Federal University of Minas Gerais
Department of Computer Science

Data Fusion Implementation in Sensor Networks Applied to Health Monitoring

by

Hervaldo Sampaio Carvalho MSc, MD

Advisor: Claudionor José Nunes Coelho Júnior

Co-Advisor: Wendi Heinzelman

Thesis defense as a Partial fulfillment
of the requirements for the degree of

Doctor of Science in Computer Science

at the Department of Computer Science
Federal University of Minas Gerais

January, 2005

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Carvalho, Hervaldo Sampaio

C331d Data fusion implementation in sensor networks applied to health monitoring / Hervaldo Sampaio Carvalho – Belo Horizonte, 2005.
145 f.: il.; 29 cm.

Tese (doutorado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Claudionor José Nunes Coelho Júnior
Coorientador: Wendi Heinzelman

1. Computação – Teses. 2. Arquitetura de computador.
3. Sistemas operacionais distribuídos (Computadores).
4. Redes de sensores sem fio. I. Orientador. II. Título.

CDU 519.6*22(043)

Thesis Supervisor

Claudionor José Nunes Coelho Jr
Adjunt Professor of Computer Science
Federal University of Minas Gerais,
Belo Horizonte (MG), Brazil

Thesis Supervisor

Wendi Heinzelman
Assistant Professor of Electrical and Computer Engineering
University of Rochester
Rochester (New York), USA

UNIVERSIDADE FEDERAL DE MINAS GERAIS

PPGCC
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DCC-UFMG

FOLHA DE APROVAÇÃO

Fusão de Dados para Rede de Sensores Aplicada à Monitoração Pessoal de Saúde

HERVALDO SAMPAIO CARVALHO

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. CLAUDIONOR JOSÉ NUNES COELHO JÚNIOR - Orientador
Departamento de Ciência da Computação - UFMG

Prof. DJAMEL FAWZI HADJ SADOK
Departamento de Informática - UFPE

Prof. RICARDO DAHAB
Instituto de Computação - UNICAMP

Prof. ANTONIO ALFREDO FERREIRA LOUREIRO
Departamento de Ciência da Computação - UFMG

Prof. ANTÔNIO OCTÁVIO FERNANDES
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 22 de março de 2005.

Acknowledgements

I would like to thank Professor Claudionor J. N. Coelho Júnior, my thesis advisor and mentor for the last four years. Professor Claudionor is one of these bright minds that always has a good point to improve an idea or to find a solution for our problems. He has taught me that the world of science is more than papers and research, patents and products are also included in this world. He has taught me that behind a student there is a human being part of a family and society. It has being a privilege to share my dreams and work with him.

I would like to thank Professor Wendi Heinzelman, my thesis co-advisor and my mentor during the time I spent as a visiting faculty at the Center For Future Health at the University of Rochester (NY-USA). Professor Wendi has been a wonderful advisor, I would say an essential person to the conclusion of my thesis. I thank her for the great number of hours she spent discussing and critiquing my papers and ideas. I´m really greateful for all the support she gave me.

I began my PhD program with Professor Berthier Ribeiro-Neto as my co-advisor and I finished my degree with a friend. Professor Berthier and his wife were very important to me by the time I lived in Belo Horizonte. He helped me to face day by day the hard times I had in the first two years.

I also would like to thank Professor Amy Murphy. I thank her for all the time she spent discussing and critiquing my papers and ideas. I would like to thank her for her friendship.

I would like to thank the opportunity and privilege to study at the Department of Computer Science at the Federal University of Minas Gerais. There, I found an auspicious and outstanding environment to learn and improve my knowledge. At this department, I had the opportunity to study with Professors Antonio Alfredo F. Loureiro, Antônio Otávio Fernandes**,** Clarindo Isaías Pereira da Silva e Pádua**,** Geraldo Robson Mateus**,** Henrique Pacca L. Luna , José Marcos Silva Nogueira**,** Mário Fernando Montenegro Campos**,** Newton José Vieira**,** Nívio Ziviani, Rodolfo Sérgio Ferreira de Resende**,** Sérgio Vale Aguiar Campos**,** some of the best Professors of Computer Science in Brazil.

I would like to thank some of the new friends I got at Belo Horizonte. I had the opportunity to share my life and my studies with some of the most talented and friendly students: Linnyer, Camillo, Pável, Júlio Conway, Gilberto, Autran, Ilmério, Maria, Cristiane, and Paulo.

While at Rochester, Mark Perillo (a bright PhD student at UR) made a big difference to my life and my learning process in Rochester. Mark, I really enjoyed your friendship and support.

I would like to thank Professor Alice P. Pentland, Professor Philippe M. Fauchet, Cecelia Horwitz, and Dolores Christensen for their support at the Center For Future Health.

I would like to thank my wife Marly and my sister-in-law Edna for their support and for the correction of the thesis format.

I would like to thank my sisters Cláudia and Jane for their support.

I would like to thank and to ask for excuses to my children Ana Luisa and Carolina. I have been far away from them and I lost very important moments of their lives and I wonder some day, they can understand my reasons.

I would like to thank my mother and my father. They have always given me support and love in all moments of my life. Everything I have done and I have been is the consequence of the infinite love they have given me.

Sometimes we believe in things that a few people believe
Sometimes we do things that a few people can do

Hervaldo.

Abstract

Recent developments in wireless networks and in miniaturization of powerful embedded devices have enabled the development of very small computing systems that are available all the time. In the literature, this type of computation has been called ubiquitous computing. Several applications of ubiquitous computing (including ones that cover life threatening situations) require fault tolerance, resilience and graceful degradation in response to different types of failures in the system. Several authors have focused on the development of middleware solutions to ease the design of ubiquitous computing applications. Others have addressed the application development field, but very few authors have addressed the relationship between middleware and application development.

Data fusion is an important component of applications for systems that use correlated data from multiple sources to determine the state of a system. The fault tolerance and resilience of these applications will depend greatly on the data fusion framework. As the state of the system being monitored and available resources change, the general data fusion framework should change dynamically based on the current environment and available resources in the system. As a consequence, a general data fusion framework should provide some results of the data fusion to a module called the decision system. This module is responsible for sending feedback to the middleware, so the middleware can appropriately reconfigure the network. Based on the current data or variable, the decision system receives from the data fusion module, the decision system should automatically inform the middleware of the application's new requirements (i.e., the application dynamically adjusts its Quality of Service requirements based on the current state of the system being monitored and informs the middleware of its current Quality of Service needs).

In this thesis we address the problem of how to implement data fusion in sensor networks, taking into account fault tolerance, resilience, and graceful degradation in a ubiquitous computing environment. We think that to achieve these goals it is necessary to develop applications upon a dynamic data fusion architecture. To achieve this goal we have created a new data fusion architecture; developed a software infrastructure based on this architecture and applied to centralized and distributed implementations; and developed a communication approach between middleware and the application. All these tools are new in the literature and represent important contributions to the data fusion implementation in sensor networks field. Furthermore, the combination of these tools represent an important contribution to sensor networks applications development.

As a proof of concept, we have developed a Personal Multi Parametric Heart Rate Monitor application based on sensor networks conception. The Personal Heart Rate Monitor consists of a body-worn sensor networks application powered by battery and connected by a wireless network. Therefore, resources such as channel

bandwidth and node energy are limited, and must be managed efficiently. The developed system is based on the proposed tools to implement data fusion in sensor networks applications which dynamically adapts as the state of the person's vital signals change and provide graceful degradation to resource changes. The Personal Multi Parametric Heart Rate Monitor developed demonstrated to be an important contribution to the medical field. It will be tested in a clinical trial to evaluate its impact in prevention and early diagnosis of diseases.

Resumo

Nos últimos anos o grande desenvolvimento dos sistemas de comunicação e a miniaturização e evolução tecnológica dos sistemas de hardware permitiram o desenvolvimento de novas aplicações. A área de redes de sensores é uma destas novas aplicações. As aplicações em rede de sensores se caracterizam pela presença de inúmeros sensores (nodos) de uma rede de comunicação sem fio, com limite de alcance do sinal e limitação de fonte de energia, pôr serem operados a bateria. Estes sensores têm capacidade de detectar ou medir algum fenômeno da natureza, processar e transmitir os dados ou a informação para outros sensores (nodos) até chegar em um ponto desta rede de tomada de decisão. Trata-se então primordialmente de um sistema distribuído com sérias restrições para implementação.

O desenvolvimento destas aplicações exige os desenvolvimento de técnicas de tolerância à falhas e adaptação a novas condições ambientais com a finalidade de que o tempo de vida do sistema seja o mais longo possível. Diversos autores têm trabalhado no desenvolvimento de sistemas que suportem estas características. Alguns trabalhos estão na área de "middleware" e outros na área de desenvolvimento de aplicações. Poucos autores têm se preocupado com a interface aplicação-"middleware".

Nos últimos anos a área de fusão de dados também tem tido um grande crescimento pelas novas exigências das aplicações que estão sendo criadas e pelo aprimoramento e desenvolvimento de novas técnicas estatísticas e de inteligência artificial. Entretanto, pouco tem sido feito na área de arquitetura de fusão de dados, e na sua implementação em redes de sensores. Além disto, a interface aplicação - middleware nos parece ser altamente dependente da implementação de fusão de dados e de sua arquitetura.

O presente trabalho se propõe a estudar o problema de como implementar fusão de dados em rede de sensores, levando em consideração a adaptação à falhas e mudanças no ambiente monitorado. Estes aspectos são determinantes para que o sistema possa degradar progressivamente, mas mantendo as necessidades exigidas pela aplicação (qualidade de serviço). Neste sentido, o presente trabalho propõe uma nova arquitetura de fusão de dados, dinâmica e adaptável a diferentes sistemas, sejam eles distribuídos ou centralizados e independente de contexto. Além disto, o desenvolvimento de um software para implementação centralizada e distribuída desta arquitetura associada ao desenvolvimento de uma linguagem para comunicação aplicação -"middleware" pôr meio da fusão de dados, vem completar o que achamos ser necessário para o desenho e desenvolvimento de uma aplicação em rede de sensores. Todas estas ferramentas também são importantes contribuições para a área de fusão de dados e redes de sensores.

Como prova de conceito, desenvolvemos uma aplicação (protótipo) centralizada de um monitor multiparamétrico móvel para monitoração da freqüência cardíaca e simulamos uma implementação de fusão de dados em rede de sensores distribuída no corpo humana. Esta aplicação demonstrou ser uma importante contribuição na área de diagnóstico precoce e prevenção de enfermidades cardiovasculares.

Contents

List of Figures

List of tables

Introduction

Sensor Networks Applications Development

A sensor networks application is characterized by a set of battery operated nodes (sensing units) linked by a network (usually wireless) to the decision system, also battery operated nodes (decision units). In some applications, both types of functionalities are executed by the same nodes. Sensor networks application needs a complex specification to achieve fault tolerance and resilience to its different functionalities in a ubiquitous computing environment. To achieve this goal, different aspects should be considered at the different levels: sensors, communication, application, data management, power management and ethical aspects:

I. Sensors
   A. Multiple types of sensors
   B. Redundant and non redundant sensors
   C. Redundant and non redundant sensors' functionalities
   D. Mobile and static sensors
   E. Battery operated sensors
   F. Wired and wireless sensors
   G. Smart or simple sensors
   H. Localization

II. Communication
   A. Wireless communication
   B. Wired communication
   C. Low power compatible protocols
   D. Security

III. Application
   A. Application development
      1. Centralized
      2. Distributed
   B. Different needs of the application (Quality of Service – QoS)
      1. Different coverage area
      2. Data metrics (Ex: accuracy and reliability)
      3. Set of sensors
   C. Multiple applications running simultaneously
      1. Transactions scheduling
      2. Applications' priorities

IV. Data management
   A. Data processing (DSP algorithms)
   B. Data Fusion algorithms
      1. Data reliability

2. Data accuracy
3. Different levels of data fusion
4. Data fusion optimal node

C. Real time or not real time
D. Data analysis
E. Decision node
F. Synchronization
G. Positioning
H. Security

V. Power management

A. Cost of communication
B. Cost of sensing
C. Cost of routing
D. Cost of processing
E. Sensor's lifetime
F. System's lifetime

VI. Ethical aspects

Most of these aspects have been addressed individually by different authors (Sensor networks review articles), but a few articles have tried to address some of these different aspects collectively. The decisions taken for each item cannot be made independently of the system as a whole. One decision interferes in one or more item, and this interference must be considered to achieve the best performance of the system. Based on this, to achieve the goal of developing a real sensor networks application, different solutions should be applied.

Multiple types of sensors can be deployed with or without restrictions. In a forest monitoring application the sensors are deployed in the environment from helicopters or planes. They can fall anywhere. In a body-worn sensor network, the body imposes restrictions for the sensors' deployment, and most of the time it should be deployed manually. These applications are also suitable to compare the type of sensor used. In the forest monitoring application it is necessary to use smart sensors (sensors with sensing, processing, memory and wireless communication capability). In the body-worn application the sensors over the skin could be accessed easily and can also be wired. This difference allows the use of simple sensors or smart sensors, depending on the objective desired. As a consequence, the communication network can be wired or wireless. In the forest monitoring application, the communication network should be wireless. The use of smart sensors and wireless communication imposes restrictions related to the wireless communication complexity and the limited power resource of the sensors.

In both types of applications it is desired to adapt to failures and different circumstances of the environment (resilience). There are two main options to achieve these goals. The first one is to employ redundant sensors in the application, so if one or more sensors fails, the redundant sensors can maintain the functionality for which they are responsible. The second option is to employ redundant sensors functionalities. In this case it is not necessary to use the same type of sensors, but sensors that have one or more functionalities in common. The second option is much better, because we can offer the

same fault tolerance and resilience specifications with a lower number of sensors, but they must be smart sensors that can be configured according to necessity.

Another aspect related to the sensors that can impose great complexity for the system is the employment of mobile sensors. Although it is not common to employ mobile sensors in the forest application, it would be desirable to use some mobile sensors to solve some of the problems related to the random deployment (to achieve the desired coverage area, to link isolated sensors that are out from the network coverage area, etc). In the body-worn sensor networks application it is expected that mobile sensors will be employed to monitor the entire body using the blood flux or the gastrointestinal tract for example. In this case, the network topology changes all the time. This aspect increases the management complexity of the system.

Sensor networks applications have some characteristics that may contribute to the large amount of complexity. For example, each node can only gather data from a limited physical area of the environment, data may be noisy, distributed processing of large amounts of sensor data, scalability, quality can be traded for system lifetime, and "Team-work" (nodes can help each perform a larger sensing task).

A common problem related to the sensor is positioning. It is important for same reasons that time synchronization is important. There are many different approaches in the literature to solving this problem, such as GPS. But GPS is overkill and an unattractive solution for energy reasons. Besides GPS, the sensor can find its own location by finding the center of mass of several beacon nodes that are heard, or finding its location through multiple measurements of angles of arrival of beacon nodes. The sensor can also have other nodes measure its location. The problem with these approaches is the small scale fading.

The wireless network, by itself, is a great problem because the transmission media (air) is shared by all the nodes in the same coverage area. Problems such as collisions, congestion, and security are an open field of research. The sensors networks require a new wireless networking paradigm, characterized by autonomous operation, highly dynamic environments, and sensor nodes added/fail. The bandwidth is limited, and must be shared among all the nodes in the sensor network. Spatial reuse is essential and efficient local use of bandwidth is needed, with no end-to-end communication, redundancy in information, or events in the environment. As a consequence, distributed computation and communication protocols require energy-conserving communication (communication is the most energy-intensive). Based on these aspects, the protocols requirements include: low power (to maximize battery lifetime), low latency (to maximize quality), self-configuring (with no fixed infrastructure), ad hoc (nodes moving, environment changing), multi-hop (ensuring nodes can all communicate), and scalable (to accommodate varying numbers of nodes).

Some authors have employed adaptive protocols exploiting high node density, turning off some sensors for long periods of time. The tradeoff must be balanced. Having many active nodes can be wasted energy due to idle power and having many sleeping nodes provides fewer routes to choose from and so route selection will likely be suboptimal.

In sensor networks routing energy-efficiency is even more important than in ad-hoc protocols. This is the reason why lightweight protocols (little overhead) are preferable. Most ad hoc routing protocols use algorithms with the fewest hops. Power/energy-aware

routing may use different metrics, reduce power consumption, distribute energy load (maximizing network lifetime), may be tightly coupled with protocols from different layers, and take advantage of data fusion opportunities and overlaps with adaptive topology protocols.

There are a great number of papers addressing application's development in general, but few papers have addressed sensor networks application's development. What makes it different from the usual applications from the distributed systems area is the wireless communication and the battery operated sensors. The following table 1 shows some of the differences between traditional applications and sensor networks applications.

Table 1: comparison between traditional applications and sensor networks applications

Traditional applications	Sensor networks applications
Users can update and maintain devices	May be impossible to update or maintain sensor nodes, due to sheer numbers as well as deployment locations
Offer communication between two specific end-users	Communication is data-centric: end-user does not care that the data came from node X, only what the data describes
Goal: providing high QoS bandwidth efficiency	Goal: prolonging lifetime of the network – Requires energy conservation – Willing to give up performance in terms of QoS or bandwidth efficiency
Data are important	End user does not require all the data – Data from neighboring nodes are highly correlated, making the data redundant – End user typically cares about a higher-level description of events occurring in the environment nodes are monitoring – Network quality is often based on quality of aggregate data set rather than individual signals
Intermediate nodes do not care what the data are	Application-specific routing improves performance
Nodes are operating (mostly) independently	Sensor network application computation - May need to be distributed throughout network (e.g., localized algorithms that achieve desired global result) - May require hierarchical structure - Enables computation / communication tradeoff - Has three processing levels: node, local, and global
Operate in (mostly) benign environments	May be deployed in hostile or dangerous territory

We can develop centralized and distributed sensor networks applications. The distributed ones, in general, are more complex than the centralized. As we have mentioned before, these aspects make the application development a challenge. Besides the complexity of application development and implementation, during the runtime, the application needs (Quality of Service – QoS) which should be achieved by the system. Applications' QoS may vary along time according to the coverage area and set of sensors necessary. Furthermore, the quality of the data characterized by its quality metrics (accuracy, reliability, noise to signal ratio, etc) should also be accomplished.

Real scenarios have many applications, running simultaneously, sharing the wireless network and common resources of data. On the other hand, the bandwidth and sensors battery are limited. So, it is necessary to apply some metric to schedule the transactions (communication between data source and application) and to determine priorities among the applications. In general, real time and high risk applications have a greater priority.

In relation to data management, different aspects should be considered such as data processing (DSP algorithms), data fusion algorithms and the different levels of data fusion, data analysis (real time or not real time), including the data fusion optimal node in distributed applications, decision node, synchronization, data transmission and access security.

One of the most important aspects related to sensor networks application development is the power management. The power management approach should increase the sensors' lifetime and as a consequence the system's lifetime. To achieve these goals it should take into account that the cost of communication (transmission > reception) is greater than the cost of sensing and processing.

Most sensor networks applications consider the goal of monitoring some type of environment. This raises the discussion of ethical aspects related to the invasiveness of the application and data privacy and security. These aspects increase when the application is related to personal data received from a body-worn sensor network. In this type of application, trials should be run to evaluate the personal and social influence of the technology.

To address some of the problems described above, we have worked on different aspects and solutions. We consider that the combined use of the proposed solutions should be used as a base to build a sensor networks application. As a proof of concept, we have developed a body-worn sensor networks environment and prototype as a centralized monitoring system based on a PDA (Personal Digital Assistant).

In the first chapter we describe some motivations for the development of data fusion algorithms in the sensor networks field. We also list some of the contributions of the present research and an overview of the thesis.

In the second chapter we propose a data fusion architecture that is general enough to be employed in different types of sensor networks applications. It was designed in UML (Unified Modeling Language) and allows a dynamic model of different data fusion necessities. As previously mentioned, one of the most important characteristics of the sensor networks application is that it is data centric. So, the data fusion architecture and algorithms have a great influence on sensor networks application's performance and implementation. Furthermore, communication is responsible for the highest cost in the sensor networks running cost. Consequently, the assumptions and decisions made on the

data management and data fusion have a great influence on system's lifetime and performance. Our proposed architecture provides an infrastructure that can be applied in both distributed and centralized sensor networks implementation.

In the third chapter we present the general aspects related to a body-worn sensor networks application development. It describes specific aspects related to the body sensing and different aspects that could be relevant to continuous health monitoring.

The fourth chapter presents a centralized implementation of a body-worn sensor networks application for human body physiological monitoring. This software infrastructure is divided into three modules. The first module manages the sensors, the second module manages the data (data fusion architecture implemented for body-worn sensor networks), and the third one manages the decision to provide the Quality of Service required. All the modules are integrated to provide fault tolerance, resilience to the system's functionalities in a ubiquitous computing environment. To accomplish the ethical and privacy aspects we developed a personal data base to store all the user data. This is responsible for the decision to open or not open the data base to the health care system.

The fifth chapter presents the data fusion implementation in a distributed application. It proposes new algorithms based on the shortest path algorithm to find the optimal data fusion node. The experiments (simulation) are based on the assumption that the data comes from different sources (sensors) and the decision should be made in a decision node. To find the best data fusion optimal node we consider the cost of sensing, processing, transmission and reception (routing) using multiple hops paths. The integration of sensors management, data fusion architecture and application implementation was achieved by the development of a language to express in a sensors networks middleware all the necessary features.

In chapter 6 we present the development of all the necessary aspects that should be considered to integrate these aspects into the middleware. We think that the technologies and solutions proposed can help application's developers to build sensor networks applications compatible with fault tolerance and resilience of their functionalities in a ubiquitous computing environment. Although we have applied all the concepts to the body-worn sensor networks application, we think that they are generic enough to be applied in different types of sensor networks applications.

Chapter 1

Motivations and Overview of the thesis

In this chapter we are going to present some motivations for the development of data fusion algorithms in the sensor networks field, the problem we want to solve, the scientific contributions of the thesis, and an overview of the thesis.

1.1. Motivations

Sensor Networks are designed to sense the environment and to communicate without the necessity of the existence of any infrastructure or centralized administration. This can be implemented in a centralized or distributed approach (multiple hops may be needed for communication). These networks can function as stand alone wireless network, meeting direct communication needs, or as an addition to infrastructure based networks to extend or enhance their coverage. One of the parameters used to measure the sensor networks performance is based on the application Quality of Service (QoS) goals. As a consequence, sensor management and network optimization should consider the application QoS desired. Sensor networks are related with constraints at the application, sensor and network infra-structure sides. At the application level a dynamic QoS system should vary to achieve the application performance desired. At the sensor side, the sensor's processing capacity, lifetime power supply, and deployment are the most important aspects to be considered. From the network point of view, bandwidth, distributed communication protocols, mobility, and the dynamic aspects of the network nodes (sensors) impose great problems to the communication optimization. Besides these aspects, all these features should be integrated at the system level, taking into account resilience and ubiquitous computing goals.

