeneous distributed components using their universal interfaces, as this involves communication between the remote components. For the prototype an EDA based bi-directional and permanently connected middleware component was developed for the real-time exchange of different formats of data using WebSockets; and (iv) *Context Data Server* – This is an efficient Node.js, which contains a JavaScript engine, based real-time server to support the WebSocket clients. This also uses many other multiple frameworks to help develop and run complex systems. This can provide universal interface to access its resources. The *Context Data Server* takes the advantage of the NoSQL databases to store the continuously arriving JSON formatted temporal data messages in a schema flexible format. This allows storage to an ANT+ sensor device profile type within a document oriented database. This storage component possesses customized storage routines, based upon an object mapping middleware (i.e., mongoose), which allows the temporal oriented storage of the ANT+ sensor data into MongoDB database. These databases support multiple schemas to store data in a hierarchy of documents. MongoDB is highly scalable and efficient to handle volumes of different kinds of data. The mongoose

based schema model is used during the prototype, which preserves the ANT+ sensor data according to its temporal relationships. The schema is efficient in context to storage, and supports the fast processing of the data.

The aims also includes a methodology description explaining the principles and practices for the real-time working of the prototype which allows the addition of vendor neutral ANT+ sensors. The methodology completely describes the phases, that how to acquire, process, interpret, represent in a format, transmit and store the sensor data. Both the remote components are developed using MVC design pattern. MVC based applications differentiate the application control and business logic and the interaction components from each other; hence facilitating the developers to concentrate on one thing at a time. This also allows the development and integration of multiple personalized views into the system. Beside this, the thesis includes depiction of an architectural framework as a guide to think and plan real-time architectures for new similar monitoring applications. The artefacts of this framework stack are classified and depicted in four layers. Each layer works as an input-output for the next layer. The four layers are: (i) things realization; (ii) interpretation and information processing; (iii) real-time communication; and (iv) data integration and application layers.

11.1 Research Questions and Hypothesis

In the first introductory chapter in section 1.6.1 we stated few research questions, to laydown a base for this research work. Restating and discussing the research questions and revisiting the background along the research contributions, in context to pragmatic shaking of the established visionary hypothesis, would definitely provide a wholistic view towards the pursuance and perception of their answers for the extension of the knowledge boundaries. Following we discuss the research questions and the hypothesis with respect to the both prototypes, in a sequence.

11.1.1 Research Questions: Healthcare Monitoring System

Healthcare monitoring systems possess smart environments with pervasive capabilities, which is a field of Gerontechnology (sec. 1.3), to provide remote healthcare means to the patients and elderly society (chp. 2) for their QoL. The purpose of a monitoring system is to provide mechanisms to be Keep-In-Touch (KIT) with the patient by monitoring his environment using the advanced ICT means. This is with the goals to develop a system and its architecture that allow the provision of the care services. Following research questions were arose during the preamble.

HC1 What kind of Smart Objects (SO) or wireless sensors would be appropriate for a given AAL problem?

HC2 How to communicate with the sensor devices?

HC3 How to integrate the sensors to build a monitoring network?

HC4 How to process and interpret different sensor messages?

HC5 What components are required to develop a basic healthcare system?

HC6 How to acquire, represent, transmit and store any kind of sensor data?

For the research questions HC1 and HC2, we discussed the sensor devices along their communication protocols in section 2.7, and read that they are the building blocks for the development of WSNs, which are the fundamental components in the monitoring pervasive environments (sec. 2.2.1). But these fields are facing many issues (sec. 2.7.3) like less resources in context to memory, processing and energy (sec. 2.7.4). Therefore the Smart Object's selection is highly important factor (sec. 2.2) and depends upon the problem domain, like we have discussed and selected for a use case in sec. 8.1.2. Since HCM applications are applied for long durations, so the factors which affect upon the selection of a wireless technology are service features, time duration, range, battery power and communication protocols. We found that ANT+ (2.8) is better in context to M2M based interoperability (1.4.1), long time battery with low energy consumption (tab. 2.2) and service features (fig. 2.6). During the first and second prototype development, to solve the healthcare problem of a patient, a total of three wireless sensors (sec. 8.1.2) were selected and made the communication enable (sec. 8.2.1) using different relevant libraries and gateways (sec. 2.8.4). These practical tasks and their outcome have helped us to validate and prove the following hypothesis 1 (1.6.2).

Hypothesis 1 Interoperable and energy efficient Smart Objects (SO) technology - allowing easy device portability, identification, integration and communication to build WSNs, with different topologies - will support the development of a pervasive HCM system for a patient.

The question HC3 deals with the sensor technology to be applied in an integrated manner. The ANT+ protocol enabled sensor technology (sec. 2.8) is easy to integrate because of its M2M based interoperability (sec. 1.4.1). After integration the developers can build different type of WSN topologies (sec. 2.7) depending upon the application domains. For a PAN topology a simple wireless network is realized using the ANT+ sensors in the prototypes (fig. 8.2). For the question HC4 the thesis suggests to have a unique static message processing and interpretation module (sec. 8.2.2) that works in real-time to process all the different profile categories of a sensor technology (sec. 2.8.1). However such a module must have an interpreter for each profile, such as an interpreter for all heart rate

sensors. Both the prototypes present such a unique message interpreter (sec. 8.2.2, 9.3). So these are valid evidences to approval of the following hypothesis 2.

Hypothesis 2 A general approach towards the device's message processing not only simplifies the data interpretation, but is also mandatory to have a specific unique module for this purpose.

For the question HC5 the very basic modules, which are considered fundamental for a healthcare monitoring system, are depicted in Figure 8.4.

The research question HC6, mainly deals with the distribution of data to a remote location, and involves the data flow phases, such as acquire, transmit and store (sec. 9.1). Beside this the question also points out the representation of different types of data and its transmission to a distributed location for the permanent storage. Therefore this question is a very general and has a broad perspective. The thesis deals this question by presenting two prototypes. The first prototype presents a static approach (sec. 8.2.4), with the use of ANT+ FIT [364] encoding for the compact data representation and storage. This is an inefficient and inflexible approach, as discussed in section 8.3.2. Sharing data with the server in encoded format rise many issues in context to the platform compatibility and data decoding at the server side (sec. 2.8.1). The thesis approves hypothesis 3 for the first prototype, but concludes that FIT based data sharing may be interoperable but is less efficient. It demands a mechanism that understands formats for both client and server communication. Similarly, during the second prototype, we have learned that real-time processing is an efficient approach (sec. 9.3). For this the system uses WebSockets and EDA, section 9.3.3. The *Windows Data Pusher Service*, that contains the ANTController and different message interpreter component (sec. 9.5.1), uses a communication middleware which processes the different formats of data and transmits it to the *NodeJS Context Server* in JSON format. Hence the second prototype also validates the following hypothesis.

Hypothesis 3 A software that understands different data formats at the client or server side may solve the transmission problem of different data structures using wrapping and transformation technologies.

This research question HC6, also points towards the storage of the different formats of data. This reflects the one dimension (i.e, *V*'s) of Big Data management frameworks i.e., *Variety* of data. During the first development of the prototype, the thesis depicts that it is possible to store different data formats in relational databases but there are many drawbacks and the system do not remain flexible and scalable (sec. 8.3.2). The thesis evaluates the hypothesis 4 and concludes that relational approaches are not sufficient to process the data-driven scenarios, such as sensor based monitoring applications. We shall revisit the same problem during the discussion of research questions related to the new generation systems.

Hypothesis 4 Relational Database Management Systems (RDMS) supports any sensor data format storage in any scenario.

11.1.2 Research Questions: Distributed Systems

AAL applications have distributed system characteristics (sec. 2.4.4), as they span upon multiple nodes holding own memory and autonomous computational logic components. The distributed components are remotely accessible through their standard interfaces to communicate and cooperate, so that they appear as a single coherent system to all the users (sec. 1.4.1). A healthcare system is developed during this research and following is the investigation of the related research questions. Following discussion will check and validate the established hypothesis, which provided the base to this research.

DS1 What are the distributed system characteristics and how to have those in an AAL system?

DS2 What data flow phases are involved in a distributed AAL system?

DS3 Given the application domain, what kind of middleware will best integrate the different components of the system and how it will handle the flow of sensor or control data within those distributed components?

DS4 What communication means would be useful for the sensor data sharing with the context data server?

The DS1 demands that the prototypes' architectures must obtain the characteristics of a distributed system. By checking the second prototype particularly for such characteristics we observe that the distributed components have autonomous computational capabilities with own separate memory (sec. 9.1); and they are integrated to each other in a way that they appear as a single system to the users.

The question DS2, asks for the sensor data management ranging from its acquisition until its storage. First prototype handles the data acquisition, interpretation, processing and encoding into FIT format (sec. 8.2.2, 8.2.4). Then to transmit those compressed files to a distributed database server for storage, it uses a batch processing transmission technique (sec. 8.2.4. Whereas the second prototype is efficient and transmits the sequential data in real-time after transforming it into JSON objects (sec. 9.3). The remote NodeJS Context server processes the JSON data for its temporal based document oriented storage (sec. 9.4).

In context to the question DS3, the middleware in the first prototype allows the integration of the components for a specific period (during batch processing) (sec. 8.2.4), otherwise the autonomous components remain separate. Whereas the

communication middleware in the second prototype is more efficient and helps in building a pervasive environment in which the different heterogeneous components get integration for real-time communication (sec. 9.3); and cooperate using events within an EDA (3.3.5). The *Node Context Server* provides universally accessible interface to any client, such as the *ANT+ Data Pusher Service*. The server is available for access to any remote distributed resource, including mobile phone, web service, Web application or a daemon service as in our case. This communication is established using Socket.IO frameworks, which is based upon WebSockets, to establish full-duplex, permanent communication (sec. 9.3.1) of asynchronous JSON objects in real-time (fig. 9.3). Hence the previous discussion provides an evidence that established hypothesis 5 is valid, which is as follows:

Hypothesis 5 In the domain of distributed healthcare, a carefully designed adhesive middleware holding components that overlap upon different system nodes. Its each processing stage may provide useful abstractions to developed components, for the implementation of applications that can access and share information and services from heterogeneous domains.

11.1.3 Research Questions: New Generation Technologies

The abundance of smart devices, along their wireless and identification capabilities, have urged the ICT researchers to expand the Internet into the lives of the individuals (1.4.3). The availability of the autonomous systems, through the universally accessible interfaces, has allowed the remote systems to come close to each other, and the communication middlewares have bridged the gap for the heterogeneity reconciliation (sec. 1.4.1, ch. 3). HCM applications in AAL domain are data intensive and produce abundance of continuous data streams, which require robust real-time transmission (ch. 6) and storage mechanisms (ch. 7). Following highlighted research questions, demands same like new generation solutions. Now we discuss, how the research provided answers to the following research question, and how hypothesis helped to look forward. We shall validate the hypothesis based on the research work.

NG1: How IoT helps us to deal with device heterogeneity?

NG2: The data-driven AAL monitoring applications produce Volumes of data, then how to manage such prolific data?

NG3: Given the new trends how we can provide a general storage to any device?

NG4: How to handle timestamp streaming data coming from monitoring devices?

NG5: How a client or server be notified about a significant change in system state; and how stakeholders and different components will receive events?

NG6: How to provide communication for user critical services in real-time?

For the NG1 research question we learned that M2M communication solves the heterogeneity between the devices through standardization (sec. 1.4.1, 2.8.1), for example ANT+ (sec. 8.1.2) allows easy integration into its network given the access credentials. The questions NG2 and NG4 demand robust solutions for the data-driven environments which produce *Volumes* of data *Velocities* (sec. 7.4.1). For these two V's of Big Data, the second prototype provides the solutions in the form of *Windows Data Pusher Service*, which transmits sequence of JSON data messages, then on the other side the *Node Context Data Server* also have middleware based temporal storage mechanisms (fig. 9.3). The question NG3 deals with the different formats of the data and demands polymorphisms with in the databases. Hence for such an unstructured data we have solutions in the form of NoSQL databases (sec. 7.4.2). The usage of MongoDB (sec. 7.4.2), as a document-oriented database provides storage of JSON formatted documents along their embedded or hierarchical features.

The critical services in healthcare, like fire alarm, always need open permanent connections; which must work in both the directions (sec. 9.3.1). AAL systems generate urgent alert events, so they need rapid transmission (sec. 9.3.4). The prototype not only provide creation and transmission of such events, rather also provide an EDA for the mutual cooperation of processes through event notifications. Such functionalities are the requirements of NG6 and NG5 questions respectively. The obtained solutions guarantees the validity of the following hypothesis 6.