The development of wireless network technology and improvements in sensors and embedded devices has enabled the convergence of mobile applications and embedded environments in the sensor networks field. One of the requirements of mobile embedded wireless devices is that they should be available at all times. In such systems, components can be inserted or excluded without stopping the entire system. In this type of system, power and bandwidth constraints should be considered. If the system is battery powered, each component of the system has a different lifetime, which is based on the battery capacity and the device's power consumption. Power and bandwidth are limiting factors to resource use.

All these problems should be addressed to guarantee that system performance degrades gracefully as resources are diminished. A home security system, robot navigation and a health monitoring system are examples of systems, where, based on data that come from sensors, a view of the environment is obtained and some decisions are made. The decisions made can be the control of an actuator (e.g., the wheels of the robot) or a change in the number and location of sensors (system's reconfiguration). To achieve the last goal, some sensors can be turned on and other sensors can be turned off. To achieve the connection between the view of world and the system reconfiguration, we can classify the changes in environment over time as different states. The requirements of an

application may change according to the state of the environment and, consequently, different components (i.e., sensors) should be used accordingly.

Dynamic systems usually share a similar architecture, sometimes consisting of three important modules: a dynamic network, middleware, and service suppliers and service consumers on the top of the middleware. A network is called dynamic when it can be reconfigured according to some system parameters. These parameters include scheduled transactions, power constraints of the network's nodes and bandwidth distribution among transactions, as well as other aspects.

Middleware is the software that connects service suppliers and service consumers. The connection between service suppliers and service consumers is called a transaction or an interaction. A service supplier is any type of network linked node (device or software) that can offer a service. Printers, sensors, databases, and applications are all examples of service suppliers. A service consumer is any type of network linked node (device or software) that requires a service from a service supplier. Actuators and applications are examples of service consumers.

Applications can be service suppliers and service consumers simultaneously. For example, a blood pressure analyzer is a service consumer when it receives a blood pressure signal from a blood pressure sensor (service supplier), but it can also send the result of its analysis (acting as the service supplier) to a display (service consumer), advising the user that his blood pressure is abnormal.

A dynamic system should be available all the time. To achieve this goal the system should be resilient, fault tolerant, and achieve all the Quality of Service (QoS) requirements specifications. Resilience is defined as an ability to recover from or adjust easily to changes in available resources, such as node failures, as well as changes in system state, such as a change from healthy to diseased. Resilience is a more general term than fault tolerance, which is the ability to recover or adapt to different types of failures, because it includes adaptation not related to failures, such as adapting the system to event detection in the environment.

As a consequence of the fault tolerance and resilience goals, we can achieve graceful degradation, which is the ability to progressively decrease system functionality in the presence of a progressive decrease in available resources. Graceful degradation is necessary to achieve the goal of computing all the time with a certain QoS (i.e., Ubiquitous Computing). Although the definition of QoS changes in different scenarios, we will define QoS as the necessary requirements to achieve a specific goal. The specific goal can be: an adequate exchange of data in a transaction, a better use of the network bandwidth, a power management approach compatible with lower power consumption, a better view of the environment by an application, and so on. The necessary requirements to achieve these goals depend on the network characteristics - for example, the bandwidth, the service supplier and service consumer requirements, and the entire system application priorities. Therefore, to achieve the entire system QoS requirements, it is necessary to achieve the QoS requirements of all its components.

We can find papers about the development of dynamic systems in three major areas: middleware, data fusion and distributed systems. As previously mentioned, middleware is the software that connects network components. Data fusion is a framework to manage data with the aim of obtaining information about the state of the system being monitored. Distributed systems study all the themes related to the use of

distributed components. Although these areas have been used in the development of dynamic systems, to the best of our knowledge, the data fusion implementation and its relationship to middleware, and the application in dynamic distributed systems has not been studied extensively. Furthermore, few researchers have considered the relationship between data fusion and middleware.

The frameworks for data fusion available in the literature are restricted to specific areas like image processing and military applications. Even the taxonomy used in these applications is very specific. It is necessary to represent data fusion as a more general model that can be applied in different contexts. Based on these aspects, the development of a general data fusion framework concomitantly to a new taxonomy can help to achieve the aim of application-middleware communication for general data fusion systems. Application-middleware communication based on a data fusion framework is new and will help to achieve the goals of resilience and graceful degradation from the application's perspective. The application module should communicate to a middleware and middleware should communicate to the network to solve the same problems from the system's perspective.

Based on the application-middleware communication described above, we need to achieve two specific goals. First, to specify an application development framework enabling the dynamic use of different data sources (data fusion framework), and second, based on the data fusion framework evaluation of the available data, a decision module should determine the current application's needs (QoS) with respect to available components. The second goal is necessary to adapt the network as the state of the system being monitored changes. Although the use of a middleware to manage the interface between the application and the network components is not obliged, it eases the system's management and allows scalability.

The application development framework can be achieved by developing two integrated modules: a data fusion framework and a decision module. "Data fusion is defined as a formal framework in which it is expressed the means and tools for the alliance of data originating from different sources to achieve some view of the environment" [1]. This definition shows that data fusion is used to combine data (physical measures from the world) to describe the world. The fusion process depends on the model representation of the world. As an example, in a security system we can fuse a sound sensor data and an image from a camera to identify the presence of a person in a room. How the data is combined depends on how the person identified is modeled. In this case, we can obtain from the sound sensor the identification of human sound (voice) with some measure of confidence. Concurrently, we can obtain an image compatible with a person with another degree of confidence. We can fuse both data to increase the confidence that a person is present in the room (inference). As we can model the world in different ways, the data fusion process varies according to the model.

Data fusion is the part of an application responsible for the management of different sources and types of data. If the data fusion framework manages data from different sources, we can design a data fusion framework that adapts to different situations of the environment (resilience), i.e., different states of the world, or different availability of the physical measures of the environment (fault tolerance and resilience). Data fusion is a well-researched area from the Artificial Intelligence and Systems fields

that recently was used to describe how data coming from different sensors could be merged in a sensor network [1][2].

The other part of the application to be developed is the decision module. The decision module is important to achieving the goal of adapting the network, i.e., changes in the nodes connected to the application, based on changes in the system's state. The data fusion module processes data from different sources and achieves a view of the world, but this module does not interfere in the physical measures. It is used to evaluate the world based on the data available. To achieve resilience, fault tolerance and graceful degradation we need more than that. We need to choose the appropriate number and type of sensors compatible to each situation of the environment, according to the availability of the data sources and compatible with the power and bandwidth constraints of the system. In the case of a security system, based on the bandwidth cost of video transmission, we can turn on cameras only if the sound detects some problem. To achieve this goal we need a view of the world based only on the sound sensor. This can be provided by the data fusion module. The decision to turn on the camera is provided by the decision module. The decision module chooses the best set of physical measures of the world to improve or decrease the view of the environment. As a consequence, based on the view of the environment that comes from the data fusion, the decision module provides new application QoS requirements to the middleware. This QoS can include the new set of necessary sensors to have a better view of the system, an increase or decrease in the data rates of the sensors, as well as other changes.

As a consequence of the data fusion framework integrated with the decision module, the application would change dynamically according to the environment and to the available resources in the system. The problem in achieving this goal is that the application is not aware of the availability of data sources in the system. This information is available at the network level, not at the application level. Therefore, we need something to connect the network to the application in a dynamic way. The connection of the network information (services available) with the application requirements is the role of the middleware. The decision module should analyze some results of the data fusion and automatically inform the middleware of the application's new requirements. Given this information, the middleware matches the current needs of the application with the available sources of data and sends this information to the network for configuration of the connections between nodes.

It is important to emphasize that both middleware and application should be compatible with fault tolerance, resilience and graceful degradation to achieve the ubiquitous computing aim. In the case that only middleware is compatible to these aspects, fault tolerance, resilience and graceful degradation goals will not be satisfied at the application level. The inverse is also true. Another important aspect is related to the application and middleware relationship. Even if both systems achieve graceful degradation, it is not enough. Middleware and application should communicate to each other in such a way that makes it possible to achieve the graceful degradation in the entire system.

1.2. Problem Statement

An example of a dynamic, distributed system that could benefit from the data fusion framework and application-middleware communication described in the next section is a Personal Heart Rate Monitor.

A multi parametric Personal Heart Rate Monitor (PHRM) is a wireless-based mobile dynamic system with the goal of monitoring the heart's condition in healthy and diseased conditions. It provides a continuous monitoring of the subjects' Heart Rate. The multi parametric PHRM consists of a body-worn sensor network powered by battery and connected by a wireless network. It is designed to use different types of physiological sensors to monitor the user's heart rate: blood pressure, pulse oximeter, blood flow, arterial pulse, Electrocardiogram (ECG), Electromyogram (EMG), Electroencephalogram (EEG), and others. The use of all of these sensors will provide the evaluation of the heart rate in different conditions of the body. The heart rate varies according to body activity, temperature, blood pressure, position and so on. The PHRM evaluates the absolute and relative heart rate values in each condition.

PHRM is a mobile system. In this case, bandwidth and power constraints should be considered. To achieve continuous monitoring in different conditions, the system should have reasonable autonomy. Node lifetime is inversely proportional to power consumption. To decrease the power consumption of the system components, it is necessary to adjust the use of sensors to the current necessity of the application. As an example, if the user is completely healthy (all data are in the normal range), we can decrease the number of sensors and turn off the sensors not in use. This approach increases the lifetime of the sensors and of the system overall. When an event is detected from the current available data (e.g., a high blood pressure) the system can turn on the sensors related to the event (i.e., ECG and pulse oximeter). This is called adaptation to environment changes. The system should also adapt to loss of available sensors (adaptation to the availability of system components). As a result, if we can adapt the system to all events occurred in the environment or at the sensors level, the system can be resilient, fault tolerant, and provide graceful degradation. Although there are different adaptation definitions in the literature, we will consider in the entire text the definition described in the scenario above.

All the sensors will be connected by a wireless network on the body to an application on the top of a middleware. The discovery service will send the available sensors to the middleware, which will send the required data to the application. The data fusion framework will process and fuse the data to come up with a view of the heart's health. Based on this view, the decision system will determine the current PHRM sensors necessity (application QoS). As an example, consider that the user is healthy and that all the data that come from the body are normal. Based on these aspects, the power management policy will turn off most of the sensors. The monitor will be based on one ECG lead and a blood flow sensor. Now, imagine that the system has just recognized an arrhythmia. As a result, the decision module will request that the middleware increase the number of leads of the ECG to 3 and ask for the blood pressure and pulse oximeter data. This information will be sent from the middleware to the network. The network will schedule the new transactions and will also be reconfigured according to the new

requirements of the system (assuming this is feasible, based on sensors available and bandwidth constraints). As a consequence, in the next step the PHRM application will receive data from 3 ECG leads, from the Blood pressure sensor and from the pulse oximeter sensor.

We have developed a data fusion architecture for applications of a network-based dynamic system with the following characteristics:

Components of the system can be inserted or excluded without stopping the entire system;
The environment changes with time and so do the physical measures from it;
The application's necessity changes according to different states of the environment, and as a consequence, it can use or reuse components in the different states;
The system is mobile and battery powered, so each component of the system has a different lifetime, bandwidth usage and power consumption;
As a network-based system, bandwidth is limited and coverage area is variable.

All of these problems will be addressed from the application perspective to make the system robust to the dynamic environment.

The PHRM is from the class of network-based mobile dynamic systems powered by battery, where an application should adapt itself to different configurations of the system (data sources moving in and moving out), different states of the environment, and considering power and bandwidth constraints. As a consequence, the solution of these problems will solve the same problems in the class of related systems. Controlling Robots based on the environment and home security systems are two examples of this class of systems to which our framework should be applicable.

We propose a solution to the problem of developing an application framework to manage data from different types of sensors to perform a Heart Rate Monitoring application in a Ubiquitous Computing environment. In this work, we will focus on the application's framework (data fusion and decisions modules) while also considering the necessary middleware and network facilities to ensure resilience to changes in available resources and changes in the environment's state.

1.3. Contributions

The contributions of this research include:

A general data fusion architecture for mobile network-based dynamic monitoring systems;
Algorithms for distributed data fusion implementation in sensor networks;
Software infra-structure for centralized sensor networks to provide fault tolerance, resilience and QoS in dynamic environment;
A language for application-middleware relationship;
An ECG interpretation algorithm based on continuous variable compilation; Multi parametric-based body-worn heart rate monitor;

QoS based on system states and user (environment) states.
Algorithm for distributed data fusion.

1.4. Thesis Overview

1.4.1. Introduction

Sensor Networks are designed to sense the environment and to communicate without the necessity of existence of any infrastructure or centralized administration. It can be implemented in a centralized or distributed approach (multiple hops may be needed for communication). These networks can function as standalone wireless networks meeting direct communication needs, or as an addition to infrastructure based networks to extend or enhance their coverage. One of the parameters to measure the sensor networks performance is based on the application Quality of Service (QoS) goals. As a consequence, sensor management, and network optimization should consider the application QoS desired.

Sensor networks are related with constraints at the application, sensor and network infra-structure sides. At the application level, a dynamic QoS system should vary to achieve the application performance desired. On the sensor side, the sensor's processing capacity, power supplier lifetime, and deployment are the most important aspects to be considered. At the network point of view, bandwidth, distributed communication protocols, mobility and the dynamic aspects of the network nodes (sensors) impose great problems to the communication optimization. Besides these aspects all these features should be integrated at the system level taking into account resilience and ubiquitous computing goals.

These different aspects should be considered in the design of sensors networks applications:

Nodes deployment;
Nodes Addressing;
Nodes design;
Routing;
Security;
Service Discovery;
Network media: 802.11b, Bluetooth, UWB and Infrared;
Network configuration and topology;
Authentication, Authorization and Accounting;
Data fusion;
Power management;
Sensors management;
Optimization;
Resilience and fault tolerance.

Although data fusion is related as one item of this list, it is related to almost all the other items because it is a low level representation of the application's quality of service.

In this thesis we are going to focus on data fusion implementation on sensor networks, considering some of the aspects described above.

1.4.2. Data Fusion

Data fusion is a formal framework used to express the convergence of data from different sources in which is expressed the means and tools for the alliance of data that originated from different sources. The US Department of Defense has defined data fusion as a multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from single and multiple sources. The resulting information is more satisfactory to the user when fusion is performed rather than simply delivering the raw data. In data fusion, information may be of various types, ranging from numeric measurements to linguistic reports. Some data cannot be easily quantified, and their accuracy and reliability may be difficult to access.

Sensor measurements have problems related to noise, errors and incompleteness. In addition, we often cannot have a complete view of the world based on data from only one sensor (incompleteness). Associated with sensor data use, we have to evaluate its reliability. Reliability attempts to represent how much confidence we have in the data that comes from the sensor. All of these aspects contribute to increasing the uncertainty in the system. Thus, we need a formal data fusion framework that represents and provides tools to manage all of these different problems. None of the frameworks described until now achieves this objective for different types of applications and scenarios.

There are different levels of data fusion. We can have data fusion from one sensor (time series), redundant sensors, redundant variables, variables and systems. We can even fuse different levels of data. We can find different approaches in the literature to treat this problem. Some researchers use statistical analysis like mean, average, median, standard deviation, correlation and variance (the Kalman filter algorithm). Other researchers use heuristical approaches to manage the uncertainty, such as probabilistic models based on Bayesian networks or uncertainty sets, possibility models based on fuzzy logic and Dempster-Shafer theory, mathematical models, learning algorithms based on neural networks and evolutionary algorithms, and hybrid systems. Which approach to use depends on different aspects, such as the type of data, the requirements of the application, and the grade of reliability desired.

Although there are different papers in the literature addressing the data fusion problem and the management of incomplete data, there is a lack of a better definition of the different levels of data processing and analysis that need fusion. Some papers address the fusion of signals, others address the uncertainty of high level data fusion using different methods, but none of them have tried to establish a formal framework that includes the different levels of data fusion. Besides this aspect, there is a need for a taxonomy that defines what is low and high level fusion. We present a formal framework based on UML (Unified Modeling Language) and describe a taxonomy that defines the different levels of data fusion. Next we discuss the data fusion implementation in distributed and centralized sensor networks scenarios. Finally, we present a sensor networks application of a body-worn sensor network. We applied the framework and taxonomy described here to solve the dynamic management of data in a Personal Health Monitoring System (PHMS). Our framework can be applied to any of the class of problems characterized by

monitoring the environment using different types of sensors, such as a home security system, a military application in a war, control of robots based on the environment, etc.

1.4.3. System Architecture

This section shows the relationship of the data fusion with the whole system. We think that to achieve all the requirements to make a system available at all times we need to integrate the network, the service suppliers (data sources) and services consumers (applications). The application development can be divided into several integrated modules; in this paper we will address the data fusion and decision modules.

A network-based system needs all of its components (network, middleware and application) to deal with dynamic changes in the availability of resources and changes in the environment. The system should adapt to the availability of sensors and their corresponding signals, and it should also adapt to changes in the measurements themselves.

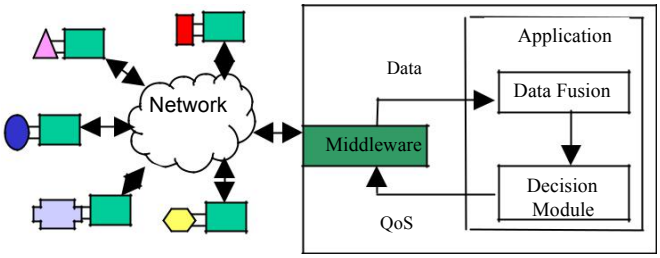


Figure 1: Network, middleware, and application

Figure 1 shows a block diagram of a general system. The sensors (service suppliers) and the application (in this case service consumer) are nodes of the network. The middleware is the software that connects the sensors to the application through the network. The application has two integrated modules, the data fusion module and the decision module. The application sends its QoS requirements to the middleware, and the middleware sends the sensor data to the application. Next we present the data fusion architecture and discuss the distributed and centralized data fusion implementation.

1.4.4. A General Data Fusion Architecture

We have developed a data fusion architecture that can be applied for network-based dynamic distributed systems or conventional stand alone centralized systems. The following characteristics can co-exist in both systems: components of the system can be inserted or excluded without stopping the entire system; the environment changes with time, as do the physical measures from it; an application's necessity changes according to different states of the environment, and as a consequence, it can use or reuse components in the different states; the system can be mobile and battery powered, so each component of the system has a different lifetime, bandwidth usage and power consumption. In the case of a network-based system, bandwidth is limited. All of these problems should be addressed from the application perspective to make the system robust to the dynamic environment. We think that some of these problems can be addressed at the data fusion module of the application.

1.4.5. Centralized Data fusion implementation in sensor networks

Figure 2 shows the diagram of the software tool for applications development based on a body-worn sensor networks. The application is developed based on the data fusion module and the decision module. The data fusion module is based on the architecture proposed by the authors in [4] and shown in figure 1. It is responsible for the fusion of data from the same type of sources (redundant data) and data from different types of sources. The different levels of the data fusion system are composed by different types of agents responsible for processing and analyzing different types of data. The data fusion system provides dynamic software to represent the changes in the environment (body and environment where the body is placed), conditions of the sensors, and capability of the processing unit. This adaptability of the system is compatible with fault tolerance and ubiquitous computing. The decision module makes changes in the system to follow the new necessities. It is also responsible for the control of messages delivered and actuators, log and exchanges of information with other information systems (for example the electronic medical record).

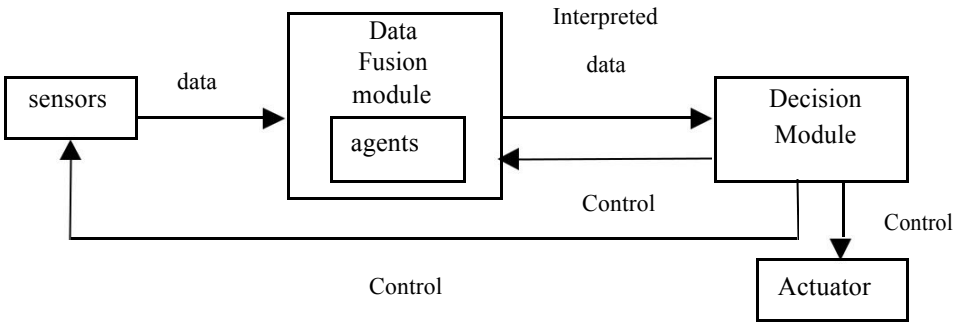


Figure 2: Centralized Data Fusion block diagram

We have applied this architecture to develop a centralized sensor networks application applied to the physiological signals monitoring.

1.4.6. Distributed Data fusion implementation in sensor networks

Data fusion is an important aspect related to the application development in a sensor networks environment. It is directly related to the application quality of service specifications and also related to different aspects of the sensor networks environment. Based on these aspects, different aspects should be considered in the data fusion implementation in sensor networks:

1) **Communication cost** (receiving and transmitting)
    a. Influence of different sizes of data. If the data is higher than the packet, we need to add the number of necessary messages to send all the data;

2) **Data fusion cost**;
3) **Data fusion precedence**;
4) **Sensing cost**;
5) **Node capability**: power, computation capability and functionalities capability
    b. All the nodes have the same capabilities;
    c. Heterogeneous nodes;
6) **Different data rates:** it can determine the use of shortest or longest paths;
7) **Real time versus not real time**: real time applications need a short path and smaller delays;
8) **Security**: Adds overhead in the communication cost;
9) **System lifetime**: how many paths can we use to provide a larger system's lifetime? From the lifetime point of view, the use of as many paths as possible is useful;
10) **Data throughput**: the shortest paths are related to a higher throughput than the larger paths;
11) **Transmission delay**: the transmission delay is higher in the long paths than in the shortest paths;
12) **Time synchronization:** it is easier to obtain on shortest paths synchronization than in the longest paths;
13) **Location integration**:
    d. Scenario 1: any source, anywhere;
    e. Scenario 2: sources to be fused are near each other;
14) **Scalability:** We can apply the algorithm to the entire network or divide the network in smaller domains represented by different destination nodes, and integrate the domains' destination nodes to scale to the entire network;

We have developed some algorithms to find the best optimal node to fuse the data from different sensors considering the aspects described above.