Hypothesis 6 Internet of Things (IoT), Big Data and Real-Time Web are new emerging technologies and are useful in variety of purposes, such as device integration, Event-Driven architectures, different data structure processing and schema flexible storage of high volumes of time-series data.

11.1.4 Research Questions: Architecture and Architecture Framework

An architecture is the representation of the high level system components and their cooperation; and explains the over all system working (1.4.1). One may use modular, layer, central or distributed architecture patterns to sketch SHE (1.4.1). Designing flexible and scalable architectures is an issue, and frameworks help the architects to think and build using a set of artefacts (1.4.2). Such an architecture

and guiding framework are the requirements of the following questions. Let's check the answers from our product and lets see how useful were the hypothesis during modeling and frame building.

AAF1: How the common technical artefacts can be represented as an abstract software architecture for a real-time HCM system?

AAF2 What framework skeletal and its underlying artefacts, would support architecting a new real-time SHE to meet the current technical challenges?

The first research question asks to find the artefacts and then to depict those as a picture showing their relationships. The artefacts of the developed prototype are its components, ranging from the data acquisition to the storage of the data. Figure 9.1 shows the two basic components interacting with each other through a middleware. Using a modular, layer and distributed architecture pattern a detailed figure 9.3 depicts the main architecture of the system with a real-time communication focus. Similarly, figure 10.1 shows the high-level logical organization of the system with a focus upon NoSQL storage features.

To answer the question the hypothesis is to identifying the key design elements with respect to significant layers. The framework design of the product 9.7 focuses upon four main layers, i.e. hardware, hardware communication, data interpretation, communication and storage. Each such layer has sub-elements as shown in the figure 9.4. Software architects can use this frame to design their architecture, for real-time SHE, for a real-time monitoring scenario. Looking upon the results we conclude the validity of the hypothesis.

Hypothesis 7 Try to cover the scope of the system by organizing the elements of the system and showing their relationships. Try to break the system into smaller sub-systems, according to physical and logical characteristics.

Hypothesis 8 Selecting the key artefacts and depicting them as a layered stack may support the design of a conceptual skeleton as a reference framework.

11.2 Anticipating HL7 Standards

Without any irregularity or exception, most of the application servers define their own specific data formats, hence they do not use an official standard, like HL7, for the health data exchange [167]. A reason for this trend is the customization, authorization policies, user's data privacy and trust. Current research has developed an architecture, which acquires and stores the sensor data; but the data usage or analysis part is yet to develop. Therefore making a comparison with HL7 data sharing standards would be early at this stage. But since the architecture is flexible

so the adaptation of an HL7 standard is possible. Following we discuss some HL7 standards with respect to the characteristics of the project.

Because CDA is implemented in XML, so we can make our project XML-capable to work with it. CDA compatibility works in two ways, (i) parsing a received CDA document is possible by having its display within a browser using an XML Extensible (XSL) Stylesheet; and (ii) for CDA generation, implementers have variety of mechanisms [201]. Similarly HL7 v2 says that CDA message encoding should be in JSON [205], which is the characteristic of this project. To comply with FHIR standard requirements, the ANTContext server also supports RESTful architecture. Therefore customizing the architecture for next generation exchange of resources or documents would also be possible. CDA introduces the concept of "incremental" semantic interoperability; so to range the complexity, within our project we can introduce different compliance levels [201].

CCOW standard is one of the tough standards to meet, as its goal is the real-time adjustment of the context data of all the opened applications according to the chosen patient, just by changing the context within a single application. This is possible by allow delegation of rights within different applications. In the latest CCOW v1.5 [332] the Simple Object Access Protocol (SOAP)-based applications and agents can send and receive SOAP messages to the context manager. Once the primary standards are achieved, HL7 v3 provides Clinical and Administrative Domain protocol, in which an Event Publish & Subscribe Service can subscribe to a consolidate messages regarding relevant clinical events, and the project in hand can support these features technically through its EDA middleware [204].

11.3 Conclusion

The increasing population of ageing society demands new ICT based solutions, such as smart home environments, to meet the new challenges. The main mechanism is to keep an eye over a patients, and to continuously monitor their health and activities in real-time, for the provision of healthcare, comfort and safety to improve their quality of living. Such SHE depend upon sensor technologies, which continuously transmit abundance of different types of sensor data. Hence, these SHE applications are data-driven, and decides their actions and flow according to the data. But there are many responsible factors which have affected adversely the development of such healthcare environment systems, such as devices heterogeneity, multiple vendors, different data formats, inefficient data representations and transmission techniques, simple one directional communication, persistent communication middlewares, unscalable architectures, inefficient methodologies, and lack of framework guidance to develop real-time systems.

This thesis presents an architecture of a real-time distributed system, its method-

ology describing the principles and practices how to acquire, transmit and store sensor data; and an architectural framework as a reference for future real-time similar applications. This research facilitates the users, developers, and architects. A user is flexible in many ways, such as in choosing any ANT+ vendor's hardware, he don't have to login to turn on or run the Windows service, he can access the remote server from any where, the service allows him to save his preferred settings and profiles in XML file, and he can enjoy real-time services using the real-time Web architecture. Whereas a developer can use the described methodology, to follow the methods and principles to monitor environments using M2M sensor data in real-time scenarios ranging from data acquisition to storage and analysis. Using MVC architecture he will be flexible in designing his application logic and control and will able to develop as many interfaces as he want for the patients. This separation of view from the Model and Controller will also allow him to target personalized interfaces to address the needs of the disabled elderly people. This design pattern will give him flexibility both at the control unit as well at the interface level. He will able to design different processes and services target-able by an event generated from either end of the system within an EDA. The WebSockets based real-time bi-directional communication, especially the Socket.IO frameworks will be robust to deliver timely data. Its bi-directional and permanently connected state will allow either sides to know automatically that when the other side has left or disconnected. Using Socket.IO framework a developer can design specific groups of users as namespaces to address each of them separately based upon the EDA logic. This EDA will also allow him to design and develop publish-subscribe systems and to allow lose coupling between the distributed component's. Special JSON messages needs no further processing and are executable and storable in Node.js server, which uses only one real-time connection with the database to deal all the client requests. Node Program Manager (npm) allow the addition of Node.js distributed autonomous middlewares as an additional services for the application hence facilitating the building of large ecosystems. Using NoSQL databases, such as MongoDB (a document-oriented database), a developer will be flexible in writing application logic targeting different models of an application to save sensor data in schema flexible format. The ORM like Mongoose will allow the developer to keep the schemas out of the database and also to be rational in business logic. Document-oriented databases are schema flexible, as one can alter the schema at the run time hence providing flexibility to the programmers to save an object instances in different formats. Alongside NoSQL databases are highly scalable, robust and flexible in storing and analysing data. The storage techniques such as de-normalization based schema models allow robust storage and efficient query processing.

An architect can use the Architectural framework as a reference to design and develop SHE by selecting artefacts for the four layers, i.e. (i) M2M hardware sen-

sor device at the things realization layer; (ii) extracting and processing M2M profile semantics based upon static interpreters at the interpretation and information processing layer; (iii) usage of WebSockets, JSON and real-time Node.js Server and event driven based communicate at the real-time communication and middleware layer; and (iv) leveraging the NoSQL storage and Node.js middlewares to build applications and services at the data integration and applications layer.

Using this architectural framework, the designed real-time flexible architectures will be scalable both horizontally as well as vertically, means that the system will be deployable at a large scale. These technical architectures will be domain neutral and their applications will not be limited to the healthcare domain only.

11.4 Future Work

Although the thesis presents significant contributions to SHE domain with the above described goals; but there are still many features and research problems that have not been scrutinized, or that have appeared as a consequence of the advancement steps during the course. Since this is a project oriented work having flexible and scalable architecture, so there is big possibility to add additional features to the system. Following subsections discuss the extension of the current research in context to two dimensions – i.e., (a) With respect to the prototype's specific components; and (b) In general addition of the possible features or functionality.

11.4.1 Prototype Specific Future Directions

Specific future directions with respect to the different developed components during the final prototype are discussed in this subsection. Defining a complete event-driven system between the two main distributed components along a use case description will increase the usability of the product. Since the prototype uses Socket.IO which allows the usage of name-spaces, so using this way one can define groups and can target a set of users in a differentiated manner. The use of sockets allows the usage of different endpoints or paths. This will allow us to define topic based endpoints, and namespace having list of users to build pub-sub system. There are many other middlewares that can enhance the productivity of the system. The system handles the data of the ANT+ sensors with their least message rate, which makes it more static and less useful. Extending it for the other message rates along an automatic window size adjustable feature, will definitely improve the functionality. Extending the work for other less deep , hierarchical or denormalized temporal schema models will result into interesting results; especially their comparison with other schema models, such as per-minute, per hour etc. Defining different MongoDB aggregate queries will produce interesting re-

sults. Performing predictive analysis during the data acquisition and to generate events upon analysis can make the work more responsive.

11.4.2 In General Future Directions

Few important missing features we have already identified during different chapters but not covered in this prototype are described below.

Context Awareness: This is to understand the context and react accordingly by generating special events after inferencing the environment. The program logic for such decision making approaches mainly reside at the user's end. For this different AI techniques are possibly embedded into the system.

Web Portal and Services: The project can be extended by having a fully fledged data analysis and visualization distributed components at the context data server end. Such component will help the different stakeholders to interact with each other, such as hospitals, doctors, user's friends and relatives. Beside this there is lot of potential to add valuable services into the system.

User Interfaces: Since in the AAL domain the interface interaction is an important factor, and the current thesis supports the addition of any number of Views or interfaces because of its flexible architecture, especially because of its MVC based design. Therefore different addition of interfaces into the system is an important research direction to follow in the long run.

Distributed Systems: The system can be extended to provide adaptability according to the user's context and based upon user's interactions. Since AAL applications are life saving systems, therefore fault tolerance management against the life critical services is an important factor to add to the project scalability. Therefore in this way reconfiguration will be achieved. Once the services are added into the project, and since they depends upon each other for the combined effect, therefore for these purposes a SOA based approach will be highly desirable for the scalability and management.

Personalization: In order to understand its user and his needs by modelling his profile and behaviour. For this domain we need to model the user's properties and behaviour, like in the facebook social profiles.

Assistive Technology: To overcome the disability of the users, assistive services will be most valuable additions. These services can restore the disable and old people to their usual normal life. This involves both the hardware based (such as, wheel chairs, etc.), as well as software based assistive technology. In this section, we discuss very limited number of software based assistive services, as they would be integratable to the system easily because of the system's flexible MVC based architecture. We can categorise assistive services into four basic types - i.e, communication, visual, physical and hearing disabilities. In context to the **communication** disorders, we have Electronic Speech Synthesizer systems, which is

is an artificial production of human speech from text or recorded speech stored within a database [42]. Microsoft Speech API for Windows XP[1], for Windows 10[2] and for Windows 7[3]&8[4], support speech synthesis and speech recognition. In the same manner to solve the **visual** impairments[5], screen readers are mostly used[6]. This may also be enhanced with screen magnifiers[7]. To deal with the **physical** disabilities, systems use tools that may understand the spoken commands by the user. For example Voice Finger[8] is a windows based tool for mouse and keyboard control with the help of voice. In context to the **hearing** disabilities, systems depends upon Real-time Closed Captioning(CC) techniques, such as Microsoft's Accessible Media Interchange (SAMI) technology[9]. This technique is very much used for the videos[10].

Streaming data: The current prototype uses a customized stream processing approach which handles the data management to store it according to the timestamps. There is big possibility to define and use other management schemas and also to define evolution algorithms. Beside this the usage of data stream processing engines, as we discussed during the Section 6.4.1 may prove a good addition to the project.

Targeting other platforms: There are different platforms that if targeted would result into valuable additions to the healthcare sector. For this purpose the proposed methodology would help the scientists to develop system for other platforms, such as Raspberry Pi[11], mobile platforms. Targeting the Pi platform there are approaches[12], [13] available to have a good start; whereas for the mobile platform we have guidance at the ANT+ website[139].

Support for other Hardware: Since the technology hardware industry, with miniaturization and resource optimization features, are the key drivers for the next the Internet of Things, therefore enhancing the system in this direction will produce valuable results. The current research product currently supports only the ANT+ sensors but its scalability for other useful products will be a good addition to target more services, and also to achieve the IoT goal. There is a range of op-

[1] www.microsoft.com/enable/training/windowsxp
[2] www.microsoft.com/enable/products/windows10/default.aspx
[3] www.microsoft.com/enable/products/windows7/default.aspx
[4] www.microsoft.com/enable/products/windows8/default.aspx
[5] www.microsoft.com/enable/guides/vision.aspx
[6] windows.microsoft.com/en-us/windows/hear-text-read-aloud-narrator
[7] http://windows.microsoft.com/en-us/windows-8/use-magnifier-see-items
[8] voicefinger.cozendey.com
[9] msdn.microsoft.com/en-us/library/ms971327.aspx
[10] http://ccextractor.sourceforge.net/about-ccextractor.html
[11] /www.raspberrypi.org
[12] blog.howrandom.net, github.com/tomwardill/developerhealth
[13] johannesbader.ch/2014/06/track-your-heartrate-on-raspberry-pi-with-ant

tions to enhance the project, like by using Nordic Semiconductor's *nRF52832*[14] and *N5 ANT SoC Module Series*[15] multi-protocol System on Chip (SOC), which concurrently supports ANT/Bluetooth Low Energy (BLE) soft devices. Similarly to connect hundreds of devices simultaneously Dynastream[358] provides, *IoT High Node* Count[16] solutions.

Similarly support for other hardware, such as *Wii family*[17] of controllers (Wii Remote, Wii Balance Board) to detect image based user's movements. This will definitely asks for the image analysis components within the system, beside the sensor communication enabling. Similarly, Microsoft's *Kinect*[18] will enhance the productivity in context to gesture detection with applications to disable users; and this has also became a trend for modern human computer interaction. Some more exotic devices such as the *Neurosky Mindwave*[19], record and can translate into controller input and can measure the mind relaxation level of the users.

Stream Analysis: Using rule based or stream processing engines might produce valuable results. We discussed some options in Section 6.4.1.

Pub-Sub System: Since the system architecture supports event-driven communication, which decouple the distributed components. The architecture supports the system enhancement for a publish subscribe based healthcare data sharing. The developers can introduce their own customized events, and can also define a set of users by using name-space feature provided by Socket.IO.

[14] www.thisisant.com/developer/components/nrf52832/

[15] https://www.thisisant.com/developer/components/n5/

[16] www.thisisant.com/news/ant-wireless-showcases-powerful-internet-of-things-high-node-count-solution

[17] http://wiibrew.org/wiki/Wiimote

[18] http://msdn.microsoft.com/en-us/library/jj131033.aspx

[19] http://store.neurosky.com/

Bibliography

[1] *Introduction to windows service applications*. 77, 79

[2] *Microsoft announces the release of windows nt workstation 4.0*. 79

[3] *Microsoft default services: Sensor monitoring service (window 8)*. 79

[4] *Microsoft default services: Sensor services (window 10)*. 79

[5] *Microsoft releases windows 2000 to manufacturing*. 79

[6] *Model-view-controller*. 80, 81

[7] *Mongodb aggregation*. 116, 168

[8] *Mongodb cloud manager*. 114

[9] *Mongodb for giant ideas*. 106

[10] *Mongodb for time series data (webinar series)*. 114, 115

[11] *Mongodb gridfs api*. 115

[12] *Mongodb limits and thresholds*. 115

[13] *Mongojack*. 113

[14] *Mongolink: An object document mapper (odm) for java and mongodb*. 113

[15] *mongoose:elegant mongodb object modeling for node.js*. 114

[16] *Neo4j*. 106

[17] *Oracle berkeley db*. 105

[18] *Poco support in .net framework*. 113

[19] *Query in java as in mongo shell*. 113

[20] *Redis is an open source in-memory data store*. 105

[21] *Scalaris, a distributed transactional key-value store*. 105

[22] *Sensor api*. 79

[23] *Service control manager*. 77

[24] *Services overview*. 77

[25] *Socket.io official site*. 43, 92

[26] *System apis and services*. 78

[27] *Technet: Net start*. 78

[28] *Tokyo cabinet: a modern implementation of dbm*. 105

[29] *Tokyo tyrant*. 105

[30] *Windows deployment and resource kits*. 79

[31] *Ieee 802.15.4 strandard on wireless lan medium access control (mac) an physical layer (phy) specifications for low rate wireless personal area networks*, tech. report, Institute Electrical Electronics Engineers 802.15 WPAN Working Group, 2006. 27, 42, 56, 57

[32] *Project voldemort*, 2013. 105

[33] E. AARTS AND R. ROOVERS, *Embedded system design issues in ambient intelligence*, in Ambient Intelligence: Impact on Embedded Sytem Design, Springer, 2003, pp. 11–29. 11

[34] E. AARTS AND R. WICHERT, *Ambient intelligence*, Springer, 2009. 10, 39

[35] T. AASEB, *Near field communication, bluetooth, zigbee and ant+ lecture notes*, 2014. 61

[36] F. ABADIE, C. CODAGNONE, L. VAN, C. PASCU, P. BAUM, A. HOIKKANEN, ET AL., *Strategic intelligence monitor on personal health systems (simphs). market structure and innovation dynamics. luxembourg: Jrf scientific and technical reports. european commission, joint research center (jrc)*, Institute for Prospective Technological Studies. Publications Office of the European Union, (2011). 9

[37] S. ABITEBOUL, *Querying semi-structured data*, Springer, 1997. 101

[38] M. R. ALAM, M. B. I. REAZ, AND M. A. M. ALI, *A review of smart homes–past, present, and future*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42 (2012), pp. 1190–1203. 12, 34, 38, 40

[39] F. K. ALDRICH, *Smart homes: past, present and future*, in Inside the smart home, Springer, 2003, pp. 17–39. 35

[40] H. ALEMDAR AND C. ERSOY, *Wireless sensor networks for healthcare: A survey*, Computer Networks, 54 (2010), pp. 2688 – 2710. xx, 9, 56

[41] A. ALINONE, *Comet and push technology*, 2007. 87

[42] J. ALLEN, M. S. HUNNICUTT, D. H. KLATT, R. C. ARMSTRONG, AND D. B. PISONI, *From text to speech: The MITalk system*, Cambridge University Press, 1987. 189

[43] Z. ALLIANCE, *Zigbee enables smart buildings of the future today*, Zigbee Whitepaper, (2007). 59

[44] Z. ALLIANCE, *Zigbee home automation public application profile*, IEEE J. Select. Areas Commun, (2007). 59

[45] ——, *Zigbee wireless sensor applications for health, wellness and fitness*, White paper. March, (2009). 59

[46] Z. ALLIANCE AND H. ALLIANCE, *Smart energy profile 2 application protocol standard*, 2013. 16, 59

[47] H. ALOULOU, *Framework for ambient assistive living: handling dynamism and uncertainty in real time semantic services provisioning*, PhD thesis, Evry, Institut national des télécommunications, 2014. 12

[48] AMSTERDAM, *Amsterdam smart city (asc)*, 2013-2020. 34

[49] A. ARASU, B. BABCOCK, S. BABU, M. DATAR, K. ITO, I. NISHIZAWA, J. ROSENSTEIN, AND J. WIDOM, *Stream: the stanford stream data manager (demonstration description)*, in Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM, 2003, pp. 665–665. 97

[50] R. ARORA AND R. R. AGGARWAL, *Modeling and querying data in mongodb*, International Journal of Scientific and Engineering Research, 4 (2013). 49

[51] M. ASP.NET, *Signalr: Persistent connections (server)*, 2013. 91

[52] ——, *Asp.net signalr hubs api guide - server (c#)*, 2014. 91

[53] A. ASSOCIATION, *Alzheimer's disease: the costs to u.s. business in 2002*, 2016. 33

[54] A. T. A. (ATA), *What is telemedicine*, 2016. 8

[55] I. N. ATHANASIADIS, P. A. MITKAS, A. E. RIZZOLI, AND J. MARX, *Noisetube: Measuring and mapping noise pollution with mobile phones*, (2009). 39

[56] J. C. AUGUSTO AND P. MCCULLAGH, *Ambient intelligence: Concepts and applications*, Computer Science and Information Systems, 4 (2007), pp. 1–27. xx, 10

[57] A. BACA, P. KORNFEIND, E. PREUSCHL, S. BICHLER, M. TAMPIER, AND H. NOVATCHKOV, *A server-based mobile coaching system*, Sensors, 10 (2010), pp. 10640–10662. 46

[58] C. BADICA, M. BREZOVAN, AND A. BADICA, *An overview of smart home environments: Architectures, technologies and applications.*, in BCI (Local), 2013, p. 78. 34, 38, 40, 54, 75, 76

[59] F. BAJABER, S. SAKR, O. BATARFI, A. ALTALHI, R. ELSHAWI, AND A. BARNAWI, *Big data processing systems: State-of-the-art and open challenges*, in Cloud Computing (ICCC), 2015 International Conference on, IEEE, 2015, pp. 1–8. 102, 116

[60] N. BAKER, *Zigbee and bluetooth strengths and weaknesses for industrial applications*, Computing & Control Engineering Journal, 16 (2005), pp. 20–25. 59

[61] S.-D. BAO, Y.-T. ZHANG, AND L.-F. SHEN, *Physiological signal based entity authentication for body area sensor networks and mobile healthcare systems*, in Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the, IEEE, 2005, pp. 2455–2458. 57

[62] T. BART, *Comparison of electronic data capture with paper data collectionis there really an advantage*, (2003). 79

[63] L. BASS, *Software architecture in practice*, Pearson Education India, 2007. 14

[64] L. M. BECA, G. CHENG, G. C. FOX, T. JURGA, K. OLSZEWSKI, M. PODGORNY, P. SOKOLOWSKI, T. STACHOWIAK, AND K. WALCZAK, *Tango-a collaborative environment for the world-wide web*, (1997). 87

[65] R.-D. BERNDT, M. TAKENGA, S. KUEHN, P. PREIK, N. STOLL, K. THUROW, M. KUMAR, M. WEIPPERT, A. RIEGER, AND R. STOLL, *A scalable and secure telematics platform for the hosting of telemedical applications. case study of a stress and fitness monitoring*, in e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on, June 2011, pp. 118–121. 46

[66] P. A. BERNSTEIN, *Middleware: a model for distributed system services*, Communications of the ACM, 39 (1996), pp. 86–98. xx, 69, 70

[67] M. BEYER, *Gartner says solving'big data'challenge involves more than just managing volumes of data*, Gartner. Archived from the original on, 10 (2011). 104

[68] D. BHATIA, L. ESTEVEZ, AND S. RAO, *Energy efficient contextual sensing for elderly care*, in Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, IEEE, 2007, pp. 4052–4055. 57

[69] G. BLAIR, G. COULOURIS, J. DOLLIMORE, AND T. KINDBERG, *Distributed Systems: Concepts and Design*, Boston: Addison-Wesley, 2012. 13, 16, 54

[70] D. E. BLOOM, D. CANNING, AND G. FINK, *Implications of population ageing for economic growth*, Oxford Review of Economic Policy, 26 (2010), pp. 583–612. 3, 4

[71] BLUAIR, *Bluetooth range*, 2016. xxii, 58

[72] S. BLUETOOTH, *Bluetooth special interest group*, Simple Pairing Whitepaper (Revision V10r00), 17 (2006). 58

[73] BLUETOOTH.COM, *Bluetooth adopted specifications*, 2016. 58

[74] ——, *Bluetooth core specifications*, 2016. 16, 58, 59

[75] R. BLUMBERG AND S. ATRE, *The problem with unstructured data*, DM REVIEW, 13 (2003), p. 62. 20, 101

[76] T. BODENHEIMER, K. LORIG, H. HOLMAN, AND K. GRUMBACH, *Patient self-management of chronic disease in primary care*, Jama, 288 (2002), pp. 2469–2475. 6, 33

[77] D. BONINO, E. CASTELLINA, AND F. CORNO, *The dog gateway: enabling ontology-based intelligent domotic environments*, Consumer Electronics, IEEE Transactions on, 54 (2008), pp. 1656–1664. 44, 76

[78] M. N. K. BOULOS AND N. M. AL-SHORBAJI, *On the internet of things, smart cities and the who healthy cities*, International journal of health geographics, 13 (2014), p. 1. 39

[79] H. BOUMA, *Gerontechnology: Making technology relevant for the elderly*, Gerontechnology, (1992), pp. 1–5. 8

[80] E. BOZDAG, A. MESBAH, AND A. VAN DEURSEN, *A comparison of push and pull techniques for ajax*, in Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on, IEEE, 2007, pp. 15–22. 87