1.4.7. Language for Middleware Application relationship

We presents a language based on graphs on how an application can specify for middleware the set of sensors it will need during the run time. Furthermore, it specifies the metrics and the value of the metrics the application Quality of Service will determine during run time. Based on this information, middleware can determine the best set of sensors compatible with a longer system's lifetime.

Another very important aspect is related to the specification of the applications priorities and sensor management. They are related to the value specified to each application and the QoS.

.

1.4.8. Conclusions

We presented some important features that should be considered in the data fusion implementation in sensor networks. It includes data fusion architecture, applications development tool (data fusion implementation and decision module), and a language for data application and middleware relationship. We have proposed general data fusion

architecture (DFA) described in a formal language of object representation (UML) that tries to represent different scenarios, specifications and features of a general data fusion system. It also allows a dynamic modification of the system according to different states of the environment or of the system. The DFA proposed is a good solution to the problem of developing an application framework to manage data from different types of sensors to perform different tasks in a ubiquitous computing environment. We have mapped the data management problem to different domains and applications to show that we can employ our data fusion architecture to diverse scenarios, including different contexts and domains. Based on the Data Fusion Architecture proposed we have developed algorithms to perform distributed data fusion and centralized data fusion systems considering different constraints imposed by the sensor networks environment. To achieve the fault tolerance and resilience goal we have developed a language for data fusion architecture and decision modules of the application integration to the middleware. Finally, we presented an infra-structure to a body-worn sensor networks application. As proof of concept we have developed a prototype of a Personal Heart Rate Monitoring system.

Chapter 2

A General Data Fusion Architecture

2.1. Introduction

Recent developments in wireless networks and in miniaturization of powerful embedded devices have enabled the development of very small computing systems that are available at all times. In the literature, this type of computation has been called ubiquitous computing. Several applications of ubiquitous computing (including ones that cover life threatening situations) require fault tolerance, resilience and graceful degradation in response to different types of failures in the system. In this paper, we address the problem of how to develop applications, taking into account fault tolerance, resilience, and graceful degradation in a ubiquitous computing environment. We think that to achieve these goals it is also necessary to develop a data fusion model compatible with network-based distributed systems and stand alone systems.

Data fusion is an important component of applications for systems that use correlated data from multiple sources to determine the state of a system. The fault tolerance and resilience of these applications will depend greatly on the data fusion framework. As the state of the system being monitored and available resources change, the general data fusion framework should change dynamically based on the current environment and available resources in the system. As a consequence, the data fusion framework should provide some results of the data fusion to a module called the decision system. This module is responsible for sending feedback to the middleware, so the middleware can appropriately reconfigure the network. Based on the current data or variable the decision system receives from the data fusion module, the decision system should automatically inform the middleware of the application's new requirements (i.e., the application dynamically adjusts its Quality of Service requirements based on the current state of the system being monitored and informs the middleware of its current Quality of Service needs).

In this chapter, we present a Data Fusion Architecture that we think is general enough to be applied in different contexts and applications. We also present the data fusion architecture proposed applied to different scenarios.

Problem: which architecture to use?

2.2. Literature review

We have developed a data fusion architecture that can be applied for network-based dynamic systems (distributed) or conventional stand alone (centralized) systems. The following characteristics can co-exist in both systems: components of the system can be inserted or excluded without stopping the entire system; the environment changes with time and so do the physical measures from it; application's necessity changes according to different states of the environment, and as a consequence, it can use or reuse components in the different states; the system can be mobile and battery powered, so

each component of the system has a different lifetime, bandwidth usage and power consumption. In the case of a network-based system, bandwidth is limited. All of these problems should be addressed from the application perspective to make the system robust to the dynamic environment. We think that some of these problems can be addressed at the data fusion module of the application.

Data fusion is a formal framework used to express the convergence of data from different sources in which is expressed the means and tools for the alliance of data originated from different sources [1]. The US Department of Defense has defined data fusion as a multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from single and multiple sources [2]. The resulting information is more satisfactory to the user when fusion is performed rather than simply delivering the data [3]. In data fusion, information may be of various types, ranging from numeric measurements to linguistic reports. Some data cannot be easily quantified and their accuracy and reliability may be difficult to access.

Sensor measurements have problems related to noise, errors and incompleteness. Noise is related to different types of interference. It is directly related to the sensor device and its environment. Errors can be related to systematic or random errors of the sensor device. In general, transducers do not operate in practical operation like they operate in theory or simulations. In addition, we often cannot have a complete view of the world based on data from only one sensor (incompleteness). Associated with sensor data use, we have to evaluate its reliability. Reliability attempts to represent how much confidence we have in the data that comes from the sensor. All of these aspects contribute to increase the uncertainty in the system. Thus, we need a formal data fusion framework that represents and provides tools to manage all of these different problems. None of the frameworks described until now achieve this objective for different types of applications and scenarios.

There are different levels of data fusion. We can have data fusion from one sensor (time series), redundant sensors, redundant variables, variables and systems. We can even fuse different levels of data. We can find different approaches in the literature to treat this problem. Some researchers use statistical analysis like mean, average, median, standard deviation, correlation and variance (the Kalman Filter Algorithm) [4]. Other researchers use heuristical approaches to manage the uncertainty, such as probabilistic models based on Bayesian networks or uncertainty sets [5][15], possibility models based on fuzzy logic and Dempster-Shafer theory [4][6][17], mathematical models [7], learning algorithms based on neural networks and evolutionary algorithms [8], and hybrid systems [8]. Which approach to use depends on different aspects, such as the type of data, the requirements of the application, and the grade of reliability desired.

E. Waltez and J. Llinas, cited by [9], have described important features related to the development of data fusion architectures: robustness and reliability, extended coverage in space and time, great data space dimension, reduced ambiguity, and a solution to information explosion.

Another important aspect of data fusion is related to system representation and the data fusion framework. Most of the papers published in this area are related to military applications and image processing. The military application field is mainly represented by the functional model developed by the Joint Director of Laboratories (JDL) from the

U.S. Department of Defense. Their functional model is represented by four levels. The first level is related to the identification and description of the objects; the second level represents an interactive process to fuse spatial and temporal entities relationships; the third level is associated with the combination of the activity and capacity of enemy forces to infer their force; and the fourth level is related with all other levels and is responsible for regulation of the fusion process [2]. Although this model has been applied in large scale projects related to military applications, it seems to be very specific to this field.

The papers related to image processing are basically applied to two fields, robot navigation and geographical data. Durant presented a framework to integrate and to model coordination and control of robot systems [10] and Arnoud proposed architecture of sensors data fusion systems, emphasizing the benefits of providing high level fusion [4]. Clement and others described a specialist-based knowledge approach [11]. Matsuyama and McKeown worked with hierarchical descriptions of image fusion [12] [13]. Growe [14] developed a framework based on semantic nets representation using fuzzy membership function to determine whether fusion is possible.

A few papers have addressed different domains. Dailey, among others, described data fusion applied to transportation [15]. Laskey and others, used Knowledge and Data Fusion in Probabilistic Networks applied to the medical diagnosis domain [16]. The authors describe the use of probabilistic networks to represent and model a medical diagnosis approach. They have shown that novel models of Bayesian networks can learn with new observations and that it should apply when we have complete or incomplete observations about the phenomenon. The use of probabilistic approaches can provide assumptions about the variables even when they are not measured. This is an important aspect related to fault tolerance and data incompleteness management.

Although there are different papers in the literature addressing the data fusion problem and the management of incomplete data, there is a lack of a better definition of the different levels of data processing and analysis that need fusion. Some papers address the fusion of signals, others address the uncertainty of high level data fusion using different methods, but none of them have tried to establish a formal framework that includes the different levels of data fusion. Besides this aspect, there is no taxonomy that defines what is low and high level fusion. We present a formal framework based on UML (Unified Modeling Language) and describe a taxonomy that defines the different levels of data fusion. We applied the framework and taxonomy described here to solve the dynamic management of data in the Personal Health Monitoring System (PHMS), but this framework can be applied to any of the class of problems characterized by monitoring the environment using different types of sensors. Thus, if we solve the PHMS data fusion problem and we can map the PHMS to other applications like a home security system, a military application in a war, control of robots based on the environment and so on, the data fusion framework we developed can be applied in all these different applications.

2.3 Solution proposed: A General Data Fusion Architecture

2.3.1 Relationship between the Data Fusion Architecture and the Entire System

We think that to achieve all the requirements to make a system available all the time, we need to integrate the network, the service suppliers (data sources) and service consumers (applications). The application development can be divided in several integrated modules. We are going to address the Data fusion and decision modules.

A network-based system needs all of its components (network, middleware and application) to be compatible with dynamic changes in the availability of resources and changes in the environment. The system should adapt to the availability of sensors and their corresponding signals, and it should also adapt to changes in the measurements. Figure 3 represents the system's block diagram. The sensors (service suppliers) and the application (in this case service consumer) are nodes of the network. The middleware is the software that connects the sensors to the application through the network. The application has two integrated modules - the data fusion module and the decision module. The application sends its QoS requirements to the middleware, and the middleware sends the sensor data to the application.

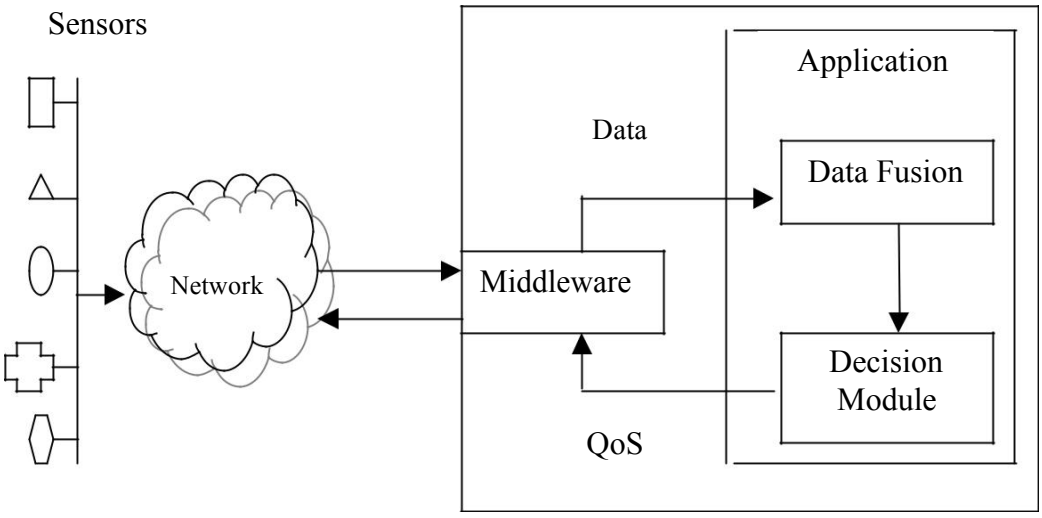


Figure 3: Network, middleware, and application relationship.

Figure 4 shows the temporal diagram of the information exchange between the system's components. When the system starts, middleware needs the QoS information from all the components (Service Suppliers, Network, Applications and from the entire system). After that, the network is configured, the transactions are scheduled and all the necessary connections (transactions) are matched by the middleware. Middleware starts to receive data and sends it to the application. The application data fusion module will process the data and pass a view of the world (environment) to the decision module. Based on this input, the decision module will determine the new requirements of the application (application QoS). Again the middleware matches the new application needs to the available resources and sends to the network the new set of connections. The network will be reconfigured and the entire process repeats in a cyclic way.

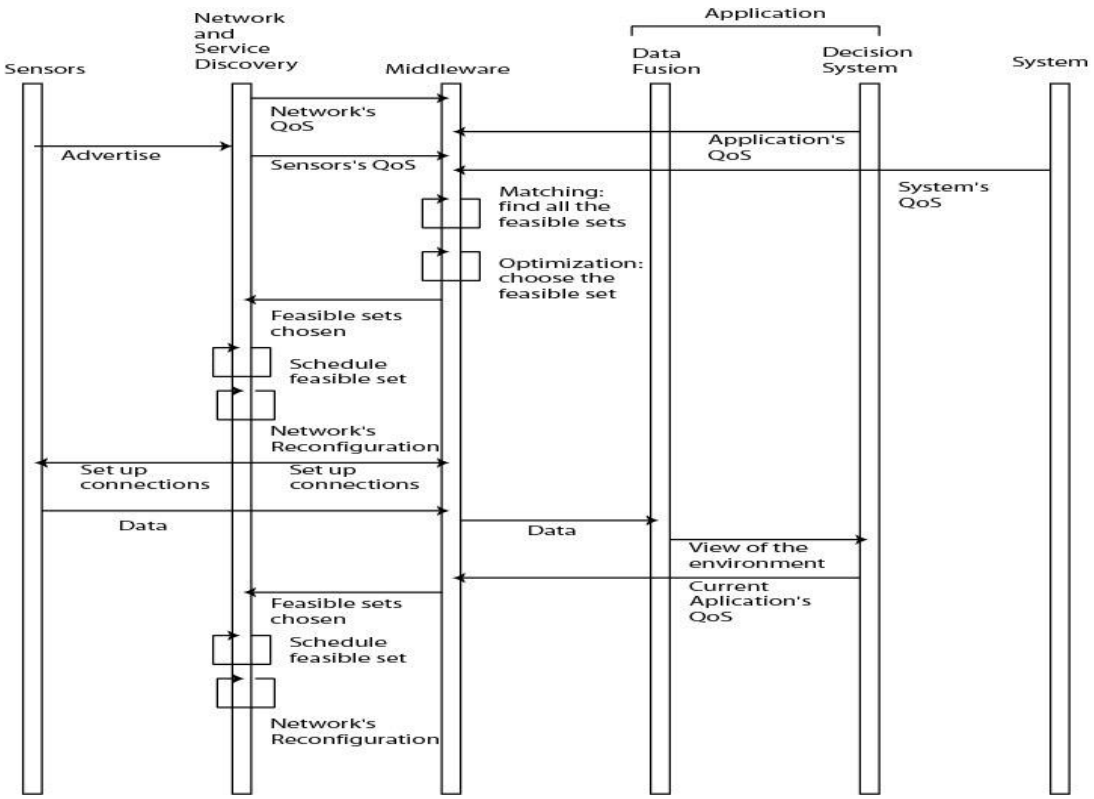


Figure 4: Temporal diagram showing the interaction among the sensors, the network, the middleware, the application and the system.

Based on the type of system described above, we are going to present the data fusion architecture and the decision modules.

2.3.2. Data Fusion Taxonomy

There are three main types of data fusion: data oriented, task oriented (variable) and a mixture of data and variable fusion. The basic idea that divides the levels is the difference between data and variable. Data is a measurement of the environment that is generated by a sensor or other type of source. Variable is determined by the presence of one or more application tasks. In general, from one type of data, we can come up with one or more variables or tasks. For example, from the data from a camera we can obtain an image of a person (variable person), an animal image (variable animal) or an object (variable object). In this case, the application has the task of identifying the image of a person, an animal or an object. These tasks can be an intermediary or the main endpoint of the application. They will be intermediary when they are a necessary step to achieve another task, or to add redundancy and fault tolerance to the system. The variable determination can be achieved by using a data analysis algorithm or using some approach, such as neural networks, that takes the data as input and gives as an output the probability of being an image of a person, an animal image or an object image. So, what determines the three levels is whether the fusion process is made at the data level

(without any analysis), after the data has suffered process of analysis (at the variable level) or transformation to a task, or in a mixture level (fusion of data and variable).

2.3.3. The Data Fusion Architecture (DFA)

Sensors exist to measure physical variables, such as a temperature, heart rate, or blood pressure. A physical variable can be measured by several sensors (which we call redundant sensors), or by a single sensor (which we call an individual sensor). Individual sensors are unique in the system, and there is no reason to use multiple sensors of this type. Redundant sensors can co-exist with multiple sensors of the same type, and it is desirable to do so to achieve fault tolerance, increase the covering area, or meet other constraints. The system should be able to differentiate these types of sensors and the data that come from them.

In general, before any data fusion can be performed, the signal that comes from the sensor should be pre-processed. The pre-processing can be as simple as an analog to digital conversion or as complex as the use of different Digital Signal Processing approaches. Different approaches and techniques should be applied to different types of signals before the data is useful for management: analog to digital conversion, error analysis, amplification, filtering, noise treatment, quantization, multiplexing, etc.

After pre-processing, the data should be fused. We propose a 3-level data fusion framework based on both data and variables. Data is defined as the data that come from a sensor. Variable is defined as a type of data that is generated after some analysis of the raw sensor data. As a consequence of this differentiation, the data fusion can be classified as low level fusion of data, high level variable fusion and a mixture level fusion. When the data fusion is performed before some analysis, it is classified as low level. After some data analysis, it is classified as high level variable fusion. There are some situations where we can fuse data and variables. A mixture fusion level class was created to represent this class of fusion. Figure 3 shows the data flow from the sensors, pre-processing unit and the data fusion framework, describing possible scenarios of data fusion represented in UML (Unified Modeling Language).

The framework presented in figures 3, 4 and 5 is new because it explicitly defines the different possible levels of data fusion, including different approaches to manage the fusion process. Furthermore, it introduces a new taxonomy for data fusion classification based on the definitions of data and variable and how to combine and fuse them. The formal representation of a data fusion framework based on UML description is also new.
Figure 3 shows the Data Fusion Architecture described in UML (Unified Modeling Language). The boxes represent classes and the arrows means the direction of possible data or variable flow. The number around the arrows represents the cardinality of the classes' instances relationship. As an example, we have only one instance of any type of sensor at a time for one instance of a pre-processing class. Text inside brackets represents constrictions. Notes are simple explanations.

The DFA shows that data comes from one sensor; it is pre-processed using operations from the pre-processing class and the pre-processed data is sent to the Low Level Data Fusion (LLDF); Each LLDF instance can receive one or more type of data as input and using any of its operation fuse them. This class can send its output (fused data) to another instance of the LLDF (multilevel LLDF), and/or to the data analysis, and/or

High Level Data Fusion (HLDF), and/or Variable interpretation, and/or Mixture level Data Fusion (MLDF) modules. The MLDF receives data from the LLDF and variable from the HLDF and its output (fused data/variable) goes back to the HLDF and/or it goes directly to the variable interpretation module. The data analysis module receives fused data, analyzing it using the appropriate algorithm to the received data and sends the resulted variables to different instances (one HLDF instance for each type of variable) of the HLDF module. At the HLDF, each HLDF variable instance performs the redundant variable fusion process and sends its resulted fused variable to another instance of the HLDF, and/or to the MLDF, and/or to the Variable Interpretation module. The Variable Interpretation (VI) module receives variables from different sources as input. It fuses all the views of the variable to provide as an output, single or multiple views about the sensed environment or data source. Its' output goes to the decision system. The decision system takes all this information and decides about modifications on the environment sensing (application's Quality of Service output), and/or control any actuator (actuator control), and/or change algorithms and data/variable flow in the data fusion model (control output goes to the LLDF, MLDF or HLDF).

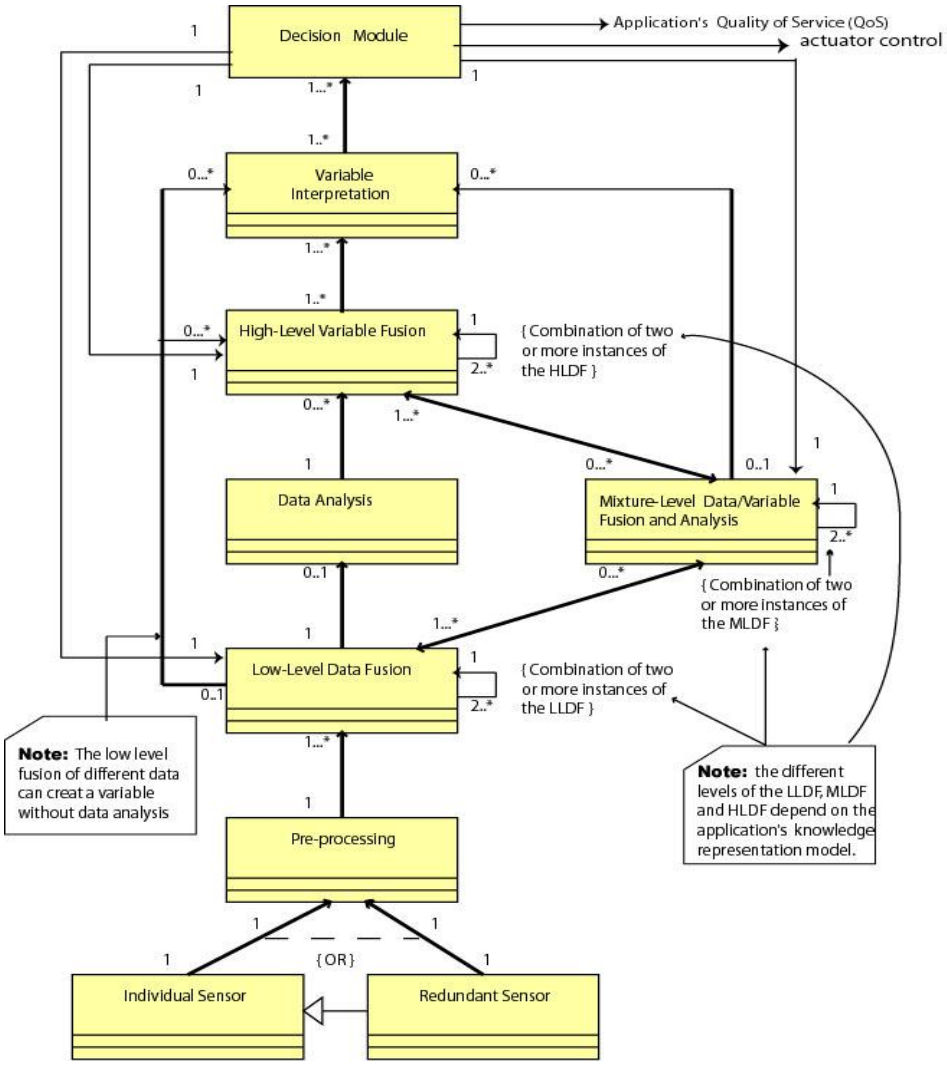


Figure 5: Data Fusion Architecture based on UML.