[81] T. BRAY, *The javascript object notation (json) data interchange format*, tech. report, IETF, Mar 2014. 148

[82] T. BRAY, J. PAOLI, C. M. SPERBERG-MCQUEEN, E. MALER, AND F. YERGEAU, *Extensible markup language (xml)*, World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210, 16 (1998). 148

[83] B. BRUMITT, B. MEYERS, J. KRUMM, A. KERN, AND S. SHAFER, *Easyliving: Technologies for intelligent environments*, in Handheld and ubiquitous computing, Springer, 2000, pp. 12–29. 44

[84] F. BUGIOTTI, L. CABIBBO, P. ATZENI, AND R. TORLONE, *Database design for nosql systems*, in Conceptual Modeling, Springer, 2014, pp. 223–231. 49, 110

[85] P. BUNEMAN, *Semistructured data*, in Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, ACM, 1997, pp. 117–121. 101

[86] S. BURBECK, *Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc)*, Smalltalk-80 v2, 5 (1992). 80

[87] D. BURKE, B. WANG, T. T. WAN, AND M. DIANA, *Exploring hospitals' adoption of information technology*, Journal of medical systems, 26 (2002), pp. 349–355. 49

[88] M. BURROWS, *The chubby lock service for loosely-coupled distributed systems*, in Proceedings of the 7th symposium on Operating systems design and implementation, USENIX Association, 2006, pp. 335–350. 105

[89] F. BUSCHMANN, K. HENNEY, AND D. SCHIMDT, *Pattern-oriented Software Architecture: On Patterns and Pattern Language*, vol. 5, John wiley & sons, 2007. 30

[90] C. CACCIA, *Management of health care information systems*, 2008. 51

[91] D. CARNEY, U. ÇETINTEMEL, M. CHERNIACK, C. CONVEY, S. LEE, G. SEIDMAN, M. STONEBRAKER, N. TATBUL, AND S. ZDONIK, *Monitoring streams: a new class of data management applications*, in Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 215–226. 97

[92] R. CATTELL, *Scalable sql and nosql data stores*, ACM SIGMOD Record, 39 (2011), pp. 12–27. 103

[93] M. CHAN, E. CAMPO, D. ESTÈVE, AND J.-Y. FOURNIOLS, *Smart homes–current features and future perspectives*, Maturitas, 64 (2009), pp. 90–97. xx, 35, 36, 37, 40

[94] S. CHANDRASEKARAN, O. COOPER, A. DESHPANDE, M. J. FRANKLIN, J. M. HELLERSTEIN, W. HONG, S. KRISHNAMURTHY, S. R. MADDEN, F. REISS, AND M. A. SHAH, *Telegraphcq: continuous dataflow processing*, in Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM, 2003, pp. 668–668. 97

[95] F. CHANG, J. DEAN, S. GHEMAWAT, W. C. HSIEH, D. A. WALLACH, M. BURROWS, T. CHANDRA, A. FIKES, AND R. E. GRUBER, *Bigtable: A distributed storage system for structured data*, ACM Transactions on Computer Systems (TOCS), 26 (2008), p. 4. 105

[96] M. CHEN, S. GONZALEZ, A. VASILAKOS, H. CAO, AND V. C. LEUNG, *Body area networks: A survey*, Mobile Network Applications, 16 (2011), pp. 171–193. 9, 57

[97] P. P.-S. CHEN, *The entity-relationship model-toward a unified view of data*, ACM Transactions on Database Systems (TODS), 1 (1976), pp. 9–36. 111

[98] K. CHODOROW, *MongoDB: the definitive guide*, " O'Reilly Media, Inc.", 2013. 46

[99] C.-Y. CHONG AND S. P. KUMAR, *Sensor networks: evolution, opportunities, and challenges*, Proceedings of the IEEE, 91 (2003), pp. 1247–1256. 55

[100] E. CHRISTENSEN, F. CURBERA, G. MEREDITH, S. WEERAWARANA, ET AL., *Web services description language (wsdl) 1.1*, 2001. 147

[101] E. CITIES, *Euro cities*, 2013-2020. 34

[102] E. F. CODD, *A relational model of data for large shared data banks*, Commun. ACM, 13 (1970), pp. 377–387. 111, 112

[103] E. F. CODD, *Does your dbms run by the rules?*, Computer World, 21 (1985), p. 11. 99, 112

[104] ——, *Is your dbms really relational*, Computer World, 14 (1985). 99

[105] W. COLITTI, K. STEENHAUT, AND N. DE CARO, *Integrating wireless sensor networks with the web*, Extending the Internet to Low power and Lossy Networks (IP+ SN 2011), (2011). 15

[106] E. COMMISSION, *Research and innovation for ageing well with ict*, (02/03/2015). 6, 7

[107] E. COMMISSION, *Horizon 2020 : The eu framework programme for research and innovation*, 2013-2020. 7, 34

[108] ——, *Smart cities and communities*, 2013-2020. 34

[109] ——, *Urban europe 2020: An eu urban agenda for the smart cities of tomorrow*, 2013-2020. 34

[110] W. W. W. CONSORTIUM ET AL., *Web services glossary*, Online: http://www.w3.org/TR/ws-gloss, (2004). 147

[111] N. C. CORPORATION, *Introducing netscape navigator 2.0*, 1996. 87

[112] O. CORPORATION, *Html5 server-push technologies, part 1 blog*, 2010. 88

[113] M. CORSO AND L. GASTALDI, *Managing ict-driven innovation in the health care industry: evidence from an empirical study in italy*, in Proceedings of the 10th CINet Conference, 2009, pp. 1–14. xx, 50, 51, 52

[114] U. K. COUNCEL AND CARE, *Community care assessment and services*. 12

[115] J. COUTAZ, *Pac, an object oriented model for dialog design*, in Proceedings Interact, vol. 87, 1987, pp. 431–436. 80

[116] D. CRANE AND P. MCCARTHY, *Comet and reverse ajax: The next-generation ajax 2.0, 2008.*, tech. report, ISBN 978-1-59059-998-3. 87

[117] A. CUZZOCREA, *Temporal aspects of big data management: State-of-the-art analysis and future research directions*, in Temporal Representation and Reasoning (TIME), 2015 22nd International Symposium on, IEEE, 2015, pp. 180–185. 20, 101, 103, 165

[118] C.-W. DAI, S.-H. YANG, AND R. KNOTT, *Data transfer over the internet for real time applications*, International Journal of Automation and Computing, 3 (2006), pp. 414–424. 84

[119] W. DARGIE AND C. POELLABAUER, *Fundamentals of Wireless Sensor Networks: Theory and Practice*, Wireless Communications and Mobile Computing, Wiley, 2010. 10, 16, 58, 60

[120] R. DAŞ AND G. TUNA, *Machine-to-machine communications for smart homes*, International Journal of Computer Networks and Applications (IJCNA)), 2 (2015). 42

[121] L. C. DE SILVA, C. MORIKAWA, AND I. M. PETRA, *State of the art of smart homes*, Eng. Appl. Artif. Intell., 25 (2012), pp. 1313–1321. 12, 40

[122] J. DEAN AND S. GHEMAWAT, *Mapreduce: Simplified data processing on large clusters*, Commun. ACM, 51 (2008), pp. 107–113. 116, 168

[123] G. DECANDIA, D. HASTORUN, M. JAMPANI, G. KAKULAPATI, A. LAKSHMAN, A. PILCHIN, S. SIVASUBRAMANIAN, P. VOSSHALL, AND W. VOGELS, *Dynamo: amazon's highly available key-value store*, in ACM SIGOPS Operating Systems Review, vol. 41, ACM, 2007, pp. 205–220. 104

[124] I. DEFINING TECHNOLOGY, *Middleware resource center*, 2014. xx, 14

[125] M. DERIU, G. PADDEU, AND A. SORO, *Xplaces: An open framework to support the digital living at home*, in GreenCom/CPSCom, 2010. 44

[126] J. E. DEUTSCH, J. A. LEWIS, AND G. BURDEA, *Technical and patient performance using a virtual reality-integrated telerehabilitation system: preliminary finding*, Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 15 (2007), pp. 30–35. 12

[127] A. DOAN, A. HALEVY, AND Z. IVES, *Principles of data integration*, Elsevier, 2012. 147

[128] C. DOUBLE, *More on ajax and server push*, 200. 87

[129] K. DUCATEL, M. BOGDANOWICZ, F. SCAPOLO, J. LEIJTEN, AND J.-C. BURGELMAN, *Scenarios for ambient intelligence in 2010*, Office for official publications of the European Communities, 2001. 10

[130] ——, *Ambient intelligence: From vision to reality*, IST Advisory Group Draft Report, European Commission, (2003). 10

[131] I. DYNASTREAM CORPORATION, *Ant+ device profile: Environment, revision 1.0*, tech. report. 122, 166

[132] ——, *Ant+ device profile: Heart rate monitor, revision 1.13*, tech. report. 122, 166

[133] ——, *Ant+ device profile: Stride based speed and distance monitor, revision 1.3*, tech. report. 122, 155, 166

[134] ——, *Ant macosx library package with source code*. 66

[135] ——, *Ant network keys*. 66

[136] ——, *Ant tech faq : Usb drivers*. 66

[137] ——, *Ant windows library package*. 65, 66

[138] ——, *Antusb2 stick*. 66

[139] ——, *Starting you project*. 65, 66, 189

[140] ——, *Ant message protocol and usage: Application notes,version 2.1*, online doc, Jul 2007. xxi, 61, 62, 63, 66, 128, 129, 165, 169, 222, 223

[141] D. EDIGER, R. MCCOLL, J. POOVEY, AND D. CAMPBELL, *Scalable infrastructures for data in motion*, in Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, May 2014, pp. 875–882. 20, 101

[142] EDITOR, *The 2012 ageing report: Economic and budgetary projections for the 27 eu member states (2010-2060)*, tech. report, European Commission, 2012. 4

[143] A. EGLOFF, *Ajax push (aka comet) with java business integration (jbi)*, in Presentation at the JavaOne 2007 Conference, San Francisco, CA, vol. 5, 2007. 87

[144] M. EISENHAUER, M. JAHN, F. PRAMUDIANTO, T. SABOL, AND J. HRENO, *Towards a generic middleware for developing ambient intelligence applications*, in 2nd Workshop on eeBuildings Data Models at CIB W078–W102, 2011, pp. 26–28. 76

[145] J. ELIAS, *Optimal design of energy-efficient and cost-effective wireless body area networks*, Ad Hoc Networks, 13 (2014), pp. 560–574. 57

[146] R. ELMASRI, *Fundamentals of database systems*, Pearson Education India, 2008. 15, 28, 101, 102, 109, 111, 112, 167

[147] H. EREN AND J. G. WEBSTER, *Telemedicine and Electronic Medicine*, CRC Press, 2015. 8

[148] T. ETSI, *102 689 v1. 1.1 (2010-08): Machine-to-machine communications*, 2010. 18

[149] ——, *102 690:" machine-to-machine communications (m2m)*, Functional architecture, (2011). 16

[150] P. T. EUGSTER, P. A. FELBER, R. GUERRAOUI, AND A.-M. KERMARREC, *The many faces of publish/subscribe*, ACM Computing Surveys (CSUR), 35 (2003), pp. 114–131. 15

[151] EUROSTAT, *Cardiovascular diseases statistics*, 2016. 5, 33

[152] W. L. EXPECTANCY, *World health ranking: Italy top 50 causes of death*, 2016. 5

[153] ——, *World health ranking: Pakistan top 50 causes of death*, 2016. 5

[154] ——, *World health rankings: Sudan-coronary heart disease*, 2016. 5

[155] M. FAYAD AND D. C. SCHMIDT, *Object-oriented application frameworks*, Communications of the ACM, 40 (1997), pp. 32–38. 17

[156] L. FERNANDEZ-LUQUE, R. KARLSEN, AND L. K. VOGNILD, *Challenges and opportunities of using recommender systems for personalized health education.*, in MIE, 2009, pp. 903–907. 36

[157] I. FETTE AND A. MELNIKOV, *Protocol overview: Rfc 6455 the websocket protocol*, tech. report, IETF, Dec 2011. 88, 90

[158] R. FIELDING, J. GETTYS, J. MOGUL, H. FRYSTYK, L. MASINTER, P. LEACH, AND T. BERNERS-LEE, *Hypertext transfer protocol–http/1.1*, 1999. 86

[159] D. FLANAGAN, *JavaScript: the definitive guide*, " O'Reilly Media, Inc.", 2006. 87

[160] A. FOR HEALTHCARE RESEARCH AND QUALITY, *Healthcare costs*, 2016. 6, 33

[161] B. A. FOROUZAN, *TCP/IP protocol suite*, McGraw-Hill, Inc., 2002. 145

[162] FP7, *Buttler eu project*, 2016. 39

[163] T. GAO, C. PESTO, L. SELAVO, Y. CHEN, J. KO, J. LIM, A. TERZIS, A. WATT, J. JENG, B.-R. CHEN, ET AL., *Wireless medical sensor networks in emergency response: Implementation and pilot results*, in Technologies for Homeland Security, 2008 IEEE Conference on, IEEE, 2008, pp. 187–192. 57

[164] D. GARLAN AND M. SHAW, *An introduction to software architecture*, Advances in software engineering and knowledge engineering, 1 (1993). 53

[165] I. GARMIN, *Garmin connect*. 131

[166] J. J. GARRETT ET AL., *Ajax: A new approach to web applications*, (2005). 86

[167] V. GAY AND P. LEIJDEKKERS, *Bringing health and fitness data together for connected health care: mobile apps as enablers of interoperability*, Journal of medical Internet research, 17 (2015). 184

[168] S. K. GHARGHAN, R. NORDIN, AND M. ISMAIL, *A survey on energy efficient wireless sensor networks for bicycle performance monitoring application*, Journal of Sensors, (2014). 47

[169] S. K. GHARGHAN, R. NORDIN, AND M. ISMAIL, *An ultra-low power wireless sensor network for bicycle torque performance measurements*, Sensors, 15 (2015), p. 11741. 48, 59

[170] S. GHEMAWAT, H. GOBIOFF, AND S.-T. LEUNG, *The google file system*, in ACM SIGOPS operating systems review, vol. 37, ACM, 2003, pp. 29–43. 105

[171] L. GOLAB AND M. T. ÖZSU, *Issues in data stream management*, SIGMOD Rec., 32 (2003), pp. 5–14. 19, 96, 103

[172] B. GONÇALVES, R. V. ANDREÃO, G. GUIZZARDI, ET AL., *Ecg data provisioning for telehomecare monitoring*, in Proceedings of the 2008 ACM symposium on Applied computing, ACM, 2008, pp. 1374–1379. 128

[173] B. GONÇALVES, G. GUIZZARDI, ET AL., *A service architecture for sensor data provisioning for context-aware mobile applications*, in Proceedings of the 2008 ACM symposium on Applied computing, ACM, 2008, pp. 1946–1952. 130, 147

[174] GOOGLE, *Mjorm (mongo-java-orm) - a mongodb java orm*. 113

[175] D. A. GRATTON, *The handbook of personal area networking technologies and protocols*, Cambridge University Press, 2013. 10, 16, 56, 57, 59, 60

[176] K. GROLINGER, M. HAYES, W. A. HIGASHINO, A. L'HEUREUX, D. S. ALLISON, AND M. A. M. CAPRETZ, *Challenges for mapreduce in big data*, in Services (SERVICES), 2014 IEEE World Congress on, June 2014, pp. 182–189. 102

[177] W. GROPP, E. LUSK, AND A. SKJELLUM, *Using MPI: portable parallel programming with the message-passing interface*, vol. 1, MIT press, 1999. 73

[178] J. GUBBI, R. BUYYA, S. MARUSIC, AND M. PALANISWAMI, *Internet of things (iot): A vision, architectural elements, and future directions*, Future Generation Computer Systems, 29 (2013), pp. 1645–1660. 36

[179] V. N. GUDIVADA, D. RAO, AND V. V. RAGHAVAN, *Nosql systems for big data management*, in Services (SERVICES), 2014 IEEE World Congress on, IEEE, 2014, pp. 190–197. 104

[180] W. GUO, W. M. HEALY, AND M. ZHOU, *Battery discharge characteristics of wireless sensors in building applications*, in Networking, Sensing and

Control (ICNSC), 2012 9th IEEE International Conference on, IEEE, 2012, pp. 133–138. 59

[181] A. GUSTAVO, F. CASATI, H. KUNO, AND V. MACHIRAJU, *Web services: concepts, architectures and applications*, 2004. 69

[182] M. HAPNER, R. BURRIDGE, R. SHARMA, J. FIALLI, AND K. STOUT, *Java message service*, Sun Microsystems Inc., Santa Clara, CA, (2002). 74

[183] I. A. T. HASHEM, I. YAQOOB, N. B. ANUAR, S. MOKHTAR, A. GANI, AND S. U. KHAN, *The rise of "big data" on cloud computing: Review and open research issues*, Information Systems, 47 (2015), pp. 98–115. 102

[184] R. HECHT AND S. JABLONSKI, *Nosql evaluation: A use case oriented survey*, (2011). 102

[185] S. HELAL, W. MANN, J. KING, Y. KADDOURA, E. JANSEN, ET AL., *The gator tech smart house: A programmable pervasive space*, Computer, 38 (2005), pp. 50–60. 44

[186] G. HESSE AND M. LORENZ, *Conceptual survey on data stream processing systems*, in Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on, IEEE, 2015, pp. 797–802. 97

[187] I. HICKSON, *The websocket api*, W3C Working Draft WD-websockets-20110929, September, (2011). 89

[188] E. HORN, *Distributed applications: Introduction to middleware, what is middleware? layer between os and distributed applications.*, 2009. xx, 13, 72

[189] M.-J. HSIEH, C.-R. CHANG, L.-Y. HO, J.-J. WU, AND P. LIU, *Sqlmr: A scalable database management system for cloud computing*, in Parallel Processing (ICPP), 2011 International Conference on, IEEE, 2011, pp. 315–324. 116

[190] H. HU, Y. WEN, T.-S. CHUA, AND X. LI, *Toward scalable systems for big data analytics: a technology tutorial*, Access, IEEE, 2 (2014), pp. 652–687. 101, 103, 104, 105, 116, 168

[191] IBM, *Websphere mq*, 2016. 74

[192] IDTECHEX, *Rfid forecasts, players and opportunities 2016-2026*, Oct 2015. 38

[193] IEEE, *The institute, special report: The internet of things.*, 2014. 18

[194] ——, *Towards a definition of the internet of things (iot)*, 2015. 18

[195] ——, *Ieee 802.15 working group for wpan*, 2016. 27, 56

[196] A. INC., *Physical activity: Exercise's effects on the heart*. 121

[197] I. T. INC., *Icefaces easy ajax push tutorial*, 2006. 87

[198] R. INC, *Node hero - understanding async programming in node.js*, 2016. 92

[199] INFOQ, *Mongodb, java and object relational mapping*. 113

[200] O. S. G. INITIATIVE ET AL., *Specification overview*, 2000. 48

[201] H. L. S. INTERNATIONAL, *Hl7 frequently asked questions*. 185

[202] ——, *Ccow information for the healthcare industry*, 2016. 53

[203] ——, *Introducing hl7 fhir*, 2016. 53

[204] ——, *Introduction to hl7 standards*, 2016. 53, 185

[205] H. INTERSECTIONS, *A json representation for hl7 v2?* 185

[206] ITU, *Itu releases 2015 ict figures*, 2015. 7, 38

[207] ——, *Ict facts and figures 2016*, 2016. 7

[208] O. JANKOVIĆ, *Nosql dokument baza podataka: prikaz skladištenja podataka sa osvrtom na podatke sa senzora*, Infoteh-Jahorina, INFOTEH-JAHORINA, 14, pp. 561–566. 49

[209] JAVA, *Java iot: Article cover story: What is pojo programming?* 113

[210] L. JIANG, D.-Y. LIU, B. YANG, ET AL., *Smart home research*, in Proceedings of the Third Conference on Machine Learning and Cybernetics SHANGHAI, 2004, pp. 659–664. 38

[211] W.-R. JIH, J. Y.-J. HSU, C.-L. WU, C.-F. LIAO, S.-Y. CHENG, ET AL., *A multi-agent service framework for context-aware elder care*, in Proceedings of The Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE), 2006, pp. 61–75. 48

[212] A. JOHANSSON, W. SHEN, AND Y. XU, *An ant based wireless body sensor biofeedback network for medical e-health care*, in Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on, Sept 2011, pp. 1–5. 45

[213] E. JOVANOV, K. FRITH, F. ANDERSON, M. MILOSEVIC, AND M. T. SHROVE, *Real-time monitoring of occupational stress of nurses*, (2011), pp. 3640–3643. 47

[214] A. KANADE, A. GOPAL, AND S. KANADE, *A study of normalization and embedding in mongodb*, in Advance Computing Conference (IACC), 2014 IEEE International, IEEE, 2014, pp. 416–421. 110, 113, 116

[215] A. B. KARAMI, A. FLEURY, J. BOONAERT, AND S. LECOEUCHE, *User in the loop: Adaptive smart homes exploiting user feedbackstate of the art and future directions*, Information, 7 (2016), p. 35. 40

[216] F. KARRAY, M. ALEMZADEH, J. A. SALEH, AND M. N. ARAB, *Human-computer interaction: Overview on state of the art*, (2008). 10, 80

[217] I. KATSOV, *Nosql data modeling techniques*, 2012. 110, 112

[218] K. KAUR AND R. RANI, *Modeling and querying data in nosql databases*, in Big Data, 2013 IEEE International Conference on, IEEE, 2013, pp. 1–7. 49, 101, 111

[219] A. M. KELLER, R. JENSEN, AND S. AGARWAL, *Persistence software: Bridging object-oriented programming and relational databases*, in ACM SIGMOD Record, vol. 22, ACM, 1993, pp. 523–528. 101

[220] D. KELLMEREIT AND D. OBODOVSKI, *The Silent Intelligence: The Internet of Things*, DnD Ventures, 2013. 36

[221] S. KHSSIBI, H. IDOUDI, A. V. D. BOSSCHE, AND L. A. VAL, THIERRYAND SAIDANE, *Presentation and analysis of a new technology for low-power wirelesssensor network*, International Journal of Digital Information and Wireless Communications (IJDIWC), 3 (2013), pp. 75–86. xx, 58, 60, 61, 62, 66

[222] K. D. KOCHANEK, S. L. MURPHY, J. XU, AND E. ARIAS, *Mortality in the united states, 2013*, NCHS data brief, 178 (2014), pp. 1–8. 5

[223] KOGONUSO.COM, *Top 12 node.js mvc frameworks for javascript users*, 2016. 95

[224] M. KOZLOVSZKY, L. KOVACS, AND K. KAROCZKAI, *Cardiovascular and diabetes focused remote patient monitoring*, in VI Latin American Congress on Biomedical Engineering CLAIB 2014, Paraná, Argentina 29, 30 & 31 October 2014, Springer, 2015, pp. 568–571. 46

[225] H.-C. KUNG, D. L. HOYERT, J. XU, S. L. MURPHY, ET AL., *Deaths: final data for 2005*, Natl Vital Stat Rep, 56 (2008), pp. 1–120. 5

[226] K. S. KWAK, S. ULLAH, AND N. ULLAH, *An overview of ieee 802.15.6 standard*, in Applied Sciences in Biomedical and Communication Technologies (ISABEL), 2010 3rd International Symposium on, IEEE, 2010, pp. 1–6. 57

[227] G. LAFORTUNE, M. DE LOOPER, G. BALESTAT, ET AL., *Health at a glance 2009: Oecd indicators*, OECD Publishing, Paris, (2009). 6

[228] A. LAHTELA, M. HASSINEN, AND V. JYLHA, *Rfid and nfc in healthcare: Safety of hospitals medication care*, in Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on, IEEE, 2008, pp. 241–244. 57, 58

[229] A. LAKSHMAN AND P. MALIK, *Cassandra: structured storage system on a p2p network*, in Proceedings of the 28th ACM symposium on Principles of distributed computing, ACM, 2009, pp. 5–5. 105

[230] G. LAMPRINAKOS, E. KOSMATOS, D. KAKLAMANI, AND I. S. VENIERIS, *An integrated architecture for remote healthcare monitoring*, in Informatics (PCI), 2010 14th Panhellenic Conference on, IEEE, 2010, pp. 12–15. 17

[231] I. LAMPRINOS, A. PRENTZA, E. SAKKA, AND D. KOUTSOURIS, *Energy-efficient mac protocol for patient personal area networks*, in Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the, IEEE, 2006, pp. 3799–3802. 57

[232] Y.-N. LAW, H. WANG, AND C. ZANIOLO, *Query languages and data models for database sequences and data streams*, in Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30, VLDB '04, VLDB Endowment, 2004, pp. 492–503. 19, 103