Figure 6 shows the classes of the individual and redundant sensors, and pre-processing classes. The class of individual sensors has attributes such as identification, type of sensor, data rates, power (lifetime), bandwidth requirement and the IEEE 1451.2 sensor characteristics. The individual class functions are related to the sensor mode (on, off, sleep, idle) as well as data rate control and battery level. The redundant sensor class heritance the individual sensor class attributes and operations and adds the characteristics related to the sensor redundancy, such as covering area and related operations. The pre-processing class includes different functions used to process analog and digital signals.

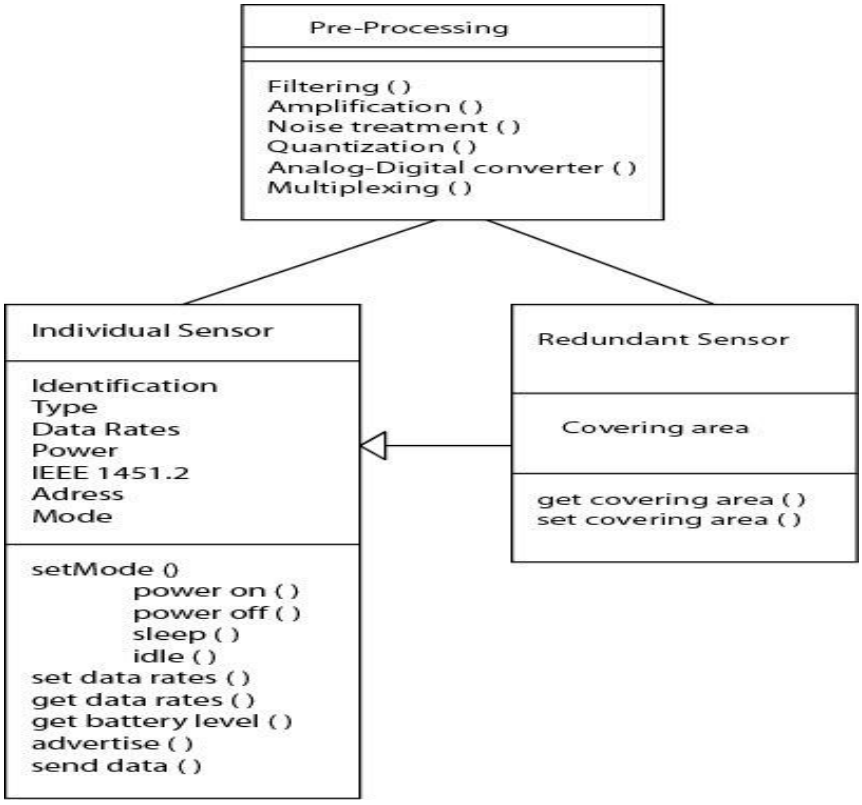


Figure 6: Individual sensor, redundant sensor and pre-processing classes.

The low level data fusion class includes different approaches that can be used in different low data fusion scenarios and is described in Figure 7. Figure 8 shows possible approaches to manage variables in the high-level variable fusion. In general, it uses the same approach as the low-level data fusion, but applied to the fusion of variables. The specific variable application fusion management approach includes the different approaches based on uncertainty management tools like belief networks, neural networks, genetic algorithms, fuzzy logic, probability theory and many others. This will depend on the knowledge model applied. Figure 9 presents the class of an unusual but used form of data fusion, the fusion of row data with variables (Mixture Level Data Fusion class).

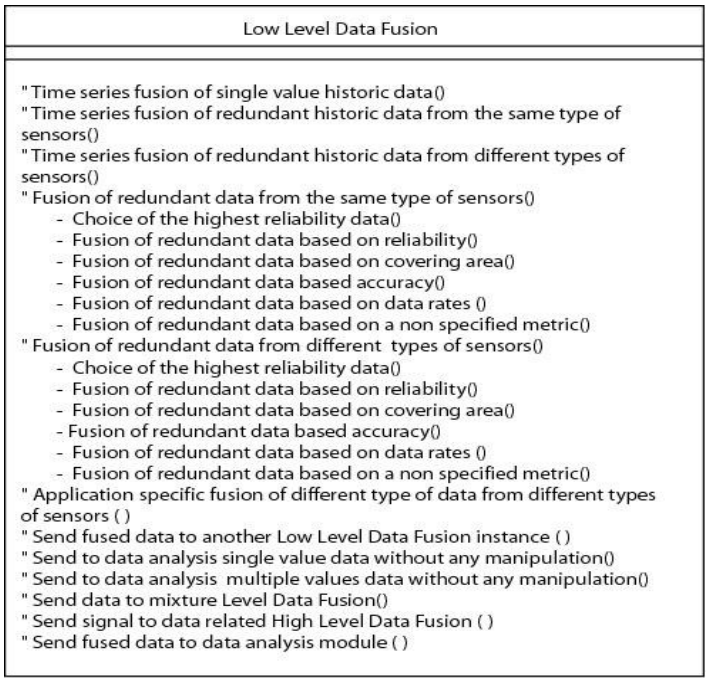


Figure 7: Low Level Data Fusion Class.

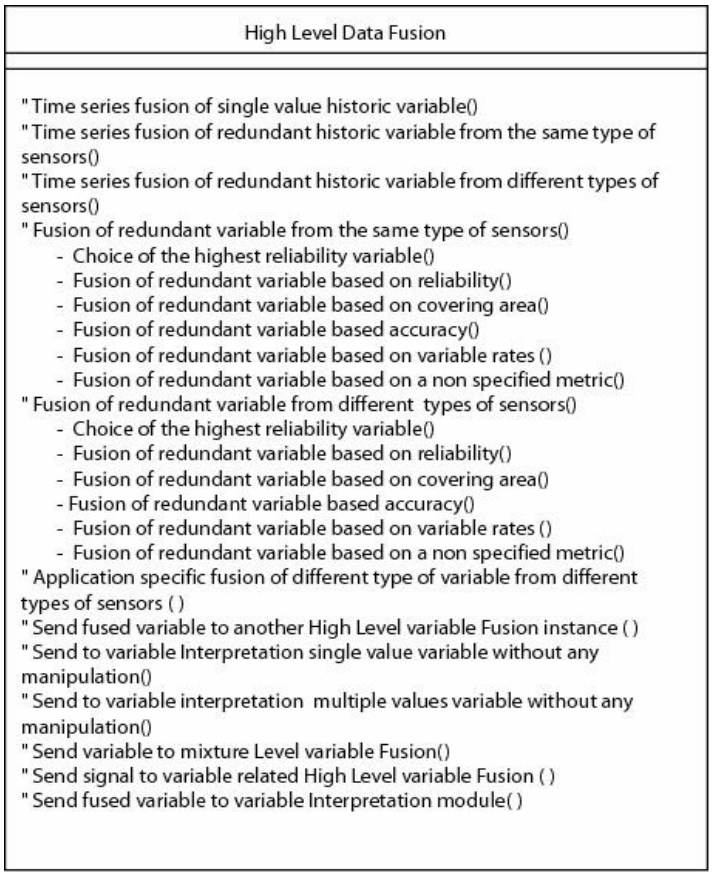


Figure 8: High Level Data Fusion Class.

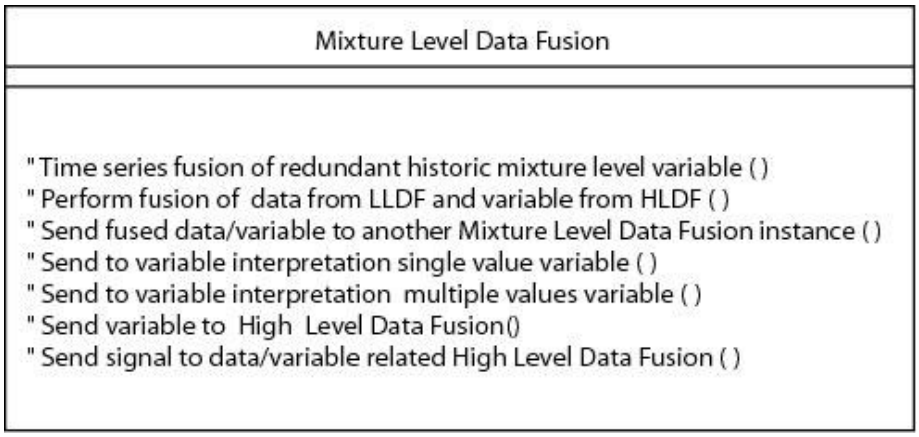


Figure 9: Mixture Level Data Fusion Class.

Figure 10 describes the two main types of input manipulation classes. These classes can be used in any part of the data fusion architecture, including the different levels of fusion in the LLDF, HLDF and MLDF. To see an example of the different level inside the HLDF see the chapter 4 describing the Health monitoring application. The synchronized class requires that all the input should be processed at the same time. If these inputs arrive at different times, we need to use an algorithm or a rule to provide the synchronization. We can classify the synchronization process in hard and soft synchronization. The soft synchronization allows some flexibility in the time for synchronization. The hard process requires a tight input time stamp. One of the available ways is to synchronize the input manipulation based on the lowest input rate. Using this approach we solve the problem of synchronization but we still have problems related to the use of the higher input rates data. Are we going to use the current or the next value of the higher input rates? So, in some situations where hard synchronization is required, we will need an algorithm to deal with the calculation of the optimal time synchronization. If the synchronization is soft we can also employ some operations from the asynchronous class. We can combine the time series approach to the weighted sun of asynchronous input to manipulate the inputs. This is a powerful approach because it takes in account historic data and the weight of ach input in the final result. The asynchronous class describes different tools to manage asynchronous inputs. These approaches can consider or not the influence of the previous ones (historic inputs data) or weight the influence of each input. The weight used in this approach can be any type of characteristic that is important for the knowledge representation such as reliability, accuracy, covering area, and so on.

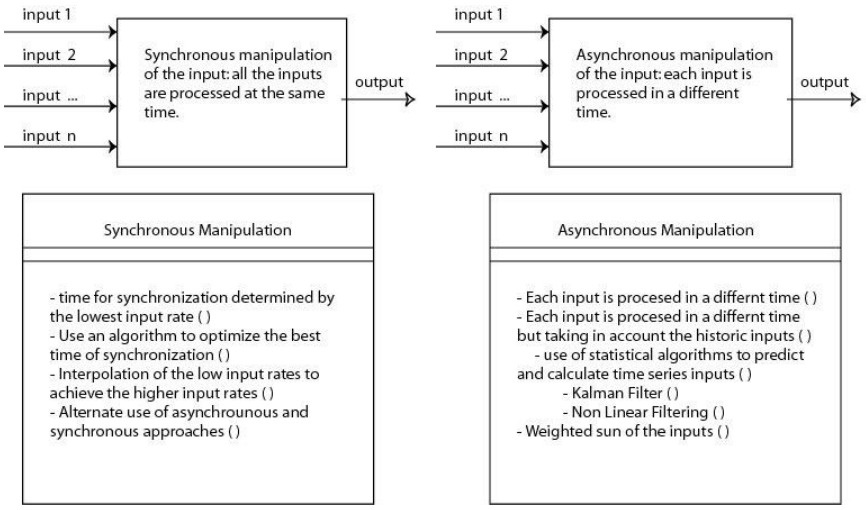
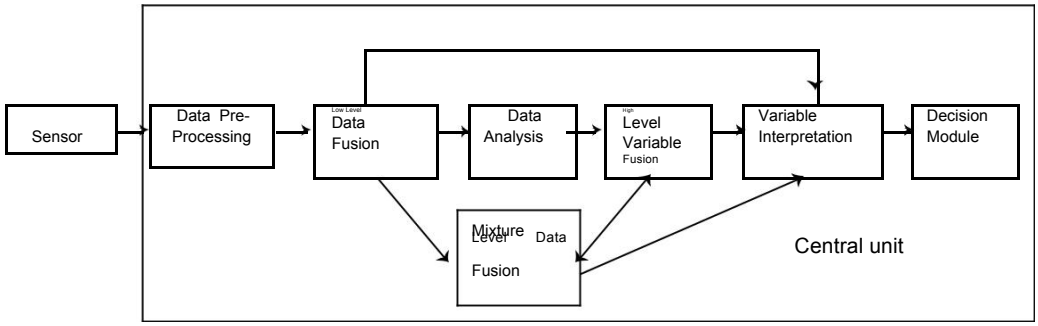
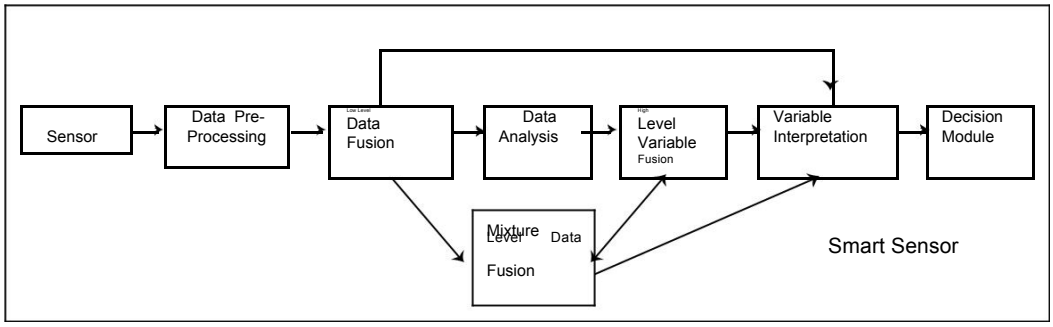


Figure 10: Input manipulation classes.

2.3.4. The DFA placement: The proposed Data Fusion Architecture is compatible with centralized and distributed approaches. Figures 11a and 11b show examples of completely centralized and completely distributed scenarios. These are the extreme cases. We can also apply it to intermediary scenarios, depending on the system's characteristics.



a) Completely centralized approach: all the data fusion functionalities are performed in a central unit.



b) Completely distributed approach: all the data fusion functionalities are available at the smart sensor.

Figure 11: Data fusion model placement.

The first scenario described in Figure 11a is an example of conventional client/server system, where different sensors are attached to a central unit and this one performs all the necessary algorithms. The decision module makes decisions about the entire system behavior. The completely distributed approach has been applied to sensors network based on smart sensors and is described in Figure 11b. In this case, the decision module applies only to each sensor's functionalities. In this case the HLDF is limited to the sensor's view of the environment.

As an example of the completely distributed approach, we can have a brief description about the use of a smart microphone network. Imagine that the smart microphone has a sound sensory device, whose capacity is dependent on the source to sensor distance. It also has a processor, memory capacity, wireless network capacity and all the available software to perform the data fusion functionalities. As a consequence, it is capable of filtering the noise of the captured sound (pre-process); performing a low level data fusion using a Kalman Filter [4] to perform time series data fusion; using an algorithm to identify human voice sound (probability of the captured sound to be human voice); calculating the sound evaluation reliability based on signal-noise evaluation; and combining this reliability value with the algorithm probability (high level data fusion). Now, it compares its result with the available results received from the neighborhood sensors. If its result is worse than some of the received results, it goes to an "idle" state and does not forward its result. Otherwise, if its result is the best human voice evaluation (highest reliability evaluation), it will continue in the "on" state and forward its result for the neighborhood sensors (propagate the information). These decisions are performed by the local decision module. This simple example shows that the data fusion architecture proposed is fully compatible with sensor networks of smart sensors.

The DFA described in UML allows the configuration of different scenarios, including the use of mobile coding. The agent can migrate to the data source to perform any type of data manipulation at the data source. This approach adds several advantages, but the decrease in network traffic is one of the most important.

2.3.5. The DFA applied to different contexts

Depending on the type of application, the data fusion system can be based on single or redundant data and variables, distributed or centralized architecture, or single or multi-agents. If the system uses different types of sensors and one agent to perform each different signal analysis, it will be based on a multi-agents system. Both centralized and distributed systems, as well as intermediary systems, can use different combinations of single and redundant data/variable and single agent or multi-agents. Although the different aspects of the system will depend on how the application is designed, the data fusion architecture can be the same using the DFA.

Figure 12 represents different instances of the DFA. Figure 12a represents the complete system with all the different features of the model. Figure 12b represents an instance of the model using the low and high level data fusion and is capable of create variables from data without making analysis of it. Figure 12c shows that in this instance the system does not use the mixture and high level data fusion. Figure 12d shows that this system can only process the data that comes from the sensors and make a decision. Figure

12e use the low and high level data fusion but needs one or more agents to analyze all the data to get variables. Figure 12f allows the creation of variables without data analysis.

These different scenarios show that all the DFA instances have in common a data source, some interpretation of the measured environment (task or variable) and a decision. This is the general architecture of a system.

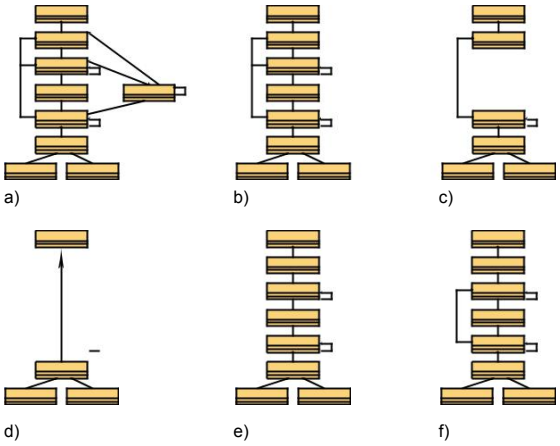


Figure 12: Different instances of the DFA.

2.3.6. Mapping different applications to the DFA: We are going to exemplify the use of the DFA in different domains and applications, such as military, robot navigation, geographical images analysis, home security system, bio terrorism detection system and health applications.

2.3.6.1 Military applications (E.g. sensing biological agents in a war area): We can apply the DFM to a completely distributed and to a semi-distributed application. Imagine the use of smart biological agent sensors deployed in a war area (completely distributed approach). Using a similar approach described in the smart microphone example or using some information dissemination algorithms applied to sensor networks such as the one described by Kulik [18], we can propagate the information of a detected biological agent (data value above a threshold) in the area until it arrives at some point where war plans can be changed or modified (E.g.: remove troops from a contaminated area). We can imagine a semi-distributed approach where each soldier carries sensors that advise him, with some grade of confidence, when some biological agent is detected in the environment (model represented in figure 12d). As a result, each soldier can perform some action according to the different situations and hierarchy. We can also have a more complex system that has some sensors that detect biological agents in the environment and others that detect modifications in soldier's bodies. A neural network can be applied using as input the data from these different sensors and giving as an output, the presence or lack of an infectious biological agent. In this case, we are using a low data fusion approach based on neural networks that directly determine a variable (without using an agent to perform data analysis). The DFA instance used here is represented in figure 12c.

2.3.6.2 Robot navigation: Imagine a robot, which based on different sensors, evaluates the environment and makes decision about its navigation. This robot can have an accelerometer sensor that detects the robot position (variable), an ultrasound sensor that determines the distance from the objects around to the robot (distance from object is another variable); this same ultrasound sensor can determine the form of the surface of the objects in the environment (variable object recognizing); Based on this variable the robot can control its wheels and navigate. All the system will use part of the DFA represented in figure 12e.

2.3.6.3 Geographical images application: Imagine an application that, based on satellite images, wants to identify different buildings in a city. The application should process the satellite images, and using different types of algorithms, try first to identify buildings areas and after, among these areas, it should try to match to a certain building image**.** In this case, the application is using the figure 12e model with different levels of high level data fusion.

2.3.6.4 Home security system: Imagine a home security application using different sets of sensors such as a sound sensor, video camera, ultrasound sensor, temperature sensor, smoke sensor, vibration sensor, and infra-red sensor. From the sound sensor we can determine if it is human voice, a broken window, an open door, or other types of sounds; from the camera we can obtain an image or presence of motion; from the infrared sensor we can detect motion, open door, open window; from the vibration sensor we can also detect motion, open door, broken window, open window; from the image we can determine whether it is compatible with a person, an animal, an object, an open door, a broken window, and so on. We can see from this example that the variable person can come from different types of sensors. All these processes have in common that the data coming from the sensor is processed, redundant data that can be fused at the low level data fusion, using an algorithm to extract the variable for each type of data and fusing the redundant variables in the high level fusion. If we fuse the variable person with the variable broken window we can come up with a new variable called intruder. This means that we are using a second level high level data fusion. We can also use a tool that based on the data from the smoke sensor (without analysis) and the image can create a new variable called fire (mixture level fusion). The entire example will use the model represented in figure 12a.

2.3.6.5 Bioterrorism detection system: Imagine thousands of biosensors distributed in a city to detect the presence of Anthrax. This application is very simple: detection of the presence of Anthrax and determining the sensor's localization. In this case, the application is using the figure 12d model.

2.3.6.6 Health application: We have developed a health monitoring system based on multiple physiologic sensors (DFA Applied to the Personal Heath Rate Monitoring System**)** that will be presented in the chapter 3. The entire example will use the model represented in figure 12e.

2.3.7. Discussion

We present a proposal of a general data fusion architecture described in a formal language of object representations (UML) that tries to represent different scenarios, specifications and features of a general data fusion system. We will show that we can employ our data fusion architecture in diverse scenarios, including different contexts and domains. As far as we know, this allows us to conclude that our model is compatible with most of the published applications using a data fusion model.

In relation to the taxonomy employed, although a common taxonomy in data fusion is something difficult to achieve, as discussed in [9] by diverse authors, we have covered almost all the possible scenarios with our taxonomy.

There are some papers in the fault tolerance related literature that use the terminology virtual sensors to represent what we call in the DFA redundant variables. Therefore, we can map the DFA to different models compatible with fault tolerance that employ the same aspect. The different approaches to choose one or more virtual sensors based on events, behavior, or other aspects, is included in the decision module. Besides this, the DFA also allows that every intermediary or final view of the sensing environment arrives at the Variable Interpretation. This aspect guarantees the presence of redundant views and provides a graceful degrading view of the environment. This means that even if the majority of the sensing units are finished, the system will still have a view of the environment, even if it is not the best. This aspect joined to fault tolerance and resilience is compatible with the ubiquitous objective.

The Decision or control module is not a formal part of a data fusion framework. In general, it does things not formally related to the data fusion process, such as to control actuators or to modify the sensing capacity. But, if this module controls the data fusion process, it should be included as part of the data fusion model. In the DFA proposed in this paper, the decision module controls the fusion process in different ways, such as changing the data/variable flow, starting new instances of the different DFA modules that depend on different types of data and/or variables and changing algorithms to process or analyze the data. We have explicitly characterized how the fusion process can be controlled by itself.

Most of the papers in the data fusion literature describe different tools to represent the knowledge, but very few papers have tried to establish a dynamic model that allows different configurations and different mechanisms to add or delete new features for the model. The DFA provides all these features to apply the same model in different contexts. Besides, we employed a well known language (UML) to represent the model. This fact can help different researches to apply the same model. This is one of the most important contributions of this paper.