[233] J. S. LEE, *Performance evaluation of ieee 802.15. 4 for low-rate wireless personal area networks*, Consumer electronics, IEEE transactions on, 52 (2006), pp. 742–749. 59

[234] J.-S. LEE AND Y.-C. HUANG, *Itri zbnode: A zigbee/ieee 802.15. 4 platform for wireless sensor networks*, in Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on, vol. 2, IEEE, 2006, pp. 1462–1467. 59

[235] J.-S. LEE, Y.-W. SU, AND C.-C. SHEN, *A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi*, in Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE, IEEE, 2007, pp. 46–51. 61

[236] A. LEFF AND J. T. RAYFIELD, *Web-application development using the model/view/controller design pattern*, in Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International, IEEE, 2001, pp. 118–127. 80

[237] P. LEGGETTER, *Real-time web technologies guide*. xx, 15, 85, 91, 94

[238] F. L. LEWIS ET AL., *Wireless sensor networks*, Smart environments: technologies, protocols, and applications, (2004), pp. 11–46. 55

[239] J.-B. LEZORAY, M. T. SEGARRA, A. P. KHAC, A. THÉPAUT, J.-M. GILLIOT, AND A. BEUGNARD, *A design process enabling adaptation and customization of services for the elderly*, in IWAAL-2010: International Workshop on Ambient Assisted Living, 2010. 12, 17

[240] J.-B. LÉZORAY, M.-T. SEGARRA, A. PHUNG-KHAC, A. THÉPAUT, J.-M. GILLIOT, AND A. BEUGNARD, *A design process enabling adaptation in pervasive heterogeneous contexts*, Personal and ubiquitous computing, 15 (2011), pp. 353–363. 12, 13

[241] Y. LI AND S. MANOHARAN, *A performance comparison of sql and nosql databases*, in Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference on, IEEE, 2013, pp. 15–19. 20, 102

[242] Y.-B. LIN AND I. CHLAMTAC, *Wireless and mobile network architectures*, John Wiley & Sons, 2008. 55

[243] Q. LIU AND X. SUN, *Research of web real-time communication based on web socket*, (2012). 84

[244] A. LO SCALZO, A. DONATINI, L. ORZELLA, A. CICCHETTI, S. PROFILI, AND A. MARESSO, *Italy: Health system review [health systems in transition] copenhagen. denmark: European observatory on health care*

systems, World Health Organization Regional Office for Europe, (2009). 50

[245] P. LOCATELLI, N. RESTIFO, L. GASTALDI, AND M. CORSO, *Health care information systems: architectural models and governance*, Innovative information systems modelling techniques, (2012), pp. 73–98. 49

[246] P. LUBBERS, B. ALBERS, F. SALIM, AND T. PYE, *Pro HTML5 programming*, Springer, 2011. 89, 90

[247] K. MA AND R. SUN, *Introducing websocket-based real-time monitoring system for remote intelligent buildings*, International Journal of Distributed Sensor Networks, (2013). 43, 44

[248] M. MAHEMOFF, *Ajax design patterns*, "O'Reilly Media, Inc.", 2006. 87

[249] A. MAINWARING, D. CULLER, J. POLASTRE, R. SZEWCZYK, AND J. ANDERSON, *Wireless sensor networks for habitat monitoring*, in Proceedings of the 1st ACM International Workshop on Wireless SensorNetworks and Applications, WSNA '02, New York, NY, USA, 2002, ACM, pp. 88–97. 8

[250] K. MALHI, S. MUKHOPADHYAY, J. SCHNEPPER, M. HAEFKE, AND H. EWALD, *A zigbee-based wearable physiological parameters monitoring system*, Sensors Journal, IEEE, 12 (2012), pp. 423–430. 47, 58

[251] R. MALL, *Real-time systems: theory and practice*, Pearson Education India, 2009. 83, 85

[252] A. MARDAN, *Boosting your node. js data with the mongoose orm library*, in Practical Node. js, Springer, 2014, pp. 149–172. 114

[253] S. MARZANO, *The new everyday: Views on ambient intelligence*, 010 Publishers, 2003. 11

[254] G. C. MAYOL RAMIS, *Design and implementation of a bidirectional, secure and real time communication between windows phone 8 app and windows store app.*, (2014). 159

[255] A. MCEWEN AND H. CASSIMALLY, *Designing the internet of things*, John Wiley & Sons, 2013. 38

[256] N. R. MEHTA, N. MEDVIDOVIC, AND S. PHADKE, *Towards a taxonomy of software connectors*, in Proceedings of the 22nd international conference on Software engineering, ACM, 2000, pp. 178–187. 14

[257] V. P. MEHTA, *Getting started with object-relational mapping*, Pro LINQ Object Relational Mapping with C# 2008, (2008), pp. 3–15. 113

[258] E. MEIJER AND G. BIERMAN, *A co-relational model of data for large shared data banks*, Communications of the ACM, 54 (2011), pp. 49–58. 101, 110, 116, 168

[259] P. MEMBREY, E. PLUGGE, AND D. HAWKINS, *The definitive guide to MongoDB: the noSQL database for cloud and desktop computing*, Apress, 2011. 44, 107

[260] M. MICHAEL, J. E. MOREIRA, D. SHILOACH, AND R. W. WISNIEWSKI, *Scale-up x scale-out: A case study using nutch/lucene*, in Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, IEEE, 2007, pp. 1–8. 100

[261] B. M. MICHELSON, *Event-driven architecture overview*, Patricia Seybold Group, 2 (2006). 75

[262] MICROSOFT, *Architectural patterns and styles*. 54

[263] ——, *Messagewebsocket class*, 2016. 93

[264] MICROSOFT.COM, *Description of the database normalization basics*, 2013. 112

[265] A. MILENKOVIĆ, C. OTTO, AND E. JOVANOV, *Wireless sensor networks for personal health monitoring: Issues and an implementation*, Computer communications, 29 (2006), pp. 2521–2533. 9, 57

[266] D. MIORANDI, S. SICARI, F. DE PELLEGRINI, AND I. CHLAMTAC, *Internet of things: Vision, applications and research challenges*, Ad Hoc Networks, 10 (2012), pp. 1497–1516. 39

[267] MONGODB, *Mongodb architecture guide 3.2-a mongodb white paper*. xx, 107, 108

[268] ——, *Mongodb crud operations*. xx, 108, 109

[269] ——, *Mongodb languages drivers*. 113

[270] ——, *Mongodb object-document mapper in java*. 113

[271] ——, *Polymorphism mongodb*. 106

[272] A. MONIRUZZAMAN AND S. A. HOSSAIN, *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison*, arXiv preprint arXiv:1307.0191, (2013). 106

[273] C. MORNINGSTAR AND F. R. FARMER, *Elko ii: Against statelessness (or, everything old is new again)*, 2009. 88

[274] M. H. MURPHY, A. M. NEVILL, E. M. MURTAGH, AND R. L. HOLDER, *The effect of walking on fitness, fatness and resting blood pressure: A meta-analysis of randomised, controlled trials*, Preventive Medicine, 44 (2007), pp. 377 – 385. 121

[275] A. MYSQL, *Mysql*, 2001. 40, 46

[276] B. O. NESBITT, *The slow load technique/reverse ajax - simulating server push in a standard web browser*, 2005. 87

[277] P. NEUBAUER, *Graph databases, nosql and neo4j*, tech. report, infoQ.com, 2010. 28, 104, 106, 107

[278] B. NEWS UK, *Ants' home search habit uncovered*, 2009. 60

[279] H. F. NIELSEN, J. GETTYS, A. BAIRD-SMITH, E. PRUD'HOMMEAUX, H. W. LIE, AND C. LILLEY, *Network performance effects of http/1.1, css1, and png*, in ACM SIGCOMM Computer Communication Review, vol. 27, ACM, 1997, pp. 155–166. 86

[280] N. NURSEITOV, M. PAULSON, R. REYNOLDS, AND C. IZURIETA, *Comparison of json and xml data interchange formats: A case study.*, Caine, 9 (2009), pp. 157–162. 148

[281] D. OBASANJO, *An exploration of object oriented database management systems*, Retrieved October, 21 (2001), p. 2016. 102

[282] A. H. OMRE AND S. KEEPING, *Bluetooth low energy: wireless connectivity for medical monitoring*, Journal of diabetes science and technology, 4 (2010), pp. 457–463. 58

[283] ORACLE, *Java message service concepts*, 2016. 74

[284] G. ORG, *Golden cheetah*. 131

[285] O. ORG, *Focus on health spending: Oecd health statistics 2015*, (2015). 6

[286] W. ORG, *About html5 websocket*, 2013. 89

[287] ——, *The benefits of websocket*, 2013. 88, 90

[288] W. H. ORGANIZATION, *Facts about ageing*, 2016. 3, 34

[289] ——, *Who disability and rehabilitation: Model disability survey (mds)*, 2016. 4

[290] ——, *Who health statistics and information systems: Who study on global ageing and adult health (sage)*, 2016. 3

[291] F. OUTSMART, *Eu project, part of the future internet private public partnership,*, OUTSMARTProvisioning of urban/regional smart services and business models enabled by the Future Internet, online at http://www. fi-ppp-outsmart.eu/enuk/Pages/default.aspx. 39

[292] E. PANIGATI, F. A. SCHREIBER, AND C. ZANIOLO, *Data streams and data stream management systems and languages*, in Data Management in Pervasive Systems, Springer, 2015, pp. 93–111. 97

[293] A. PANTELOPOULOS AND N. G. BOURBAKIS, *A survey on wearable sensor-based systems for health monitoring and prognosis*, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 40 (2010), pp. 1–12. 10, 37, 57

[294] E. PAPOUTSOGLOU, A. SAMOURKASIDIS, M.-Y. TSAI, M. DAVEY, A. INEICHEN, M. EEFTENS, AND I. N. ATHANASIADIS, *Towards an air pollution health study data management system-a case study from a smoky swiss railway*. 49

[295] S. PARIKH AND K. STIRMAN, *Schema design for time series data in mongodb*, blog. mongodb. org, 30 (2013). 49, 114, 115

[296] M. PASTERNAK, *Restful service description language*, Oct. 19 2012. US Patent App. 13/656,032. 147

[297] J. PATEL, *Cassandra data modeling best practices*, 2012. 110

[298] I. PEÑA-LÓPEZ ET AL., *Itu internet report 2005: the internet of things*, (2005). 36

[299] T. PERUMAL, S. K. DATTA, AND C. BONNET, *Iot device management framework for smart home scenarios*, in 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE), IEEE, 2015, pp. 54–55. 42

[300] T. PERUMAL, A. R. RAMLI, C. Y. LEONG, S. MANSOR, AND K. SAMSUDIN, *Interoperability among heterogeneous systems in smart home environment*, in Signal Image Technology and Internet Based Systems, 2008. SITIS '08. IEEE International Conference on, Nov 2008, pp. 177–186. 16

[301] R. PI AND S. HAT, *Raspberry pi 2 model b*, 2015. 42

[302] I. L. PIÑA, C. S. APSTEIN, G. J. BALADY, R. BELARDINELLI, B. R. CHAITMAN, B. D. DUSCHA, B. J. FLETCHER, J. L. FLEG, J. N. MYERS, AND M. J. SULLIVAN, *Exercise and heart failure a statement from the american heart association committee on exercise, rehabilitation, and prevention*, Circulation, 107 (2003), pp. 1210–1225. 121

[303] J. POKORNÝ, *New database architectures: Steps towards big data processing*, in Proc. of IADIS European Conference on Data Mining (ECDM'13), António Palma dos Reis and Ajith P. Abraham Eds., IADIS Press, 2013, pp. 3–10. 102, 104, 105, 116, 168

[304] M. P. POLAND, C. D. NUGENT, H. WANG, AND L. CHEN, *Smart home research: projects and issues*, International Journal of Ambient Computing and Intelligence (IJACI), 1 (2009), pp. 32–45. 38

[305] J. POLASTRE, J. HILL, AND D. CULLER, *Versatile low power media access for wireless sensor networks*, in Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys '04, New York, NY, USA, 2004, ACM, pp. 95–107. 27, 57

[306] R. S. PRESSMAN, *Software engineering: a practitioner's approach*, Palgrave Macmillan, 2005. 22, 30

[307] E. PRUD'HOMMEAUX, A. SEABORNE, ET AL., *Sparql query language for rdf*, W3C recommendation, 15 (2008). 106

[308] Y. PUNIE, *The future of ambient intelligence in europe: The need for more everyday life*, Communications and Strategies, 57 (2005), pp. 141–165. 10