The JDL functional model is the most common used data fusion model in the literature, although outside the military domain it is not well accepted. It is not well accepted because it describes features for the data fusion that makes it difficult to apply in different domains. We can map the JDL model [2] to the DFA in the following way: the level 0 corresponds to the pre-processing module; the level 1 to the LLDF and data analysis until the point where the variables are generated; the level 3 is represented in the DFA by the HLDF and variable interpretation modules; level 4 is represented by the

decision module. All the functionalities provided by the JDL model are provided by the DFA, but the DFA is a more general and dynamic model than the JDL model.

2.3.8. Conclusion

We presented a general data fusion architecture described in a formal language of object representations (UML) that tries to represent different scenarios, specifications and features of a general data fusion system. It also allows a dynamic modification of the system according to different states of the environment or of the system.

The Personal Heart Rate Monitor is from the class of network-based mobile dynamic systems powered by battery, where an application should adapt itself to different configurations of the system (data sources moving in and moving out), different states of the environment, and considering power and bandwidth constraints. The DFA proposed and applied to the PHMS showed to be a good solution to the problem of developing an application framework to manage data from different types of sensors to perform different tasks in a ubiquitous computing environment. We have mapped the data management problem of the PHMS to different domains and application and show that we can employ our data fusion architecture in diverse scenarios, including different contexts and domains. As far as we know, this allows us to conclude that our model is compatible with most of the published applications using a data fusion model and that we think the DFA can be employed with success at least in these compatible scenarios.

In this work, we have focused on the application's framework (data fusion and decisions modules) while also considering the necessary middleware and network facilities to ensure resilience to changes in available resources and changes in the environment's state. As a future work, we are going to develop a dynamic communication approach between the application framework and the middleware on the top of a network. This will allow us to achieve improved quality of service of the entire system, mainly if it has more than one application running at the same time.

Chapter 3

Body-Worn Sensor Networks for Health Monitoring

Many medical events can be diagnosed and possibly prevented by continuous clinical monitoring of patients. Several signals can be monitored. Each set of signals should be chosen according to the expected result. This end point can be cardiovascular events, infections, hormonal disturbances and so on. The set including the electrocardiogram, non-invasive blood pressure measure, pulse oximetry and vascular Doppler evaluation can detect almost all sudden cardio-respiratory events for example. Mobile intelligent clinical monitoring systems based on body-worn sensor networks provide mobility and out of hospital monitoring. It seems to be important health care equipment, to be used in the follow-up of high risk patients in out of hospital situations and to monitor "healthy" persons to prevent medical events. The characteristics of local diagnosis and actuation permit an improvement and advance in the diagnosis and emergency decision support. Besides these aspects, the connection to a central monitoring, ambulance service, assistant physicians, and the personal database and electronic record make it possible a complete integration of the system with a health care system. These goals can be achieved in a remote manner, by wire line or wireless connection.

3.1. Introduction

The evolution of wireless network devices and microchips such as MEMS (Micro Electro Mechanical Systems) brought to reality the sensor networks applications [33]. Among the different types of applications, the body-worn sensor networks application is characterized by a small coverage area, limitations on the number of sensors deployed, specific areas for sensor deployment, risk and benefit concerns, and specific sensor's functionalities and characteristics.

The sensors deployment can be classified as invasive, non-invasive and semi invasive. In the invasive deployment the sensor is inserted in some part of the organism through some invasive procedure. The non invasive deployment indicates that the sensor is deployed over the surface of the body without any type of invasiveness of the body. The semi invasive deployment is characterized by the deployment of the sensor in the oral tract (month and gastrointestinal tube), urinary tract, reproductive tract, or in the eyes, nose or ears.

All the different types of deployment should consider aspects such as usability, interaction between the sensor's material and the biological environment, power supplier and battery lifetime, biological effects of the electromagnetic waves, and the health, social and economical impact of the technology employment.

The network coverage should consider the entire body. Some sensors are static and some are mobile. For example, a sensor that is swollen and goes through the gastrointestinal tract is a mobile sensor.

3.2. Related Work

Some authors have proposed mobile clinical monitoring systems based on a multiparametric wearable computer [41] or an electrocardiogram mobile device [42]. Both systems addressed the necessity of using specialized agents and were applied to non invasive measures based on Holter ECG monitoring or other devices.

In spite of the great development of digital signal processing devices [26], some of the biological signal processing still needs some analog pre-processing. The electrocardiogram is an example. This increases the size of the system. As a consequence, the development of mobile monitoring systems needs to consider the important aspect of usability. Actually, there have been great improvements in sensor technology due to nanotechnology applications to sensors manufacturing and battery improvement [35]. Power consumption is one of the great mobile applications' problems. The cost and size decrease have provided an extraordinary improvement in the study and development of sensing systems. The state of the art in biological sensors is the MEMS (Micro Electrical Mechanical Sensors) sensors. Some of these micro sensors are based on nanotechnology [33]. It has the capacity of capturing many types of biological signals, to make some restricted data processing and to transmit it to a wireless network. Scientists have developed biochemistry sensors that measure substances like glucose, oxygen, and carbonic gas; mechanical sensors that can be used to measure biological pressures and movement amplitude; and accelerator sensors that can be used to measure position and flux, and so on. This improvement brings a world of ideas to biomedical applications of sensor systems. Other authors have discussed general micro sensors network algorithms [25][30] and applied them to biomedical applications [19], where the authors discussed general biomedical applications and specifically discussed the artificial retina.

There are many ways of actuation in biomedical applications. The systems can send messages to the user (patient or not) as an advisor or as a remembrance agent, to a health system as an emergency call or to transmit an ECG signal to the patient's assistant physician, or to control drug infusion pumps and others medical devices. But the most promising applications are in the development of artificial organs. The artificial pancreas is an example of an artificial organ. It monitors the glucose level by a MEMS sensor and when necessary, a micro pump, connected thought a wireless channel, administrates insulin. Another promising application is The Personal Physician. It would be a monitoring system to prevent or to provide early diagnosis of many diseases. As a consequence, it can advise the person when something wrong was detected, such as a malignant cell or a high blood pressure.

Many authors have addressed cooperation in multi-agent systems [20][21][22][23][31][32][27][29], but few of them have tried to establish a general biological based architecture. Some have proposed a hormone based system [38], others have proposed systems based on behavior [39] and imitation [40]. Many authors have also addressed the radio-frequency transmission problems [36], and there is an official recommendation guideline to avoid radio-frequency medical devices interference.

The Biomedical Multi-Agent Monitoring Systems [27][28] and [29] design includes some important problems in mobile, wireless and multi-agent applications.

Besides this, there are others problems related to the biomedical application such as sensors sizes restriction, sensors placement restrictions, bio-compatibility of materials, biological radio-frequency influence and so on. All the solutions to these problems must consider the biological compatibility, networks and power limitations.

3.2.1. System's view

The body-worn sensor networks infrastructure is based on the general system's architecture described in chapter 2. Furthermore, it presents the general infrastructure of the body-worn sensor networks applied for health monitoring.

3.2.1.1. A general software infrastructure for sensor networks applications

The general infrastructure built is described in figure 13. Figure 13 shows the relationship between the different blocks of software to develop an integrated solution for different types of applications. The middleware software connects the processing and analysis infra-structure to the light part of the middleware that is attached to the sensors [43, 44]. The processing and analysis infra-structure is composed of the network stack, operational system and the application software. The application software is divided into data fusion module and decision module [45].

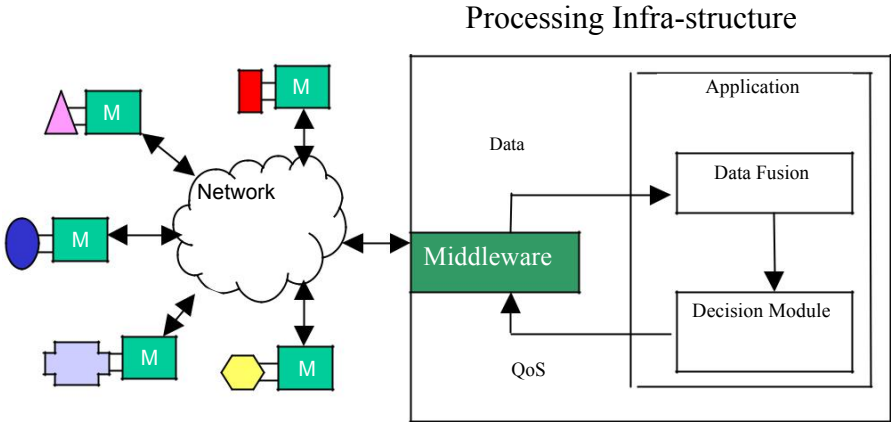


Figure 13: Network, middleware and application relationship

3.2.1.2. Body-worn sensor networks software infra-structure applied for health monitoring

The body-worn infrastructure (dotted line) includes the data management block, Objective Symptoms Review, Automatic Information Assistant, Local log and local database, sensors and actuators. The body-worn module communicates with the Web-based Electronic Patient Record, external actuators, Digital libraries and to a personal database at home (smart home).

The data management module is composed by the data fusion and decision modules. The data fusion module is implemented based on the general Data Fusion

Architecture (DFA) described in chapter 2. The data management module receives as input different types of data, such as symptoms, physiological data, and user's health (medical) data. It generates outputs such as control to sensors and actuators. It also generates outputs as queries for digital libraries. Finally, the data management module is responsible for sending data and variables for the local database and log, and for the personal database at home. The automatic Information Assistant receives as input documents from the digital libraries as a result of the queries elaborated by the data management module. It has been described in [46][47][48][49][50][51].

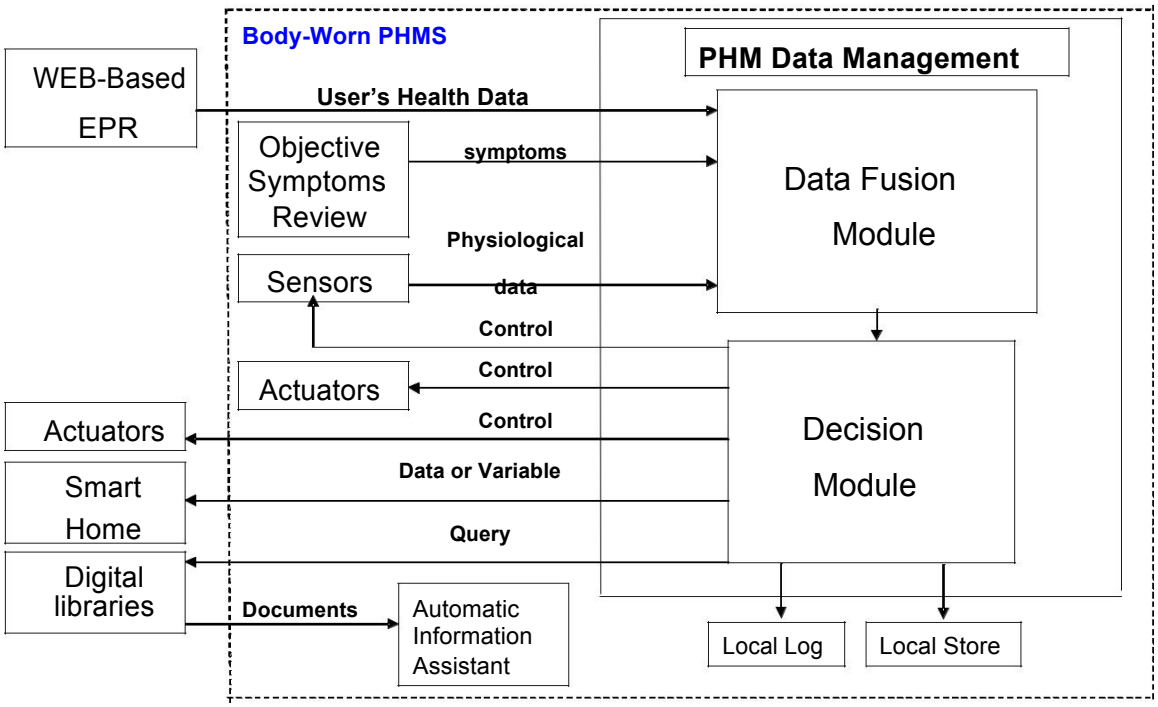


Figure 14: Body-Worn Sensor Networks Infrastructure

3.3. System Architecture

The system architecture is composed by the hardware and software components joined by the network infrastructure.

The application development: Figure 14 shows the diagram of the software tool for applications development based on a body-worn sensor networks. The application is developed based on the data fusion module and the decision module. The data fusion module is based on the architecture proposed by the authors in [45]. It is responsible for the fusion of data from the same type of sources (redundant data) and data from different types of sources. The different levels of the data fusion system are composed by different types of agents responsible to process and analysis different types of data. The data fusion system provides dynamic software to represent the changes in the environment

(body and environment where the body is placed), conditions of the sensors, and capability of the processing unit. This adaptability of the system is compatible with fault tolerance and ubiquitous computing. The decision module makes changes in the system to follow the new necessities. It is also responsible for the control of messages delivered and actuators, log and exchanges of information with other information systems (for example the electronic medical record) (figure 15).

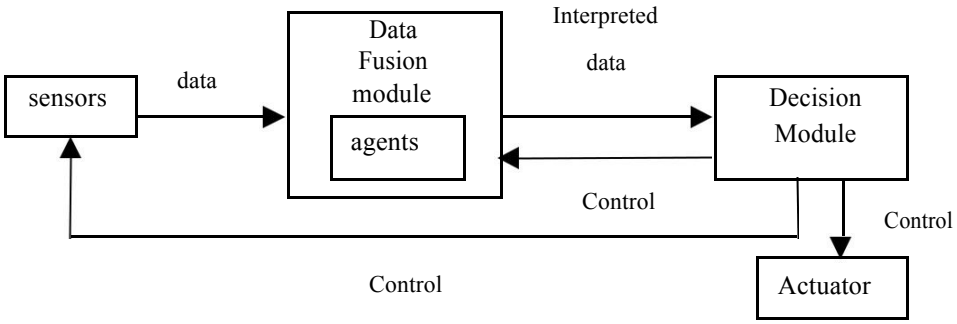


Figure 15: Application's block diagram

3.4. Network

The monitoring system is covered by a wireless network that connects the sensors and agents, agents and agents, and agents and actuators. These components are related to each other in the following manner: sensors are connected to the nearest agent that is connected to the other agents by a network of agents. These agents are connected to the actuators by a network of agents and actuators. Figure 16 summarizes these networks. In these systems data and control are based on broadcast messages.

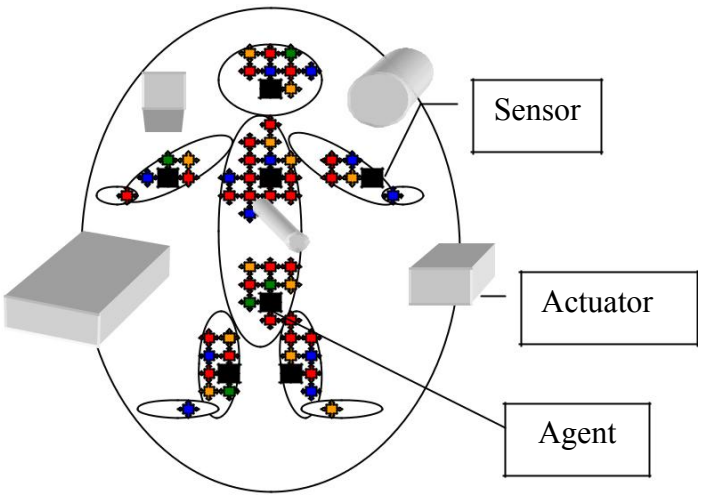


Figure 16: Network of sensors and actuators

The system design includes the integration of hardware and software components thought a wired and wireless network. This includes the development of multi agents system, health knowledge based, learning algorithms and cooperation; fault tolerance approach including power management and network management. All these aspects should be considered because the biomedical application includes certain specifics that sometimes completely change the usual approach to solving the problem.

To achieve the goal of developing a body-worn sensor network we need to use sensors, intelligent agents to manage the information and actuators to return some decision to the user or the system (figure 17). Sensors and agents have the following characteristics: type (specialization), number, body placement, and functionality. The number and types of the actuators depend on the kind of answer the system should return to the user or to itself. The communication can occur in a wired or wireless network. In a mobile environment, power and network problems should be considered in system design. Besides these aspects, in biomedical applications it is necessary to take into account the biologic compatibility of the system.

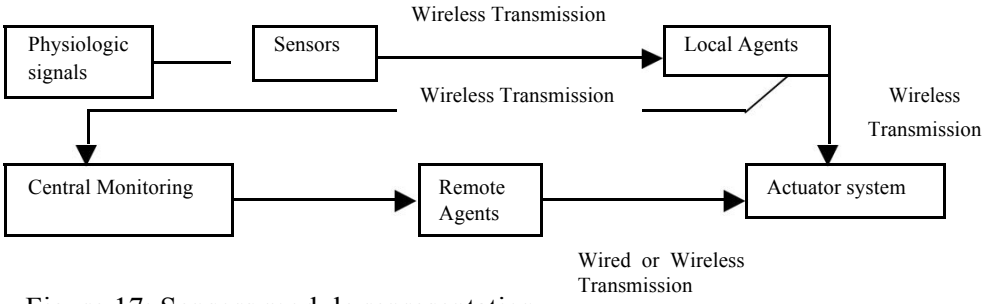


Figure 17: Sensors module representation

3.5. Hardware infrastructure

The hardware infrastructure is mainly related to the sensors and actuators. Besides this, depending on the application, it can be necessary to add processing units along the network coverage area. Intelligent agents can be embedded into these devices to process the raw data that comes from the sensors, or embedded in the sensors itself.

3.5.1 Sensors

There are some characteristics that should be considered when developing a body-worn sensor networks application, such as sensors functionalities, size and placement, biocompatibility, and lifetime. Different types of biomedical sensors can be used in the body-worn sensor networks. The sensors can be used to measure physical variables such as pressure, movement, temperature, electrical activity, and flow. The sensors can also be used to measure chemical substances in the fluids of the organism (sweat, blood, urine, saliva, etc). Some sensors deployed in the gastrointestinal tract can take pictures from the gastrointestinal mucosa, measure the PH, measure the tonus of the sphincters, or detect the presence of bacteria and parasites. These measurements can also be done in the genital and urinary tracts.

The sensors deployed over the surface of the body should be compatible with the movements of the muscles and joints. So, the size and placement are related to the size of the areas compatible with no restrictions to movement and positions. The article of Gemperle [37] describes in great details the different areas of the body surface compatible with a better usability**.** The sensors deployed using the semi -invasive approach should consider the PH of the area, the presence of microorganisms, the local habitat and the movement of the tract. The size and placement of sensors deployed in an invasive approach will depend on the local where it will be inserted (blood vessels, brain cortex, abdominal cavity, etc). Furthermore, the interaction of the sensor material with the biological environment determines the biocompatibility of the material. This is a great problem for the invasive and semi-invasive sensors.

Although great progress has been achieved though the development of cardiac pacemakers, it is a challenge to develop invasive sensors for long term use. Different concerns are related to long term use: battery lifetime, material degradation and fault, and interaction between sensor and body. On the other hand, the battery of the non-invasive sensors can be exchanged or powered easily. Some sensor characteristics adapted to body-worn sensor networks.

3.5.1.1. Types of Sensors: The types of sensors used are related to the system needs. The number of monitoring variables depends on many aspects: clinical necessity in a patient based system, number of variables in which we are interested in a preventive monitoring, and technologies employed to develop the system. Several biological variables can be measured and consequently monitored. There are biochemistry markers, fluids dynamics, gases distribution, volumetric changes, movement detection and electrical activity variables that can be measured. The problem is that many of them can only be monitored in an invasive manner. So, if the system is non-invasive it is necessary to restrict the monitoring system to the variables that are important to the context and can be measured in this way. In this case, specific biomedical devices like Electrocardiogram, ECG Holter monitoring, non invasive Blood Pressure devices, Pulse Oximetry and others, can be used isolated or in a wearable computer application. Another solution compatible with mobility is the technology such as nanotechnology-based micro sensors. The type of sensor devices has an important influence in the amount of mobility and usability of the system.

3.5.1.2. Number of Sensors: The number of sensors depends on the variable to be measured and the level of redundancy necessary to the development of a fault tolerant approach. If the variable is the Blood Pressure (BP), sensors can be placed in any arterial system (mainly arms and legs). BP measured in a specific place, is valid for the entire organism. In the case of the electrocardiogram (ECG) depends on the number of leads used. Generally, it varies from eighteen derivations to one. More than twelve derivations are used only in static ECG evaluation. Monitoring systems commonly use twelve or less derivations. Twelve derivations use ten sensors or at least ten electrodes. Pulse oximetry usually needs only one sensor. Blood flux should be measured at least in the main artery of members (brachial, femoral), head (carotid) and aorta. Oxygen measurements should be done at the artery and vein. Ideally, it should be done in the heart (right ventricle) or in the pulmonary artery. Since it is an invasive measure, it can only be measured in a mobile

manner if the patient needs an artificial cardiac pacemaker and if it uses an oxygen sensor together with its ECG electrode. If oxygen can be measured in both places it brings great monitoring advantage to evaluate circulatory function. The oxygen consumption (VO2) can be calculated. Carbonic gas (CO2) is measured in an artery (arterial concentration) or in the expired air (capnography). In both cases it uses only one sensor. Biochemistry markers like Lactate, myoglobin, troponin, glucose and others need one sensor each placed in a vein (blood contact). Heart rate can be derived from the more accurate ECG, or from pulse oximetry data. Respiratory frequency and amplitude can be measured in superior abdomen, thorax, supraclavicular area or neck with specific sensors. Electromyography (EMG - muscles electrical activity) should be done near the muscles we want to monitor. The goal of monitoring body movements and identification of exercise activities would be achieved if sensors were placed in muscles of the limbs, neck, thorax and back. The region's combination of these sensors can diagnose the type of exercise the user is practicing. Electroencephalogram (EEG or measure of cerebral activity) is captured by placing electrodes or sensors on the head. Positioning sensors should be placed in the limbs, thorax and head. The association of these positions can determine body positions standing, sitting or lying.

3.5.1.3. Sensor Redundancy: One of the most important fault tolerant problem approaches is to increase the number of sensors above the necessary numbers determined by the parameters described above. The redundant sensors should be placed in different areas of the body when the physiology permits. This would take advantage of the agent's distribution and improves the fault tolerance of the system.