[309] A. QUEIRÓS, A. SILVA, J. ALVARELHÃO, N. P. ROCHA, AND A. TEIXEIRA, *Usability, accessibility and ambient-assisted living: a systematic literature review*, Universal Access in the Information Society, 14 (2015), pp. 57–66. 17

[310] R. RADA, *Information systems and healthcare enterprises*, IGI Global, 2007. 50

[311] J. RADATZ, A. GERACI, AND F. KATKI, *Ieee standard glossary of software engineering terminology*, IEEE Std, 610121990 (1990), p. 3. 16

[312] S. M. RAKHUNDE, *Real time data communication over full duplex network using websocket*, IOSR Journal of Computer Science, 5 (2014), pp. 15–19. 15

[313] H. REN AND M. Q.-H. MENG, *Bioeffects control in wireless biomedical sensor networks*, in Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on, vol. 3, IEEE, 2006, pp. 896–904. 57

[314] A. RESMINI AND L. ROSATI, *A brief history of information architecture*, Journal of Information Architecture, 3 (2012). 51

[315] A. RIBEIRO, A. SILVA, A. R. DA SILVA, ET AL., *Data modeling and data analytics: A survey from a big data perspective*, Journal of Software Engineering and Applications, 8 (2015), p. 617. 101, 104

[316] J. J. RODRIGUES, *Health Information Systems: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*, vol. 1, IGI Global, 2009. 50

[317] S. ROGERS, *Health insurance datablog:healthcare spending around the world, country by country*, (30/06/2012). 6, 33

[318] K. RÖMER AND F. MATTERN, *The design space of wireless sensor networks*, Wireless Communications, IEEE, 11 (2004), pp. 54–61. 55

[319] A. RUSSELL, *Comet: Low latency data for the browser*, 2006. 87

[320] S. RUSSELL AND P. NORVIG, *Artificial Intelligence : A modern approach*, vol. 25, Citeseer, 1995. 10, 15

[321] S. SAKR, A. LIU, AND A. G. FAYOUMI, *The family of mapreduce and large-scale data processing systems*, ACM Computing Surveys (CSUR), 46 (2013), p. 11. 102, 116

[322] A. S. M. SALIH AND A. ABRAHAM, *A review of ambient intelligence assisted healthcare monitoring*, International Journal of Computer Information Systems and Industrial Management (IJCISIM), 5 (2013), pp. 741–750. 9, 34, 41

[323] B. SANTO, *Embedded*, IEEE spectrum, (2001), pp. 36–41. 83

[324] A. SANTOS, J. MACEDO, A. COSTA, AND M. J. NICOLAU, *Internet of things and smart objects for m-health monitoring and control*, Procedia Technology, 16 (2014), pp. 1351–1360. 9

[325] H. SAYUTI, R. A. RASHID, A. L. MU'AZZAH, A. HAMID, N. FISAL, M. SARIJARI, A. MOHD, K. M. YUSOF, AND R. A. RAHIM, *Lightweight priority scheduling scheme for smart home and ambient assisted living system*, International Journal of Digital Information and Wireless Communications (IJDIWC), 4 (2014), pp. 114–123. 42

[326] D. SCHIEMANN, *The forever-frame technique*, 2007. 87

[327] A. SCHMIDT AND K. VAN LAERHOVEN, *How to build smart appliances?*, Personal Communications, IEEE, 8 (2001), pp. 66–71. 128

[328] D. SCHNEIDER AND M. BECKER, *Runtime models for self-adaptation in the ambient assisted living domain*, Technical Report COMP COMP-005-2008 Lancaster University, (2008), p. 47. 16

[329] A. SEFFAH, *Patterns of HCI Design and HCI Design of Patterns: Bridging HCI Design and Model-Driven Software Engineering*, Springer, 2015. 82

[330] T. M. SELDEN AND M. SING, *The distribution of public spending for health care in the united states, 2002*, Health Affairs, 27 (2008), pp. w349–w359. 6

[331] SELFLOOPS, *Selfloops: Heart rate variability*. 131

[332] R. SELIGER, *Overview of hl7's ccow standard*, Health Level Seven, Inc. URL: http://www.hl7.org/library/committees/sigvi/ccow_overview_2001.doc, (2001). 185

[333] R. SESSIONS, *Comparison of the top four enterprise architecture methodologies*, (2007). 17

[334] Z. SHELBY AND C. BORMANN, *6LoWPAN: The wireless embedded Internet*, vol. 43, John Wiley & Sons, 2011. 43

[335] K. G. SHIN AND P. RAMANATHAN, *Real-time computing: A new discipline of computer science and engineering*, Proceedings of the IEEE, 82 (1994), pp. 6–24. 19, 143

[336] E. M. SIMONSICK, P. S. MONTGOMERY, A. B. NEWMAN, D. C. BAUER, AND T. HARRIS, *Measuring fitness in healthy older adults: the health abc long distance corridor walk*, Journal of the American Geriatrics Society, 49 (2001), pp. 1544–1548. 121

[337] D. SNOW, *Adding a 4th v to big data - veracity*. 104

[338] R. SNYDERMAN, *Personalized health care: From theory to practice*, Biotechnology journal, 7 (2012), pp. 973–979. 36

[339] M. SOINI, J. NUMMELA, P. OKSA, L. UKKONEN, AND L. SYDNHEIMO, *Wireless body area network for hip rehabilitation system*, Ubiquitous Computing and Communication Journal 3.5, (2008). 46

[340] S. SOLAIMANI, W. KEIJZER-BROERS, AND H. BOUWMAN, *What we do–and dont–know about the smart home: an analysis of the smart home literature*, Indoor and Built Environment, (2013), p. 1420326X13516350. 39

[341] Z. SPECIFICATION, *Zigbee alliance*, URL: http://www.zigbee.org, 558 (2006). 40, 59

[342] SPORTPLUSHEALTH, *Sport plus health*. 131

[343] STACKOVERFLOW.COM, *How to decide when to use node.js?* 95

[344] R. STEINMETZ AND K. NAHRSTEDT, *Multimedia systems*, Springer Science & Business Media, 2013. 74

[345] M. STONEBRAKER, U. ÇETINTEMEL, AND S. ZDONIK, *The 8 requirements of real-time stream processing*, ACM SIGMOD Record, 34 (2005), pp. 42–47. xx, 84, 97, 98, 99

[346] M. STRASSER, E. HELM, A. SCHULER, M. FUSCHLBERGER, AND B. ALTENDORFER, *Mobile access to healthcare monitoring data for patients and medical personnel*, in 24th International Conference of the European Federation for Medical Informatics Quality of Life through Quality of Information, 2012. 46, 48

[347] C. STRAUCH, *Nosql whitepaper*, tech. report, Stuttgart: Hochschule der Medien, 2014. 104, 106, 107

[348] I. STRAVA, *Strava: The social network for athletes*. 131

[349] C. STROZZI, *Nosql-a relational database management system*, Lainattu, 5 (1998), p. 2014. 104

[350] O. SUN MICROSYSTEMS, *Java.net class socket*, 2003. 87

[351] T. TALEB, D. BOTTAZZI, M. GUIZANI, AND H. NAIT-CHARIF, *Angelah: a framework for assisting elders at home*, Selected Areas in Communications, IEEE Journal on, 27 (2009), pp. 480–494. 48

[352] P.-N. TAN, M. STEINBACH, AND V. KUMAR, *Introduction to Data Mining, (First Edition)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. 161

[353] P.-N. TAN, M. STEINBACH, V. KUMAR, ET AL., *Introduction to data mining*, vol. 1, Pearson Addison Wesley Boston, 2006. 10, 40

[354] A. S. TANENBAUM AND M. VAN STEEN, *Distributed systems: principles and paradigms*, vol. 2, Prentice hall Englewood Cliffs, 2002. 12, 15, 28, 29, 41, 70, 71, 85, 92, 140, 142, 144, 145, 146, 147

[355] D. I. TAPIA, A. ABRAHAM, J. M. CORCHADO, AND R. S. ALONSO, *Agents and ambient intelligence: case studies*, Journal of Ambient Intelligence and Humanized Computing, 1 (2010), pp. 85–93. 34

[356] A. TECHNICA AND WIRED, *Http/2 finished, coming to browsers within weeks*, 2015. 86

[357] THISISANT, *Thisisant: The facts*. 120

[358] ——, *Thisisant: the wireless sensor network solution*. xx, 58, 60, 61, 62, 190

[359] ——, *Ant+ common pages, revision 2.4*, online doc, Dynastream Innovations Inc., Jul 2007. 128, 154, 166, 168

[360] ——, *Ant file share (ant-fs) technical specification*, online doc, Dynastream Innovations Inc., January 2012. 64, 124

[361] ——, *Ant device pairing : Application notes*, tech. report, Dynastream Innovations Inc., Mar 2013. 63

[362] ——, *Fit file types : description, revision 1.6*, online doc, Dynastream Innovations Inc., August 2014. 64

[363] ——, *Fit sdk: Introductory guide, revision 1.6*, online doc, Dynastream Innovations Inc., August 2014. 64, 131

[364] ——, *Flexible and interoperable data transfer (fit) protocol, revision 1.7*, online doc, Dynastream Innovations Inc., August 2014. xx, 63, 64, 124, 180

[365] ——, *Garmin foot pod*, 2016. 123

[366] ——, *Garmin tempe wireless temperature sensor*, 2016. 123

[367] ——, *Top brands. compatible devices.*, Apr 2016. xxi, 122, 123, 129, 153

[368] J. TIDWELL, *A pattern language for human-computer interface design*, Available via DIALOG, (1997). 80

[369] S. TILKOV AND S. VINOSKI, *Node. js: Using javascript to build high-performance network programs*, IEEE Internet Computing, (2010), pp. 80–83. 95

[370] K. J. TURNER, *Flexible management of smart homes*, Journal of Ambient Intelligence and Smart Environments, 3 (2011), pp. 83–109. 55

[371] O. UML, *2.0 superstructure specification*, OMG, Needham, (2004). 14

[372] E. I. UNIT, *The future of healthcare in europe*, A report from the Economist Intelligence Unit sponsored by Janssen. Londres: The Economist Intelligence Unit Limited, (2011). 6

[373] E. VALCHINOV, A. ANTONIOU, K. ROTAS, AND N. PALLIKARAKIS, *Wearable ecg system for health and sports monitoring*, in Wireless Mobile Communication and Healthcare (Mobihealth), 2014 EAI 4th International Conference on, Nov 2014, pp. 63–66. 46

[374] J. VAN DEN BROECKE, *Pushlets-whitepaper*, 20160516. http://www.pushlets.com/doc/whitepaper.html, 6 (2002). 87

[375] O. VERMESAN AND P. FRIESS, *Internet of things: converging technologies for smart environments and integrated ecosystems*, River Publishers, 2013. 18, 39

[376] G. VIRONE AND A. SIXSMITH, *Toward information systems for ambient assisted living*, in The 6th International Conference of the International Society for Gerontechnology, Pisa, Italy, 2008, pp. 4–7. 15

[377] W-TEK, *Hs-2 plus*, 2016. 123

[378] ——, *W-tek wearing technology*, 2016. 123

[379] C. WAIDE, *Object-oriented database*, Retrieved July, 21 (2003), p. 2016. 102

[380] A. WALKER AND T. MALTBY, *Active ageing: a strategic policy solution to demographic ageing in the european union*, International Journal of Social Welfare, 21 (2012), pp. S117–S130. 6, 33

[381] J. WAN, M. CHEN, F. XIA, D. LI, AND K. ZHOU, *From machine-to-machine communications towards cyber-physical systems.*, Comput. Sci. Inf. Syst., 10 (2013), pp. 1105–1128. 16

[382] R. WANT, *An introduction to rfid technology*, Pervasive Computing, IEEE, 5 (2006), pp. 25–33. 58, 59

[383] Z. WEI-PING, L. MING-XIN, AND C. HUAN, *Using mongodb to implement textbook management system instead of mysql*, in Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, IEEE, 2011, pp. 303–305. 110

[384] M. WEISER, *Some computer science issues in ubiquitous computing*, Commun. ACM, 36 (1993), pp. 75–84. 11

[385] M. WEISER, R. GOLD, AND J. S. BROWN, *The origins of ubiquitous computing research at parc in the late 1980s*, IBM systems journal, 38 (1999), p. 693. 36

[386] I. H. WITTEN AND E. FRANK, *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2005. 10, 40