3.5.1.4. Sensor Placement: Macro sensors, like specialized devices, (HOLTER, pulse Oximetry), need to be placed on the body (skin). Micro sensors need to be implanted if the user will use it for a long period of time and the technology is adequate for long use.

3.5.1.5. Sensor Functionality: A sensor as a device can be either a sensor and transmitter unit (relay sensor), or can be a sensor, transmitter and a processing unit as well. The signal processing must be done in some part of the system and can be placed at the sensors level or at others devices, depending on many factors like power and network management, sensors size, sensor placement and complexity of the processing unit. Because of power and size limitations, if the sensors have some processing function, it may involve the analog pre-processing, analog to digital conversion and digital signal processing.

3.5.2. Actuators: an actuator is any type of device that executes a system's decision. The actuator can be inserted into the body, or be on the body's surface. Furthermore, an actuator can stay near the user in the network coverage area.

3.5.2.1. Types of Actuators: An actuator could either display messages such as warnings or take some preventive action; a telephone call to an assistant physician, user family or an emergency unit; one or more infusion pumps may control the infusion of drugs; an artificial device may control some body function like an artificial cardiac pacemaker; it

can also be an artificial organ such as an Artificial Heart or an Insulin Pump (Artificial Pancreas).

3.5.2.2. Number of Actuators: In general, only one unit of each kind is necessary. An exception is the infusion pump, since the number is limited by necessity, such as venous drugs. More than one type of actuators can be used in the same application.

3.5.2.3. Actuators' Placement: This depends on the type of actuator. If it is an artificial organ it will be placed inside or on the body. If it is an infusion pump, it will be placed around the patient. If it is a display, it can be a display joined to the body (wearable display) or a TV or Computer video in the room.

3.6. Software Infrastructure

3.6.1 Agents: An agent is a software component that is responsible for processing or analyzing some type of data. For each signal or piece of information a large knowledge base to process and make a decision (actuation decision) is necessary, making mandatory the development of specialized agents, such as the Blood Pressure agent; the ECG agent; the EEG agent; the Oxygen agent; the EMG agent; the blood flux agent, the remembrance agent; and so on. A single user may be connected to several agents. The number of processing units will depend on the coverage area of the wireless channel (centimeters to meters) and whether we are using smart sensors or not. In the case of multiple processing units, it would be better to distribute a certain number of device agents on the body: at least one for each limb, one for the abdomen, another to the thorax, one for the neck and head, and another for the back. There are different approaches proposed in the literature for multiple agent systems [23].

The agent's devices would do almost all the signal processing work and network transmission. These functions would cause a great increase in power consumption. As there are as many agents as there are variables, the agents' functions are determined by its specialization. Some of the variables monitored need an intensive processing and transmission rate. There are some variables that are discrete signals and others that are continuous. Some of the signal characteristics complicate or simplify its processing and evaluation. Some agents have special functions, such as to modulate or to regulate other agents' functions.

3.6.1.1. Types of Agents: For each signal or piece of information a large knowledge base to process and make a decision (actuation decision) is necessary, making mandatory the development of specialized agents, such as the Blood Pressure agent; the ECG agent; the EEG agent; the Oxygen agent; the EMG agent; the blood flux agent, the remembrance agent and so on.

3.6.1.2. Number of Agents: A single user may be connected to several agents. In the case of micro sensor-based systems, it would be better to distribute a certain number of device agents on the body: at least one for each limb, one for the abdomen, another to the thorax, one for the neck and head, and another for the back. Since power consumption of transmission is related to the reachable area, this distribution decreases the sensors power

consumption. The use of multiple receptors for the micro sensors wireless signal may help to decrease the power consumption. The micro sensors need to send a signal to a limited area. This will also help to identify the sensor's location in case it is mobile.

3.6.1.3. Agents Functionality: The agent's devices would do almost all the signal processing work and network transmission. These functions would cause a great increase in power consumption. As there are as many agents as there are variables, the agents' functions are determined by its specialization. Some of the variables monitored need an intensive processing and transmission rate. There are some variables that are discrete signals and others that are continuous. Some of the signal characteristics complicate or simplify it's processing and evaluation. Some agents have special functions, such as to modulate or to regulate other agents' functions.

The Electrocardiography, electromyography and electroencephalography are the most power and network bandwidth consumption monitoring variable. ECG will need a pre-processing unit responsible for the signal amplification, analog to digital conversion, potential comparison, digital filtering, and digital signal processing like FFT. The agent ECG will be responsible for ECG analysis and diagnosis, heart rate variability evaluation, compression, data encryption, and local storing of the ECG data. As its processing work is intensive, and needs to be a real time decision device, we are developing a hardware agent based on a FPGA. This can bring the advantage of personalized algorithms to the patients' necessity. It can advise users about problems, make a phone call to an emergency unit in critical situations like malignant arrhythmias, or modulate other agent's functions.

The Blood Pressure Agent is responsible for the detection of abnormal blood pressure measures, detection of error measures, circadian variability and possible control of infusion pump or advising use of drugs for High or low Blood Pressures. The Pulse Oximetry Agent is responsible for the oxygen blood saturation measure, arterial pulse rate and pulse analysis, including variability and error corrections. It regulates others agents. The Blood Flux Agent is responsible by blood flux analysis and error correction. It regulates other agents. The Respiratory Agent is responsible for the respiratory rate and amplitude (inspired volume) analyses, and error correction. It can control a respiratory machine or advise the user about respiratory problems. The Heart Rate Agent compares the heart rate from ECG and the Pulse rate from Pulse oximetry. It evaluates the rates abnormalities, corrects errors and controls a pacemaker or advises the user about problems. The Gas Agent detects Oxygen and Carbonic Gas abnormalities, errors and modulates other agents. The Metabolic Agent detects lactate abnormalities and regulates others agents. The Lesion Detection Agent (Myoglobin and Troponin) detects abnormal levels of substances and advises the user about it. The Remembrance Agent reminds the user about preventive actions, times to take medicine pills and other applications. The Resilience Agent detects system failures and makes decisions based on a previous knowledge based approach. The Electronic Stethoscope Agent identifies heart sound problems and advises the user about it. It modulates other agents.

3.6.1.4. Adaptive and Dynamic Resilience Agent: This agent is responsible for the management of the services utility, having as the main objective function the power management and as secondary goal the network traffic optimization. These are very

important goals in the mobile applications in critical situations like health care services and vital signals monitoring. The mobile environment has power supply problems and possible failures with network connectivity. Many papers in this area have addressed these problems from the hardware point of view. This paper emphasizes the great importance of application participation in the power supply management. Most of the time, it is assumed that the application is necessary all the time, but the reality is that when the battery power goes down, the applications stops independently of what has been assumed before. So, instead of have a complete stop in the application, the power management supply agent can decide what to turn off before the system is completely off. This goal can be achieved considering some premises: not all the system is completely critical and we almost always are able to choose in different situations what is most important. It is clear that in certain situations we need all the monitoring aspects. But this kind of patient needs in hospital care or at least an ambulance care. The mobile care is transitory and the power supply would not be a problem.

This class of agents would be responsible for the progressive decrease in the transmissions and monitoring level through the following decisions: decrease in the frequency of data sampling; decrease in the number of sensors (first the redundancy and after the absolute and relative decrease); knowledge based priorities and situations for turning off the parts of the monitoring system; priorities of functions like data processing, interpretation, transmission, saving and compression; decisions of compressing and saving data in critical situations (like saving only abnormal data).

A distributed system can have failures in many parts. Sensor failure will not bring great problems because of the presence of sensors' redundancy. Agent failure would have the problem of losing the sensors signal from its covering area. One of the solutions would be to amplify the nearest agent network covering to receive the sensors signals, but this is not good for the power management system. But as the agent's battery can be easily changed, it can be an adequate solution. Besides this, the system is prepared to substitute agent functionality. The absence of messages from the disturbed agent is a signal to another agent to assume its' functionality. This occurs after a timeout that depends on the signal frequency of sampling. The order of substituting agents is established previously by the relationship among the agents. The agent most dependent on the other is the first on the list and so on. If the first substituting agent fails, after another timeout, the second agent tries to assume its functionality. If an agent doesn't receive many signals at some time, continuously, it concludes that it is its' own failure. It needs to advise the others that it became "sick". The sickness of an agent is transmitted to all the other agents. This is important to avoid more than one agent controlling an actuator. If the agent processing fails, but there is no transmission failure, it would be tolerated because other agents would assume its functionality. In this case it would not be necessary to amplify the nearest covering network, because this functionality is preserved. Actuators problems must be diagnosed. If this does not occur, the agent may try to increase or decrease its objective function to achieve the necessity of the system. What would not happen? This problem would not be interpreted by others agents as an agent problem, because they would receive messages from it. An actuator problem cannot be resolved by the system. The only thing it does is to detect the actuator problem.

3.6.1.5. Agents Placement: Agents can be embedded into the sensors or it can be placed in larger devices than sensors on the body's surface. This will facilitate the agent's device battery and placement change. It can be placed in any place on the body surface according to Gemperle. He and Cols have studied possible body positions for wearable devices, considering body movements and muscle activity [37].

3.6.2. Knowledge base and learning process: The knowledge-based system can be divided in two components. One is related to normality or abnormality classification of the signals. Each agent has the knowledge related to the signal for which it is responsible. The data are pre-processed in a fuzzy system based on the descriptive statistics of normal population. They give the amount of abnormality or normality of the objective function of the agent. Since the signal varies in a range, when it is not in the expected range (over or under), the system tries to make it return to normal range. The other component is an embedded mathematical modeling of the interactive process among agents. The agent decision process suffers influence from other agents. This influence is modeled in a mathematical function based on differential equations. So, the learning process is the result from the interaction between the mathematical modeling and the fuzzy logic evaluation.

3.7. Health problems related to radio-frequency transmission

There are three possible problems related to radio frequency and humans: biological effects of electromagnetic waves; compatibility of radio-frequency transmitters and medical devices; electromagnetic interference in hospital environments. The Council of Scientific Affairs recommends that wireless devices should stay as far as possible from medical devices [36]. While experimental studies have suggested that serious adverse effects related to specific power levels are great, clinical reports suggest that it is rare. The diversity of radio-frequency sources and its broad spectrum of frequencies make it difficult to predict the biological risks. Experimental studies of the long time biological effects of low radio-frequency transmission must be done.

3.8. Conclusion

We have presented a description of general body-worn sensor networks built in a network of micro sensors, agents and actuators. The system was designed considering and optimizing power management and wireless network problems to achieve the resilience and ubiquitous computing goals. Furthermore, some specific aspects related to the health and body-worn sensor networks application were considered.

We also introduced the concept of a resilience agent. This type of agent was more complex than a fault tolerant agent. It included the fault tolerant agent's functionalities added adaptability to new circumstance without failures. Besides this, it considered clinical aspects to decide about power consumption and Network optimization.

Chapter 4

Centralized Body-Worn Sensor Networks - The Personal Heart Rate Monitoring Application

4.1. Introduction

The development of wireless network technology and improvements in sensors and embedded devices have enabled the convergence of mobile applications and embedded environments. One of the requirements of mobile embedded wireless devices is that they should be available at all times. In such systems, components can be inserted or excluded without stopping the entire system. In this type of system, power and bandwidth constraints should be considered. If the system is battery powered, each component of the system has a different lifetime, which is based on the battery capacity and the device's power consumption. Power and bandwidth are limiting factors to resource use. All these problems should be addressed, to guarantee that system performance degrades gracefully as resources are diminished. A health monitoring system is one example of these systems, where, based on data that come from sensors, a view of the environment is obtained and some decisions are taken. The decisions taken can be the control of an actuator (e.g., infusion pump of a medicine) or a change in the number and location of sensors (system's reconfiguration). To achieve the last goal, some sensors can be turned on and other sensors can be turned off. To achieve the connection between the view of world and the system reconfiguration, we can classify the environment changes over time as different states. The requirements of an application may change according to the state of the environment, and as a consequence, different components (i.e., sensors) should be used accordingly. These systems are classified as dynamic systems.

A dynamic system should be available all the time. To achieve this goal, the system should be resilient, fault tolerant, and achieve all the Quality of Service (QoS) requirements specifications. Resilience is defined as an ability to recover from, or adjust easily to, changes in available resources, such as node failures, as well as changes in system state, such as a change from healthy to diseased. Resilience is a more general term than fault tolerance, which is the ability to recover or adapt to different types of failures, because it includes adaptation not related to failures, such as adapting the system to event detection in the environment. As a consequence of the fault tolerance and resilience goals, we can achieve graceful degradation, which is the ability to progressively decrease system functionality in the presence of a progressive decrease in available resources. Graceful degradation is necessary to achieve the goal of computing all the time with a certain QoS (i.e., Ubiquitous Computing). Although the definition of QoS changes in different scenarios, we will define QoS as the necessary requirements to achieve a specific goal. The specific goal can be: an adequate exchange of data in a transaction, a better use of the network bandwidth, power management approach compatible with lower power consumption, a better view of the environment by an application, etc.

4.2. Problem Statement

A multi parametric Personal Heart Rate Monitor (PHRM) is a wireless-based mobile dynamic system with the goal of monitoring the heart's condition in healthy and disease conditions. It provides a continuous monitoring of the subjects' Heart Rate. The multi parametric PHRM consists of a body-worn sensor network powered by battery and connected by a wireless network. It is designed to use different types of physiological sensors to monitor the user's heart rate: blood pressure, pulse oximeter, blood flow, arterial pulse, Electrocardiogram (ECG), Electromyogram (EMG), Electroencephalogram (EEG), and others. The use of all of these sensors will provide the evaluation of the heart rate in different conditions of the body. The heart rate varies according to body activity, temperature, blood pressure, position and so on. The PHRM evaluates the absolute and relative heart rate values in each condition.

PHRM is a mobile system. In this case, bandwidth and power constraints should be considered. To achieve continuous monitoring in different conditions, the system should have reasonable autonomy. Node lifetime is inversely proportional to power consumption. To decrease the power consumption of the system components, it is necessary to adjust the use of sensors to the current necessity of the application. As an example, if the user is completely healthy (all data are in the normal range), we can decrease the number of sensors and turn off the sensors not in use. This approach increases the lifetime of the sensors and of the system overall. When an event is detected from the current available data (e.g., a high blood pressure), the system can turn on the sensors related to the event (i.e., ECG and pulse oximeter). This is called adaptation to environment changes. The system should also adapt to loss of available sensors (adaptation to the availability of system components). As a result, if we can adapt the system to all events occurring in the environment or at the sensors level, the system can be resilient, fault tolerant, and provide graceful degradation. Although there are different adaptation definitions in the literature, we will consider in the entire text the definition described in the scenario above.

All the sensors will be connected by a wireless network on the body to an application on the top of a middleware. The discovery service will send the available sensors to the middleware, which will send the required data to the application. The data fusion framework will process and fuse the data to come up with a view of the heart's health. Based on this view, the decision system will determine the current PHRM sensors necessity (application QoS). As an example, consider that the user is healthy and that all the data that come from the body are normal. Based on these aspects, the power management policy will turn off most of the sensors. The monitor will be based on one ECG lead and a blood flow sensor. Now, imagine that the system has just recognized an arrhythmia. As a result, the decision module will request that the middleware increase the number of leads of the ECG to 3 and ask for the blood pressure and pulse oximeter data. This information will be sent from the middleware to the network. The network will schedule the new transactions and will also be reconfigured according to the new requirements of the system (assuming this is feasible, based on sensors available and bandwidth constraints). As a consequence, in the next step the PHRM application will

receive data from 3 ECG leads, from the Blood pressure sensor and from the pulse oximeter sensor.

We will develop a data fusion framework for applications of a network-based dynamic system with the following characteristics:

- Components of the system can be inserted or excluded without stopping the entire system;
- The environment changes with time and so do the physical measures from it;
- Application's necessity changes according to different states of the environment, and as a consequence, it can use or reuse components in the different states;
- The system is mobile and battery powered, so each component of the system has a different lifetime, bandwidth usage and power consumption.
- As a network-based system, bandwidth is limited and coverage area is variable.

All of these problems will be addressed from the application perspective to make the system robust to the dynamic environment.

The PHRM is from the class of network-based mobile dynamic systems powered by battery, where an application should adapt itself to different configurations of the system (data sources moving in and moving out), different states of the environment, and considering power and bandwidth constraints. As a consequence, the solution of these problems will solve the problems related to the body-worn sensor networks.

We propose a solution to the problem of developing an application framework to manage data from different types of sensors to perform a Heart Rate Monitoring application in a Ubiquitous Computing environment. In this work, we will focus on the application's framework (data fusion and decisions modules).

4.3. Solution Proposed: Implementation of Data Fusion Architecture for the Personal Heart Rate Monitor System

A network-based Personal Heart Rate Monitor (PHRM) system needs all of its components (network, middleware and application) compatible with dynamic changes in the availability of resources and changes in the environment. The PHRM system should adapt to the availability of sensors and their corresponding signals, and it should also adapt to changes in the measurements. Figure 18 represents the PHRM's block diagram. The sensors (service suppliers) and the application PHRM (in this case service consumer) are nodes of the network. The middleware is the software that connects the sensors to the heart rate monitoring application through the network. Middleware is connected to the sensors and applications. The application has two integrated modules, the data fusion module and the decision module. The application sends its QoS requirements to the middleware, and the middleware sends the sensor data to the application.

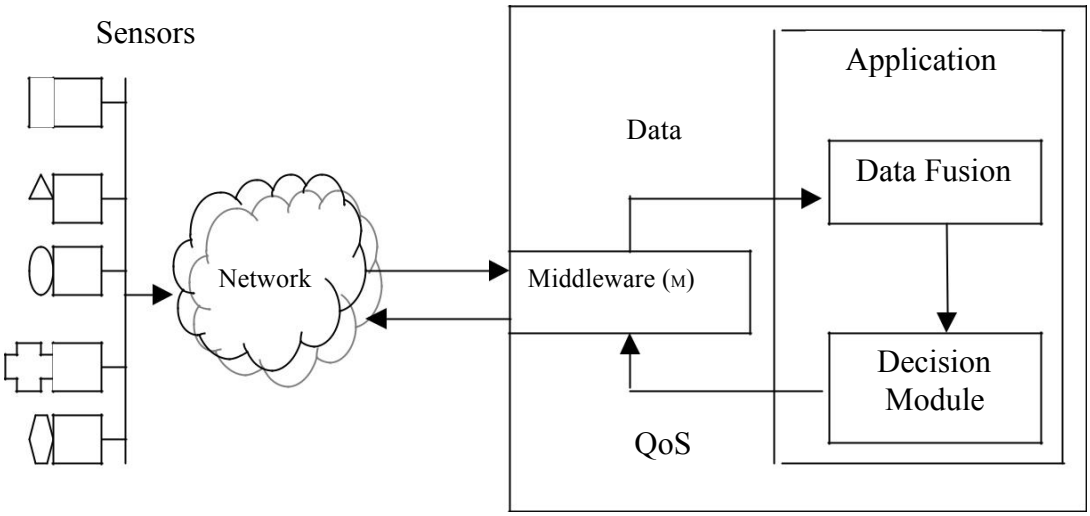


Figure 18: Network, middleware, and application relationship.

Figure 19 shows the temporal diagram of the information exchange between the system's components. When the system starts, middleware needs the QoS information from all the components (Service Suppliers, Network, Applications and from the entire system). After that, the network is configured, the transactions are scheduled and all the necessary connections (transactions) are matched by the middleware. Middleware starts to receive data and sends it to the application. The application data fusion module will process the data and pass a view of the world (environment) to the decision module. Based on this input, the decision module will determine the new requirements of the application (application QoS). Again the middleware matches the new application needs to the available resources and sends to the network the new set of connections. The network will be reconfigured and the entire process repeats in a cyclic way.

Consider an example. When the PHRM starts, the middleware should receive from the network all the available sensors and the bandwidth constraint (network QoS); from the application it receives information as to what type of sensors to connect (application's QoS). In this case we will start the system with all the available sensors; from the sensors it receives the data rate, battery power level and power consumption (sensor's QoS). If there are 2 or more applications in the system, the system should send to middleware the relative priorities of the applications (system's QoS). Based on all this information, middleware will match the services (sensor's data) available to the application needs, to create feasible sets. Feasible sets of sensors are defined as the set of sensors that achieves the current QoS requirements of the application. Middleware will choose one of these feasible sets to optimize the tradeoff between application performance and resource use, and it sends the chosen feasible set to the network. The network will schedule the transactions and set up all the necessary connections. In this case, the application will receive all the available data from the sensors. Now the data fusion framework will fuse the input data and give to the decision module the grade of normality or abnormality of each one of the signals used as input. Based on these evaluations, the decision module will determine the new necessity of the application. If all the data are normal, it will decrease the number of sensors to a lower level of monitoring. Again, the middleware matches the new application needs to the available

sensors and sends to the network the selected feasible set (now only part of the available sensors). In the next step the network will reschedule the transactions. Now the network needs to be reconfigured to close the connections that are not necessary anymore (at least temporarily). The data fusion receives the new set of signals and the system continues in a cyclic process.