[387] A. WOLLRATH, R. RIGGS, AND J. WALDO, *A distributed object model for the javaˆ tˆ m system*, Computing Systems, 9 (1996), pp. 265–290. xx, 72, 73

[388] C.-L. WU, C.-F. LIAO, AND L.-C. FU, *Service-oriented smart-home architecture based on osgi and mobile-agent technology*, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 37 (2007), pp. 193–205. 54

[389] W. YE, J. HEIDEMANN, AND D. ESTRIN, *An energy-efficient mac protocol for wireless sensor networks*, in Proceedings of the IEEE Infocom:Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, New York, NY, USA, June 2002, USC/Information Sciences Institute, IEEE, pp. 1567–1576. 27, 57

[390] W. YE, J. HEIDEMANN, AND D. ESTRIN, *Medium access control with coordinated adaptive sleeping for wireless sensor networks*, IEEE/ACM Transactions on Networking, 12 (2004), pp. 493–506. 57

[391] G. M. YOUNGBLOOD, D. J. COOK, AND L. B. HOLDER, *Managing adaptive versatile environments*, Pervasive and Mobile Computing, 1 (2005), pp. 373–403. 43

[392] A. ZANELLA, N. BUI, A. CASTELLANI, L. VANGELISTA, AND M. ZORZI, *Internet of things for smart cities*, Internet of Things Journal, IEEE, 1 (2014), pp. 22–32. 34, 36, 39

[393] P. ZH, *Ea methodology vs. ea framework*, 2013-2020. 17

[394] W. ZHANG, R. STOLL, N. STOLL, AND K. THUROW, *An mhealth monitoring system for telemedicine based on websocket wireless communication*, Journal of Networks, 8 (2013), pp. 955–962. 47

[395] W. ZHANG, K. THUROW, AND R. STOLL, *A context-aware mhealth system for online physiological monitoring in remote healthcare*, International Journal of Computers Communications & Control, 11 (2015), pp. 142–156. 47

[396] Y. ZHANG AND H. XIAO, *Bluetooth-based sensor networks for remotely monitoring the physiological signals of a patient*, Information Technology in Biomedicine, IEEE Transactions on, 13 (2009), pp. 1040–1048. 47, 58

[397] T. G. ZIMMERMAN, *Personal area networks: near-field intrabody communication*, IBM systems Journal, 35 (1996), pp. 609–617. 56

Appendix A: Self Declaration

I hereby declare that I have prepared my present work independently and only using lawful means.

Camerino, October 22, 2016

Nadeem Qaisar Mehmood

Appendix B: ANT+ Protocol

.1 ANT Protocol

.1.1 ANT+ Network Configuration

You read in the previous chapters (such as in 2.8.3) about the implementation of the a simple ANT+ network. Here we will discuss how to build an ANT network using the independent channels.

This is the simplest way to implement the networks. For the simple network we depicted in section 2.8.3, we need three independent channels. Following is the discussion step by step about each of the individual channel, initial configuration detail. For more details please refer to section 5.2 and 5.3 in the *ANT Message Protocol and Usage document* [140] provided by the ANT Alliance. The configuration details for the "*Channel between B(master) and A(slave)-Broadcast Data type*" are presented in the following table 1.

Table 1: Channel between B(master) and A(slave)-Broadcast Data type

Channel Parameter	**B**	**A**	**Description**
Network Number	0	0	Default Public Network
RF Frequency	66	66	Default Frequency 2466MHz
Device Number	1	1	Serial Number of Node B
Transmission Type	1	1	Transmission Type (no shared address)
Device Type	1	1	Device Type of Node B
Channel Type	0x10	0x00	For B-Bidirectional Transmit channel, For A-Bidirectional Receive Channel
Channel Period	8192	8192	Default 4Hz Message Rate
Data Type	0x4E	0x4E	Broadcast

Similarly, the configuration details for the "*Channel between C(master) and A(slave)-Acknowledge Data type*" are presented in the following table 2.

Table 2: Channel between B(master) and A(slave)-Broadcast Data type

Channel Parameter	C	A	Description
Network Number	0	0	Default Public Network
RF Frequency	66	66	Default Frequency 2466MHz
Device Number	10	1	Serial Number of Node C
Transmission Type	1	1	Transmission Type (no shared address)
Device Type	2	1	Device Type of Node C
Channel Type	0x10	0x00	For C-Bidirectional Transmit channel, For A-Bidirectional Receive Channel
Channel Period	8192	8192	Default 4Hz Message Rate
Data Type	0x4F	0x4E	Acknowledged

Similarly, the configuration details for the "*Channel between D(master) and A(slave), Broadcast Data type*" will be the same as between B and A, see table 1. Only the difference is in the Device number, which will be 2 here.

.1.2 Establishing an ANT+ Network given the Configurations

There is a precised sequence of events and message transactions between the host and ANT device in order to establish a channel between the two nodes. Also see section 5.3 in the *ANT Message Protocol and Usage document* [140]. The prerequisite to build a channel is that the master and the slave both should have the common knowledge of the channel configuration details as we discussed already. The figure 1 illustrates the steps required to follow to establish a channel between a master and slave node. The steps are as:

- Both nodes must have the pre common knowledge about the channel.
- A **Network Key** is desired if you want a managed network, otherwise for a public network one can use the default network key. This network key will be assigned to the channel.
- Then **Channel Type** needs to set for the channel one wants to open.

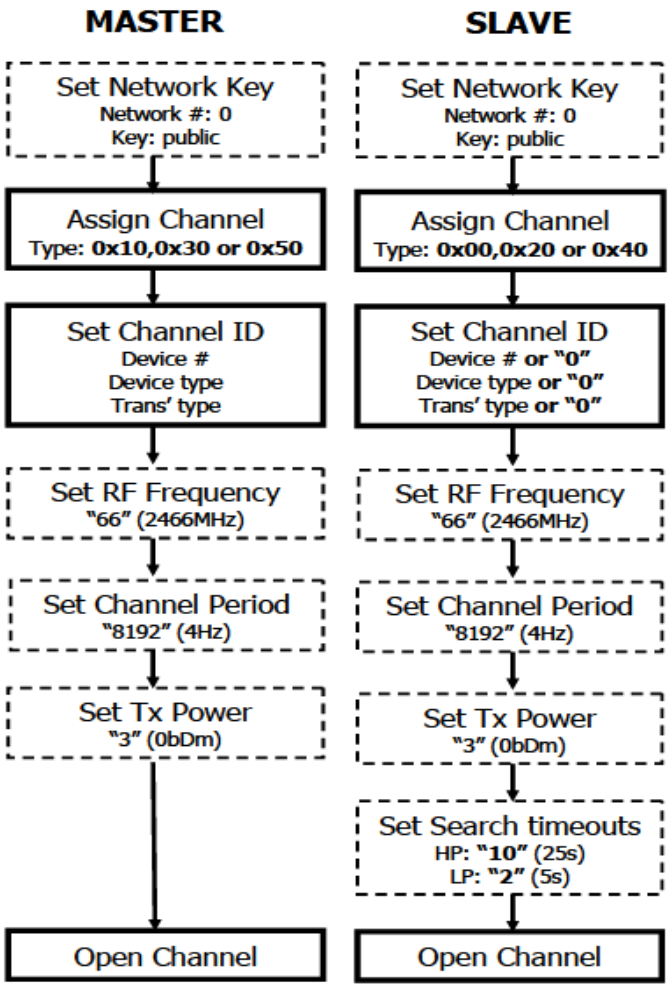


Figure 1: Process to establish master-slave ANT+ channel [140]

- Channel parameters such as RF frequency, Channel period, Tx power and search timeouts can also be set but are not required in order to establish the channel. As shown in the figure 1.

- Finally open the channel.
  - The master establishes the channel by transmitting 8-byte data packets in the designated timeslot at the established message rate. The masters host controller will optionally provide new data to the ANT engine for continuing transmissions.
  - The slave on the other hand, will immediately start searching for a master that matches the channel ID.

- There are sequence of events and message transactions between the host and ANT for each participating node.

- For the default the host uses the following minimum commands.
  - ANT_AssignChannel()
  - ANT_SetChannelId()
  - It should be noted that the channel numbers do not need to match on either side of the channel
  - ANT_OpenChannel()
  - Once opened, the masters host provides using the ANT_SendBroadcastData()
  - After slave's channel openning; ANT informs the host with a ChannelEventFunc() message, when a message is received from Node B

- To establish a channel for *B(master) and A(slave)-Broadcast Data type*, everything is same as above except the following differences
  - The channel id may vary as the channel numbers do not need to match on either side of the channel
  - The master's host provides ANT with data using the ANT_SendAcknowledgedData()

Appendix C: ANT+ XML Configs

```
0 <?xml version="1.0" encoding="utf-8"?>
1 <ClientSettings>
2  <Version type="System.String">APS_01</Version>
3  <NodeServer>
4   <nodeurl>http://193.205.92.159:</nodeurl>
5   <nodeport>3002</nodeport>
6  </NodeServer>
7  <GeneralSettings>
8    <appname>ANTPusherService</appname>
9   <IsDebugEnabled type="System.String">true</IsDebugEnabled>
10    <logPathSelection type="ANTPusherService.Settings+LogDirectory">
11     AppData
12    </logPathSelection>
13    <logPath type="System.String">
14     C:\Users\user\AppData\Roaming\ANTPusherService\logs
15    </logPath>
16    <isAutoInitEnabled>true</isAutoInitEnabled>
17    <porttype type="ANTPusherService.Settings+USBporttype"> USB
18    </porttype>
19    <usbDeviceno type="System.Byte">0</usbDeviceno>
20    <deviceType type="System.Byte">1</deviceType>
21   <transmissionType type="System.Byte">1</transmissionType>
22    <network type="ANTPusherService.Settings+Network">AntPlus
23    </network>
24    <broadcastChannelPeriod type="System.UInt16">8192
25    </broadcastChannelPeriod>
26    <maxBlockSize type="System.UInt32">0</maxBlockSize>
27    <friendlyName type="System.String">ANTPusherService</friendlyName>
28    <passKey type="System.Byte[]"> 0xxx,0xxx,0xxx,0xxx,0xxx,0xxx,0xc3,0x45
29 </passKey>
30  </GeneralSettings>
31 </ClientSettings>
```

Appendix D: Node Context Server

.2 JSON Config for the ANT Node Context Server

```
{"name": "SocketIO4Net-Test-Server",
  "author": "J. Stott", "version": "0.0.3",
  "description":"Server to demonstrate SocketIO4Net.Client connectivity",
  "homepage": "http://socketio4net.codeplex.com/",
  "main": "server.js",
  "dependencies": {
    "body-parser": "^1.14.1",    "debug": "~0.7.2",
    "email-validator": "~0.1.2", "express": "~3.4.4",
    "express-mongoose": "^0.1.0","moment": "^2.10.6",
    "mongoose": "^4.2.10",      "socket.io": "~0.9.16",
    "socket.io-wildcard":"~0.1.1"},
  "devDependencies": {
   "grunt": "~0.4.1", "grunt-contrib-jshint": "~0.6.4"}
}
```

.3 MongoDB Shell

After Installing MongoDB Libraries[20], the following command starts MongoDB server to store data files in the journal directory location using the –dbpath option. The –port option can be used to tell the port number as an argument, upon which the server listens the connections.

`>mongod --dbpath=C:\mongodata --port 27018`

Using anohter command prompt shell, the `>mongo` command will start accessing the server, and will display the version and shall connect with the *test* database automatically. If the server is running on another port than the default port (e.g.,`27017`), then server's HTTP IP address and the specific port can be passed as an argument. For example, `localhost:27018.` The `show dbs` command will display all the databases at the server. The `use` *`databasename`* command, will select a specific database; and the `show collections` command will show all the collections within that database. The `db.mycollectionname.find()` command will fetch all the documents from the collection, as in this case `mycollectionname.`

[20] www.mongodb.com/download-center

.4 Heart Rate Monitor Temporal Document

Figure 2: MongoDB Temporal Storage Document View

The Figure 2 shows the view of a temporal heart rate document for a simple query. The *use healthcare* tells the database to select healthcare database for operations. The *show collections* asks MongoDB to show the collections (or tables in RDBMS); which tells that, there are three collections, i.e. *pulses*, *footpod*, and *temps*. A collections containing the indexing information is also displayed. The query which fetches the temporal documents of the heart rate monitor from the database is as: $>$ *db.pulses.find().pretty()*. Note that the database has saved the heart rate monitor messages as doc-per-hour schema defined using Mongoose.