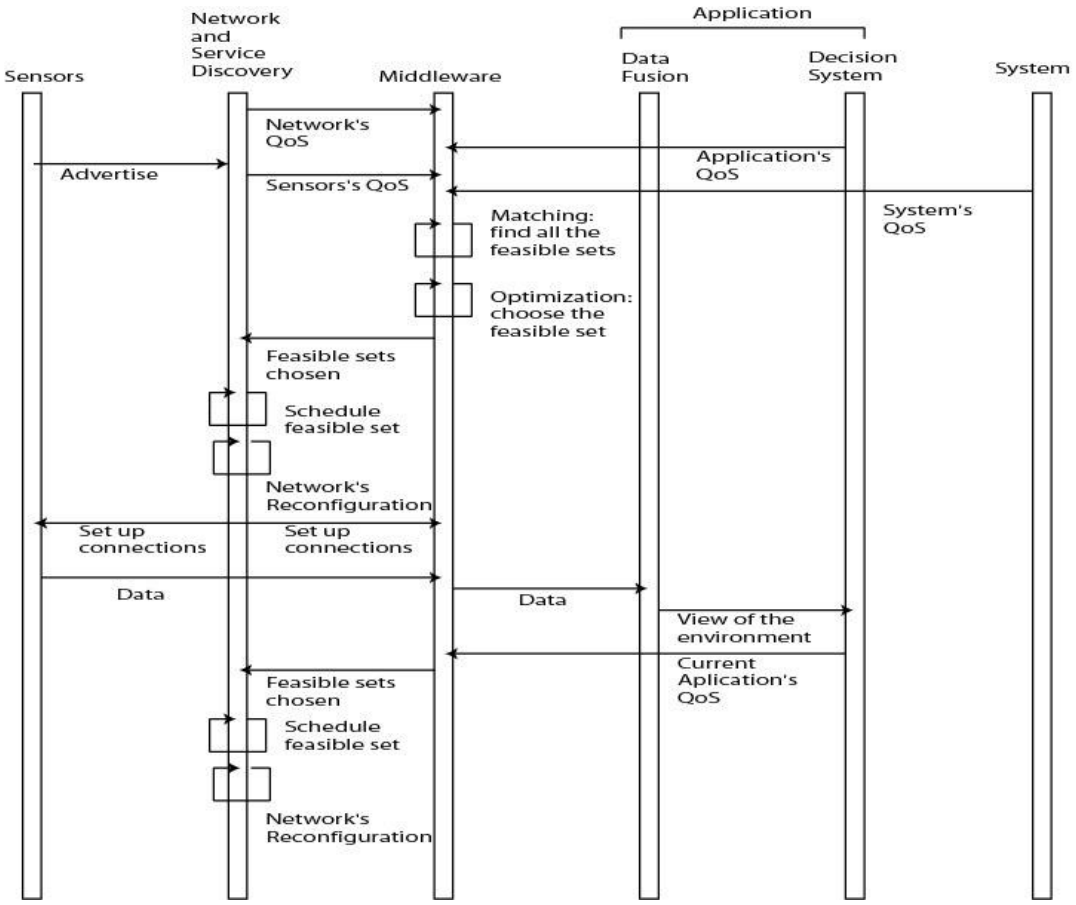


Figure 19: Temporal diagram showing the interaction among the sensors, the network, the middleware, the application and the system.

We have described the data fusion framework and the decision modules in chapter 2. The Data Fusion framework manages data from different types of sensors to perform the PHRM in a Ubiquitous Computing environment. Figure 20 represents one instance of the proposed data fusion framework applied to the Personal Heart Rate Monitor.

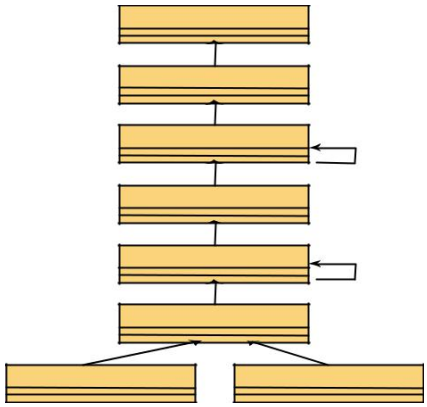


Figure 20: DFA instance applied to the Personal Heart Rate Monitor

4.4. Data Fusion Applied to the PHRM

The heart rate is the result of the electrical activation of the heart, resulting in mechanical contraction of the cardiac muscle. The ventricular contraction results in dynamic changes in blood flow and blood pressure, as well as deformation of the arterial wall (arterial pulse). As a consequence of these physiological aspects, we can measure the heart rate directly by analyzing the cardiac electrical activity or we can measure it indirectly by analyzing the homodynamic changes. The Electrocardiogram (ECG) is the graphical representation of the electrical activity of the heart. From the ECG analysis we can obtain the heart rate and an ECG diagnosis. The latter includes, among other aspects, a determination of whether the ECG is normal or abnormal and what abnormality is present (ischemia, infarct, arrhythmia, cardiac chambers enlargement, and other abnormalities). The heart rate monitor is a simple device that is based on one ECG lead to determine the heart rate (commonly used in exercise evaluations). The homodynamic changes can be evaluated through the blood flow and arterial pulse. Different cardiovascular and respiratory exams use the blood flow and the arterial pulse variables to determine the blood oxygen saturation (Pulse Oximeter), Blood Pressure (Blood pressure device), and Blood Flow (Doppler). As a consequence, these devices can obtain an indirect measure of the heart rate among other variables.

As defined before, data fusion is classified in three different levels: low level data fusion, high level data fusion and mixture level data/variable fusion. The Blood pressure signal can be fused before any analysis (low data fusion). After the Blood pressure signal is analyzed, we can come up with two different variables: the blood pressure and the heart rate. The heart rate can be measured not only through the analysis of a Blood pressure signal, but it can be also measured from an ECG, blood flow or pulse oximeter

analysis. As a consequence, we can have the variable Heart Rate from different types of sensors (redundant variable from different types of sensors). If we want to fuse all the heart rate variables, we are going to use the high level variable fusion approach. So the data analysis defines the low and high level of the data fusion system. In the heart rate monitoring applications we do not fuse data with variables (mixture level data/variable fusion), but in some applications it is possible.

Figures 21, 22 and 23 show the application of the data fusion framework presented before, applied to the multi parametric PHRM application. Figure 21 shows an example of the use of an individual sensor, pre-processing, low-level data fusion and data interpretation classes. The different types and positions of the ECG electrodes generates after a pre-processing period that includes among others functions the signal amplification; use of a high pass filter (0.5 Hz); use of a low pass filter (25 Hz); use of a notch filter (60Hz); multiplexing; and use of an analog to digital converter. The multiplexing function generates the four types of ECG with different number of leads. The low level ECG fusion module only forwards the data. This occurs because the management of the redundancy in the ECG data from different leads needs a signal interpretation. The ECG interpretation module will be presented in a different topic and is an important contribution of this work. It is responsible for recognizing the ECG waveforms and evaluating the grade of normality or abnormality of each wave independently and the sequence of waves in a continuous monitoring. The ECG signal interpretation generates two different variables, the heart rate and the ECG diagnosis.

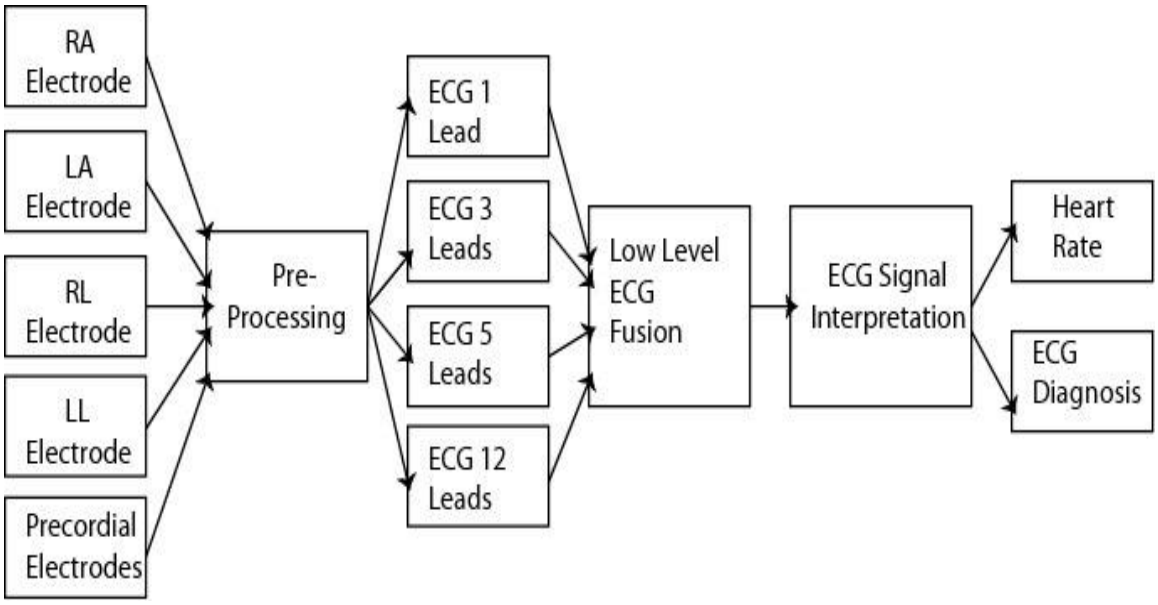


Figure 21: Pre-processing, low-level data fusion, and data analysis.

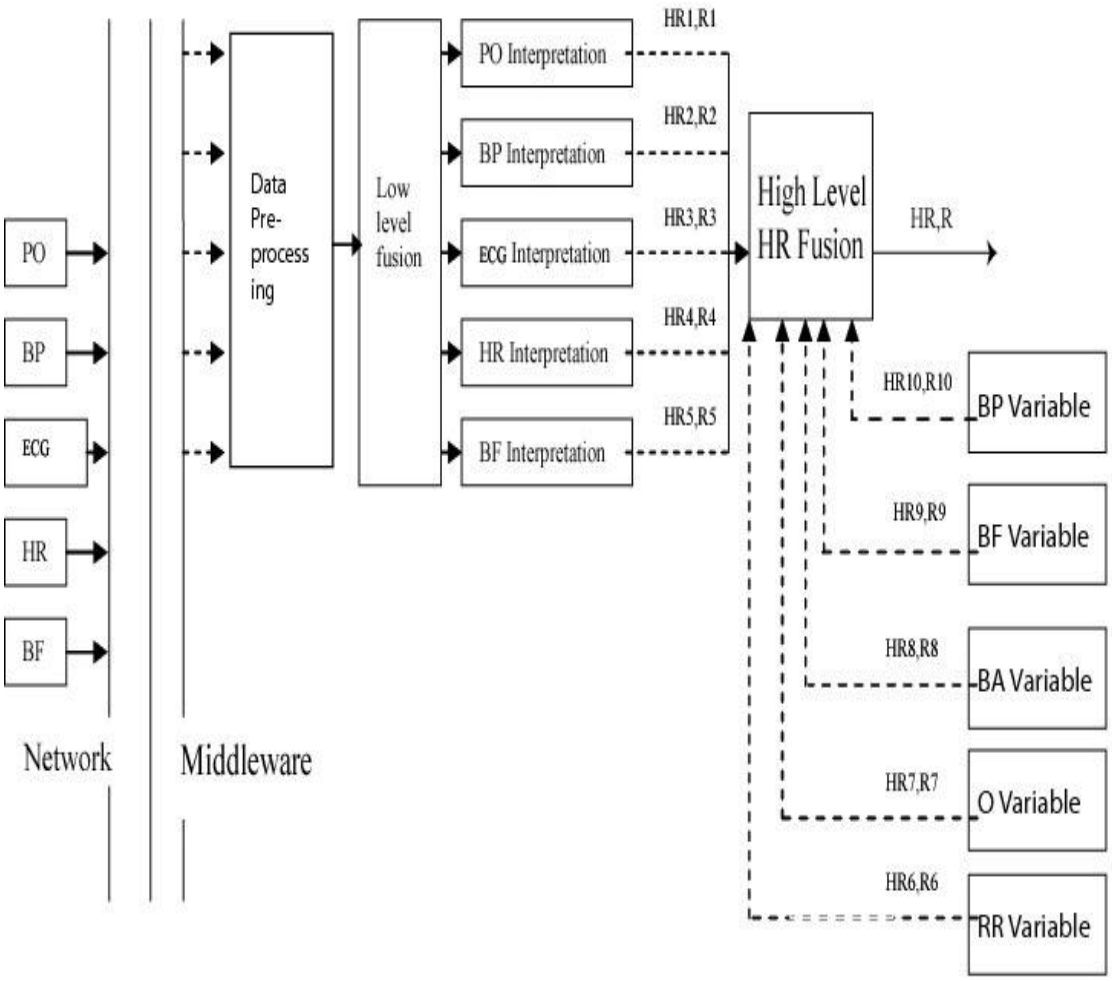


Figure 22: Heart rate variable redundancy.

Figure 22 shows the heart rate variable management. We can measure the variable heart rate from different types of sensors. We can measure it from a Pulse Oximeter sensor (PO), from a Blood Pressure sensor (BP), from an ECG system (ECG), from a specific heart rate measuring device (HR) and from a Blood Flow sensor (BF). Each sensor has a different reliability to measure the heart rate. The ECG system has the highest reliability and the Pulse Oximeter has the lowest reliability. Based on power constraints and the system's request for certain reliabilities, the middleware can request from the network a specific set of sensors and present that data to the application. If the application has requested a HR measure with the highest reliability, middleware would present the ECG data to the application (which has reliability of 1). The ECG signal would be pre-processed. Then, an ECG interpreter module (ECG Interpreter) would analyze the ECG data and provide to the Heart Rate high level fusion (HR Fusion), the

variable (HR) with a reliability value (R), the ECG Diagnosis high-level fusion (ECG-Diagnosis Fusion), and the variable ECG Diagnosis with a reliability value (R) (figure 21). Depending on the system's state, the application can receive and use data from more than one type of sensor.

Redundant variables come from different sensors with different reliabilities and from different locations. Variable is defined as a triple composed by the measured variable, the sensor reliability to measure that specific variable and the sensors placement. For example, the measurement of the heart rate using one Pulse Oximeter placed on the left arm (LA) has a reliability value of 0.7. So, the triple would be represented as (PO-HR value, 0.7, LA). Some of these variables can be objective variables (numeric measures) and others can be subjective (linguistic variables). For example, the measure of the heart rate based on an ECG is an objective measurement. Subjective variable is inferred from a set of interpreted variables. They are subjective assumptions on a specific condition, such as whether the heart rate is high or low based on the knowledge that the blood pressure is low. Subjective assumptions are defined as a triple composed by the variable V, its reliability measure R (which is very low due to subjective evaluation) and zero, because it is not related to any location (V,R,0).

At the High Level HR Fusion module, we can fuse redundant disposable heart rate data (HR1 to HR5) and the very low reliability HR predicted data from Blood Pressure, Blood Flow, Body Activity, Oxygen and Respiratory Rate Interpreted Variables (HR6 to HR10). So, based on the quantity and quality of disposable data, the HR fuser module can perform different procedures.

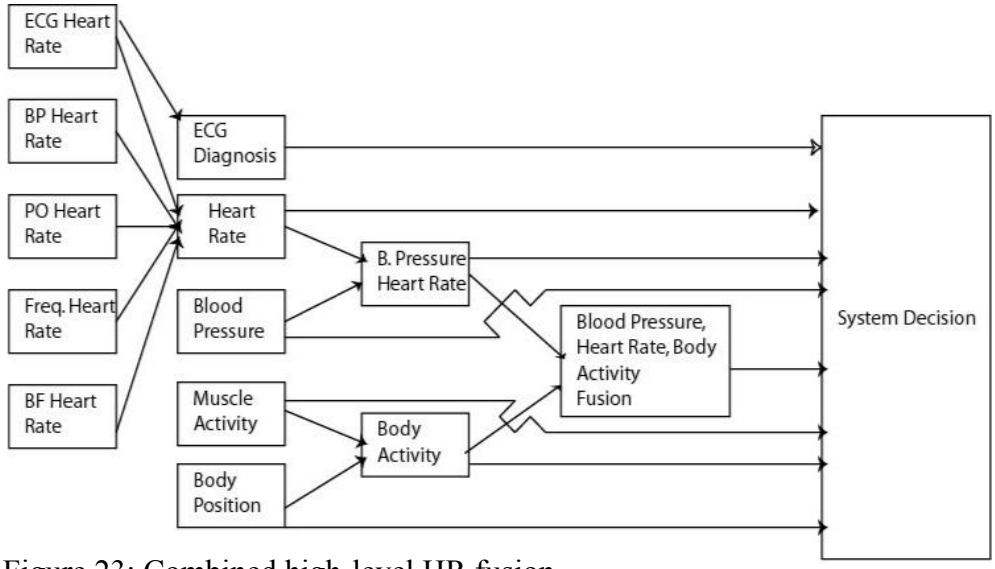


Figure 23: Combined high-level HR fusion.

Figure 23 shows the combined high-level heart rate fusion. The heart rate is combined with the blood pressure and results in the Blood Pressure-Heart Rate fusion variable. At the same time, the Muscle Activity is combined with the Body Position resulting in the Body Activity variable. The next step is the fusion of the Blood Pressure-Heart Rate variable with the Body activity variable resulting in the Blood Pressure-Heart Rate-Body Activity variable. The output of each fusion level goes to the System Decision

module. This provides input redundancy to the system decision and guarantees that some decision can be taken in the case of failure of a fusion module.

Each fusion level evaluates the current heart rate value, isolated or joined to other variables. The current heart rate value has two types of analysis: static and dynamic. The first is related to normal and abnormal interpretation of the variable according to each body's situations. The dynamic evaluation is related to changes in the body's state. The question is whether the amount of variable increase or decrease is normal or abnormal. To achieve the static and dynamic evaluations, the interpreter should use a reference table, where heart rate is correlated with the subject's age, to evaluate the expected heart rate basal level. This is very important because the basal heart rate of a child 1 month old is near 140 beats per minute, while the basal HR value of a man 60 years old is around 60 to 70 beats per minute. These reference values are useful in static evaluations, but HR Interpretation should consider dynamic situations. In these cases HR interpretation will occur in each fusion level, if they are available (Blood Pressure-Heart Rate, Blood Pressure-Heart Rate-Body Activity) to obtain a heart rate interpretation based on the body's state. As an example, we can consider the case where the body is exercising. In this case, the HR interpretation should consider whether the heart rate is compatible with this situation.

4.4.1 Techniques for PHRM Data Fusion

In this section we discuss some techniques that we will use to fuse the data in the low-level data fusion and in the high-level variable fusion.

4.4.1.1 Techniques for low-level PHRM data fusion

The low-level PHRM data fusion includes the fusion of data from one sensor, fusion of time series data from each redundant sensor's data, and fusion of redundant sensors' data.

a) Fusion of data from one sensor: In this case we will forward the triple: data value, reliability value and sensor's location. No data fusion will be applied, the value is passed through.

b) Fusion of time series data from a redundant sensor's data: For this operation we use a Kalman Filter [4] [14]. We are trying to determine an estimate of a variable (current estimate) from measured data and previous estimates.

State Estimate actualization:

Current estimate = previous estimate + Kalman gain * (current measure – previous estimate) (1)

Kalman gain (K) = variance of the previous measure/ (variance of the previous measure + variance of the current measure) (2)

c) Fusion of redundant sensors' data: Our approach will be based on descriptive statistics of the data. If the data has a Gaussian distribution, we can use average ± 1SD. Otherwise, we will use median ± 10 percent. These ranges are based on normal expected differences in measuring the same variable along the body. We should work with the following tradeoff: if we increase the range, we will not detect problems immediately after they occur, but we will have excluded most of the sensor noise and measuring error problems. If we decrease the range, we will be able to detect problems as soon as they occur, but we will have problems excluding noise and measurement error. Therefore, this range depends on the sensors' reliabilities and on body's signals variability.

The redundant sensor data should be compared if it is in the range (average ± 1SD or median ± 10 percent). If all data are in the range, the system will forward the average or median. If one or more redundant data are out of the range, they are called outliers and the system should forward the average or median plus the outlier values and address of the outliers. The average or median will be analyzed by its specific agent. The outliers should be studied to differentiate if there is noise due to sensor error or if they are correct measures indicating an abnormality. If the outliers indicate abnormalities, we can turn on redundant sensors (if available) that cover the same area (increase area coverage reliability). Another approach is the use of different types of sensors that can give an idea if there are other abnormalities in the focused area (increase variable reliability). So, if the redundant sensors' data of the same area are in the range and there are no other abnormalities in data of the same area, we conclude that the outlier was caused by sensor error. Otherwise, we should increase the reliability of the area and probably of the system because of what the outlier may indicate.

It may be helpful to consider the blood flow example. Imagine that we have 6 redundant blood flow sensors. One sensor is placed at the extreme of each limb, another sensor is placed at the abdominal aorta artery and the last one is placed at the carotid artery. In general, the same measures in different places of the body vary by less than 10 percent of their values. This means that all the data that come from the blood flow sensors should be in a 10% variability range. Now imagine that the data that come from 5 of the sensors are in the range and the sixth value is below the others and out of the range. We should evaluate whether the outlier is a real measure or whether it is an incorrect measure. If it is a real measure this means that the blood flow for that place decreased (problem detected). If it is an incorrect measure, it should be discarded. We may use data from the blood pressure sensor at the same locality as the outlier measurement come from to determine whether the blood flow data is accurate at that location. If the blood pressure value is below the range of the other blood pressure values, the blood flow data is a real value and the patient has a problem.

This approach is very different from other applications, in which several methods have been employed to decrease estimation error by eliminating data outliers, i.e., data that lie outside a specific confidence interval like 0.95 or 0.99. In our case, an outlier should be investigated because it can be an earlier signal of a disease.

4.4.1.2. Techniques for high-level PHRM variable fusion

There are many different situations that can occur in high-level fusion. We will look at each one in turn.

a) No sensor data (HR1 to HR5): This means that we do not have measured heart rate data available. If there are available interpreted variables, HR Fuser can predict HR value using an approach based on a knowledge-based probability system. This prediction has a very low reliability value and is very subjective. We should compare this predicted HR value with the last time series fusion HR, R values stored. For example, we can predict the heart rate value based on the interpretation of the body activity. If the body activity interpretation says that the body is exercising, we can predict the value of the heart rate, but with a very low reliability value. If none of the interpreted variables (Blood pressure, body activity, temperature, etc.) are available, the HR Fuser cannot forward any HR, R values.

b) Only one type of data from sensors (HR1 to HR5): If only one data comes from the sensors (HR1 to HR5), HR Fuser will use HR, R from a single data source. It should compare this value with the last time series fusion HR, R values stored. For example, if the HR fuser receives the heart rate from only the ECG (ECG_HR), it will use only this value to evaluate the heart rate. To decrease the error, we will use the time series fusion approach.

c) High-level HR Fuser receives data from redundant sensors of the same type (two or more data of the same type HR1 to HR5): This can occur in the case of the low level HR fuser forwarding the median and one or more outliers. This means that the outlier is not in the normal range expected (if the data has a parametric distribution, average ± SD, otherwise median ± 10%). HR Fuser will evaluate whether the outlier data is an incorrect measure or an abnormality. If it is an incorrect measure, HR Fuser will use the median or average and discard the outlier. If not, it will send the information of an abnormal HR measure in the sensor location to the decision system. Based on this information the decision system will make a decision based on the sensors' covering area. For example, if the HR fuser receives two values of the HR from the blood flow sensors, one is the heart rate average of all blood flow sensors in the range (median ± 10%); the second one is the HR measure from the right arm blood flow sensor that is an outlier. The HR fuser must decide whether the outlier is an incorrect HR measure or if there is any problem in the right arm. To achieve this goal, the system should compare all the available data from the right arm. If there are no other available data in that area, the decision system can ask middleware to turn on sensors in that area. Otherwise, if all the different data that come from the right arm are normal, the system considers the HR measure from the right arm blood flow sensors as an incorrect measure and discards it. If all the different data that come from the right arm are abnormal, the system considers the HR measure from the right arm blood flow sensors as an abnormal value, indicating that some problem has occurred in the right arm blood flow homodynamic.

d) Fusion of redundant variables' data: If HR Fuser receives data from different types of sensors (HR1 to HR 5), it will use one of the following procedures: it can choose the highest reliability data or it can fuse the redundant variable by applying equations 3, 4 and 5 described below. In addition to redundant data fusion, the HR fuser can perform time series fusion based on mathematical approaches like Kalman Filter (equations 1 and

2). These two results (Kalman filter predictor and current data fusion) should be compared. If the error (difference between the results) is not large, HR Fuser will forward the HR, R obtained from the redundant data fusion approach. For example, depending on the QoS heart rate reliability value requested by the decision system, middleware can provide data to the data fusion module from more than one data source (e.g., heart rate from blood pressure and pulse oximeter). In this case, the high level HR fuser has to manage redundant variables from different sources. As the reliability values of both data are similar, we can use the approach to sum them up using equations 3, 4 and 5. If the HR fuser receives data from the ECG and Blood pressure, the reliability of the HR from the ECG is higher. In this case we can use the approach of taking the highest reliability value (in this case, the HR from the ECG).

When we have several measures from a variable with different reliabilities, we must devise a way to combine or fuse them in a numeric way. To fuse objective (numeric) data, we can use the following approach: as each variable has a different reliability, we should weight its value to use a statistical approach to fuse the data. Equation 3 is a normalized approach to fuse objective data.

$$\frac{\sum_{i=1}^{n} (VVi * RELIABILITYi)}{n} \tag{3}$$

where n is the number of redundant variables to be fused, VV is the Variable Value and RELIABILITY is the VV reliability value (weight).

The objective of fusing linguistics data can be achieved using tools that can transform linguistic data to numeric data. To achieve this goal, we should be able to make some correlation between linguistic terms and numeric data. This can be done using probability theory, set operations or membership functions of Fuzzy Logic. Then we follow the same approach for the objectives measures.

$$\frac{\sum_{i=1}^{n} (VLVi * LRELIABILITYi)}{n} \tag{4}$$

where n is the number of redundant linguistic variables to be fused, VLV is the Variable Linguistic Value and LRELIABILITY is the VLV reliability value (weight).

If the system has both types of variable values, we can make a fusion of objective and subjective values using the equation 5.

$$\frac{\left(\frac{\sum_{i=1}^{n} (VVi * RELIABILITYi)}{n} * \sum_{i=1}^{n} RELIABILITYi\right) + \left(\frac{\sum_{i=1}^{n} (VLVi * LRELIABILITYi)}{n} * \sum_{i=1}^{n} LRELIABILITYi\right)}{\sum_{i=1}^{n} RELIABILITYi + \sum_{i=1}^{n} LRELIABILITYi} \tag{5}$$

Equation 5 is a normalized sum of the objective and subjective values. This approach was chosen because it takes in account the normalized weight of each variable in the final variable value.

The output of the High Level HR Fusion is the Heart Rate fused data with its resulting reliability (HR, R), or current heart rate value. This variable will be interpreted and used by the Heart Rate Decision System, or it can be one of the inputs to another High-Level Variable Fusion (figure 22).

e) Fusion of different variables: We will consider the example of a body monitoring activity system. This system is based on two different types of sensors: accelerometers and electromyograph. Accelerometers can give an idea about activity, but they provide higher reliability in characterizing position. The electromyogram (EMG) is the muscles electrical activity recorded by an electromyograph. The EMG can give an idea about the body's position, but it provides higher reliability in evaluating muscle activity. The body's overall activity is related to both position and muscle activities. A person can be lying down and exercising and standing up without exercising. So, to determine the state of the body, it is necessary to create a complex system to represent the knowledge of both variables body position and muscle activity.

Figure 24 is a proposed approach to solve the body's activity fusion problem. It is based on a finite state machine that represents different states of the computation. The accelerometer sensors can give three basic states: laying down (LD), sitting (S) and standing up (SU). Depending on the electromyogram evaluation we can have 5 different levels of muscle activity: non exercise (NE), mild exercise (ME), moderate exercise (MoE), strong exercise (SE) and muscles fatigue (FE). The state machine has three starting states representing the three main states of the body's positions. From each combination of position and muscle activity we have transitions that generate an output to the system. The transitions are represented by the muscle activity evaluation and position. Let us consider the following example. The subject starts the evaluation lying down with no exercise followed by a stand-up position with mild exercise and then the laying down position with no exercise. We start the automata at the LD state, go to the LD-NE state, go to the SU-ME, and go to LD-NE state. At each transition between the states, the automata generate output to the system representing the level of the body's activity.

The level of the body activity will be combined with the current value of the heart rate (output of the HR high-level fusion) to have a new evaluation of the heart rate based on the change in the body's activity. In this case, the heart rate evaluation will be based on the relative change of the heart rate in comparison to the last body activity level. For example, if in the LD-NE state the heart rate is equal to 60, when the body changes to the SU-ME state, we should evaluate if the amount of increase in the heart rate value was what was expected or not.

The management of accelerometers and electromyography data is complex and needs the development of different algorithms to analyze the sensors' data. This part of the system is important to evaluate the heart rate. To achieve this last goal, we are going to simulate the different states of body activity.

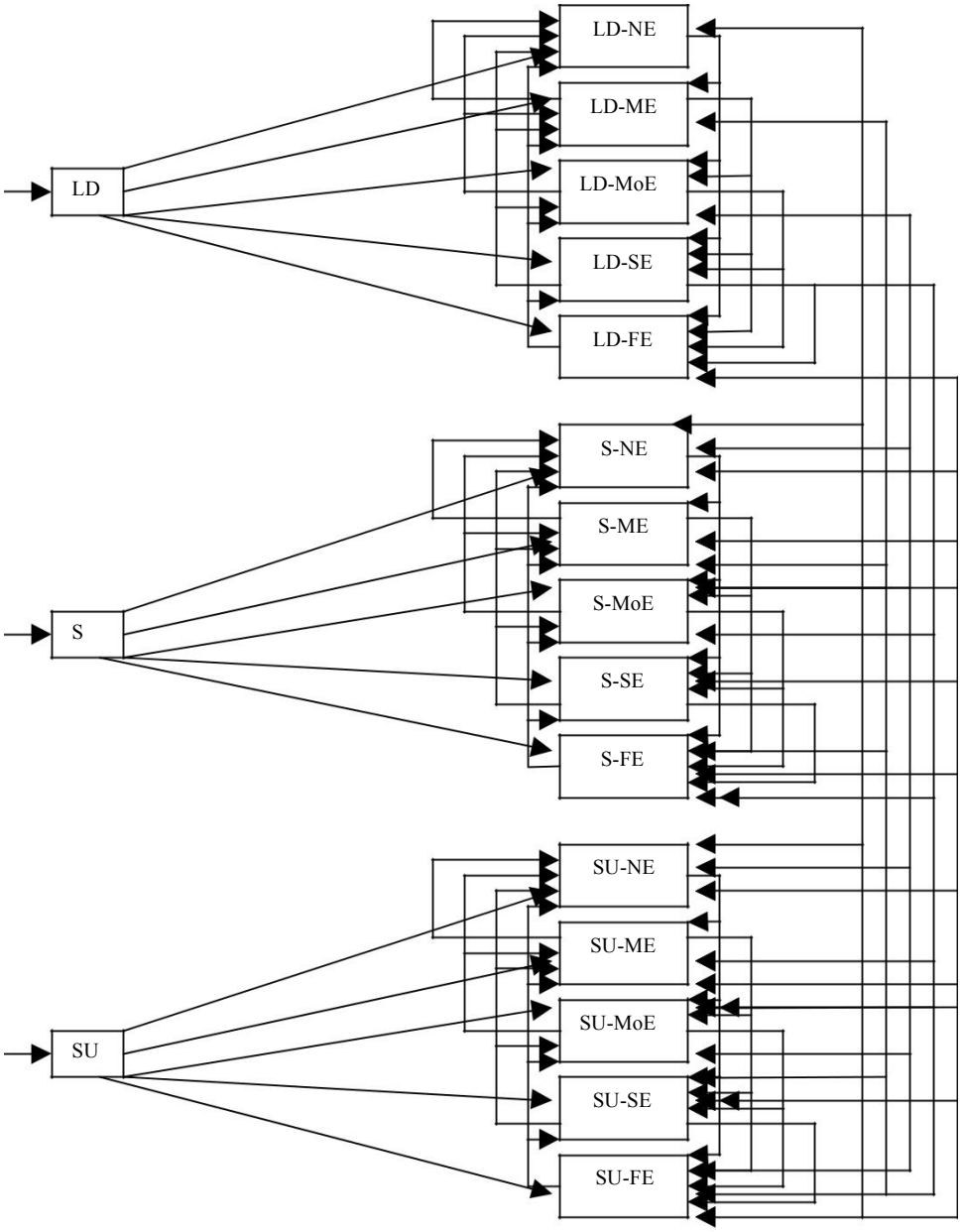


Figure 24: Variable fusion- the position and muscle activity

f) System's fusion: In the case of the heart rate monitoring system, one of the main goals is to evaluate if the heart rate is normal or abnormal according to different body conditions. Normal, in this case, means that the results are not compatible with diseases and are related with a low risk of developing new ones. So, considering the functions covered by the monitoring system, it should fuse all the information available to decide if the heart rate is normal or abnormal. More generally, if we evaluate the cardiovascular system through the ECG, blood pressure, blood flow and heart rate; the respiratory system through oxygen measures, respiratory frequency and volume evaluations; muscle activity through the EMG; metabolic evaluations through glucose and lactate measures; and cerebral activity through EEG evaluations, we should evaluate the output from all the systems available to decide if the organism is normal or not. So, the overall world reliability depends on the system reliabilities, which depend on variables' and consequently sensor's reliabilities. As we are not going to work with all these variables, the world view will be based on the variables available.

To achieve the system fusion goal, we will use fuzzy logic. Fuzzy inference theory has been applied in many different domains [17]. The following example describes how we can integrate subsystems' information to achieve the world view of normality or abnormality. We can build a membership function based on Fuzzy Logic to evaluate the subsystem grade of normality and abnormality (figure 25). We can divide the normality and abnormalities into different classes that represent different levels. Figure 25 represents the membership functions of these classes. The Y axis represents the grade of pertinence (μ) and the X axis represents a set of classes: 1= Strong low abnormality; 2 = moderate low abnormality; 3 = mild very low abnormality; 4 = low indecision; 5 = normal range; 6 = high indecision; 7 = mild high abnormality; 8 = moderate high abnormality; and 10 = Strong high abnormality.

.

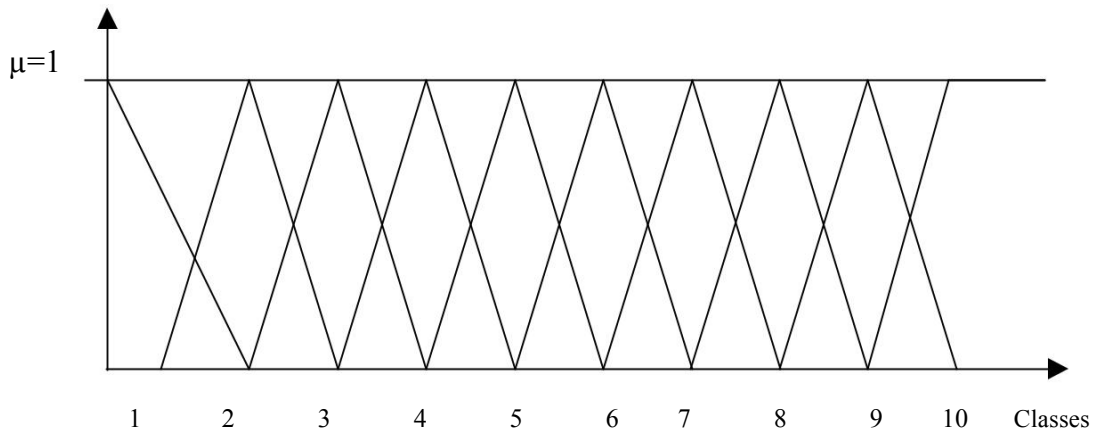


Figure 25: Pertinence function of normality and abnormality classes.

The values in the X axis that determine a class's limits depend on the variable to be analyzed. In the Heart Rate example, we can define that a heart rate below 30 beats/min is class number 1, a heart rate between 20 and 40 would be class number 2 or that a heart rate above 180 is class number 7. Let us imagine the patient has a heart rate of 25. If we go to the fuzzy membership function we would find a $\mu = a$ to class 1 and $\mu = b$ to class 2. All other classes would have $\mu = 0$. Now we would have to use an aggregation function that can be as simple as a rule based function, to fuse these two classes in the range between -1 and +1 that represents a new pertinence function of the two classes, normal and abnormal. Figure 26 represents these classes. Based on this, the defuzification process will generate a number between -1 and 1 that represents the pertinence function of the classes normal and abnormal.

The evaluation described above is related to static analysis. We should make another kind of evaluation that is related to dynamic response to different stimulus. The difference is that the X axis in this case would represent the percentage of decrease or increase from basal levels after the stimulus. This kind of analysis is more accurate, but we must have experimental results to build the classes.

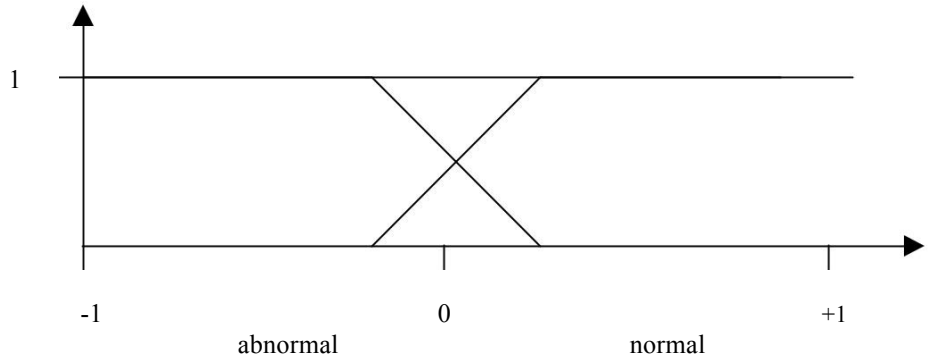


Figure 26: Pertinence function of normal and abnormal.

4.5. Evaluate Current Status

As a consequence of the fuzzy logic pertinence function approach, each variable will be classified as normal or abnormal with different levels of pertinence (figure 26). We will concatenate the variable value (HR) with its reliability R and its evaluation (Evaluation). Therefore, the result of each data fusion level is a triple (HR, R, evaluation). This information goes to the Evaluate Current Status module to fuse all the evaluations from the different data fusion modules (figure 27). In this module, the variable evaluation will be classified as an event using a rule-based approach. The event will be classified as a High Risk (HR) event, a Moderate Risk (MR) event, a Low Risk (LR) event or a Negligible Risk (NR) event.

The output of the HR Evaluate Current Status module is the triple (IVHR, R, Event), i.e., Interpreted Variable Heart Rate (IVHR) with a Reliability value R and with an evaluation (Event). This output reflects the grade of normality or abnormality of the heart rate variable from each level of the data fusion module. Depending on the triple (IVHR, R Event) evaluation, the system can send the result of this evaluation to an

actuator or to the Decision module. We can use different types of actuators, such as a display device, a phone line, the Internet (e-mail), or a Voice Adviser System using previously stored voiced sentences. To determine the current application QoS requirements, the triple (Variable, Reliability, Event) is sent to the system decision module, which will be described next.

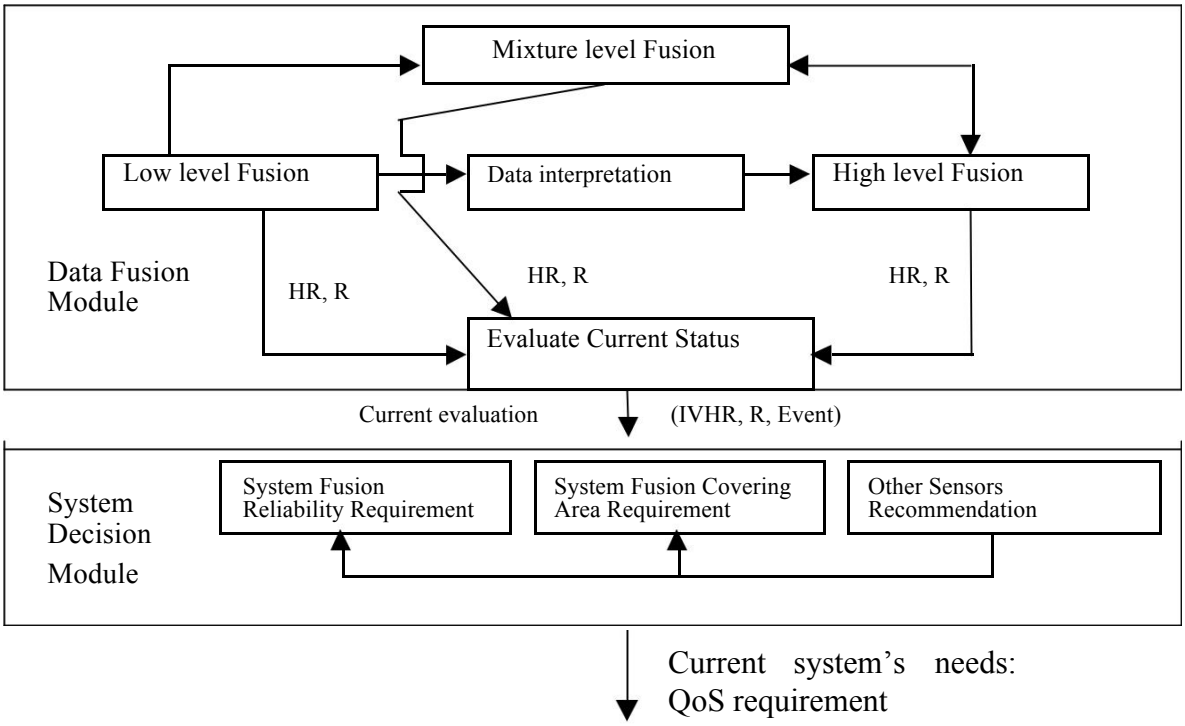


Figure 27: System diagram showing the data fusion module and the decision module.

4.6. System Decision Module

The System Decision module receives from the Evaluate Current Status module a triple: the Interpreted Variable Heart Rate (IVHR), its reliability R and the result of the evaluation (Event). The System Decision module includes the System Fusion Reliability Requirement (SFRR), the System Decision Covering Area Requirement (SDCAR) and the Other Sensors Recommendation (OSR) sub modules (figure 27). These sub modules determine the new reliability, the new covering area and the new other sensors recommendation necessities for each variable. Based on this information, the decision module will send to middleware the current system's needs (QoS requirement).

4.6.1. System Fusion Reliability Requirement

The System Fusion Reliability Requirement (SFRR) sub module is the part of the System Decision module that is responsible for evaluating the system's necessity in terms of types of sensors and their reliabilities. Based on the triple (IVHR, R, Event), this system can evaluate the new HR reliability needed by the application. We will use

discrete values to represent the range of reliability values as well as the event classification.

Events:

1. **High Risk (HR)**
2. **Moderate Risk (MR)**
3. **Low Risk (LR)**
4. **Negligible Risk (NR)**

Reliabilities States:

1. **value = 1.00**
2. **value 0.75**
3. **value 0.50**
4. **value 0.25**

Based on these evaluations we can have the following state machine to manage the variables evaluation and the associated risk with the variables reliability requirements (Figure 28). Each variable needs to have its own state machine.

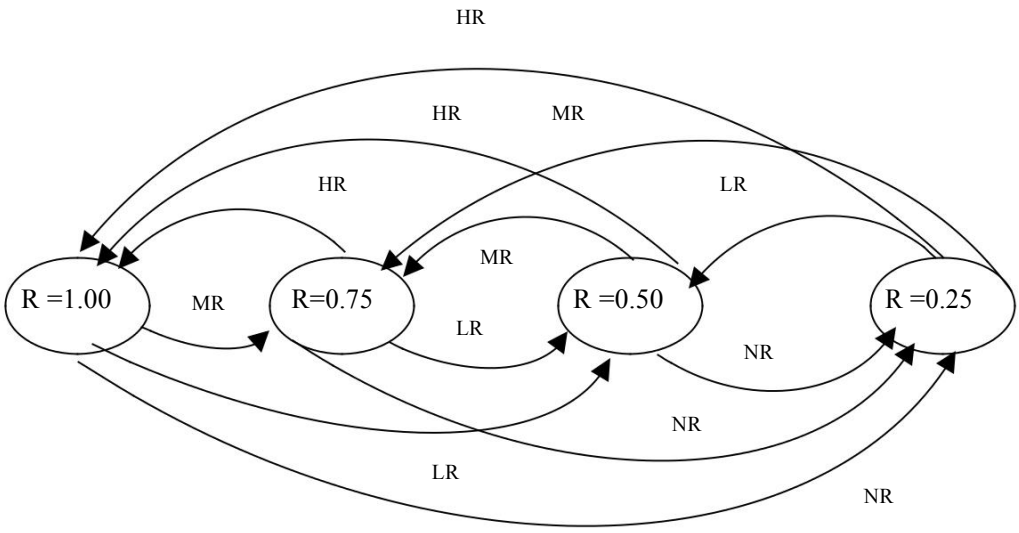


Figure 28: SFRR State Machine.

If the current reliability state is 0.25, and an event was detected and classified as a low risk event, the current reliability state would change to the reliability state equal to 0.50. For example, imagine that all the data that came from the available sensors are in the normal range and that the user is healthy. Based on this evaluation, the current reliability state would start at the 0.25 reliability state. If no event is detected, it remains in the same state. However, if a high risk arrhythmia is detected (high risk event), the

new current reliability state would be 1.00. This means that the decision system will require a new set of sensors to achieve the new variable reliability requirement (reliability equal to 1.00).

4.6.2 System Fusion Covering Area Requirement

The System Fusion Covering Area Requirement (SFCAR) sub module is called when an outlier is identified. It decides the new covering area requirements. It does not apply to the heart rate from the ECG, but can be applied to the other sensors.

We will use discrete values to represent the range of covering area values as well as the presence of an outlier event evaluation.

Event of the outlier:

1. **High Risk (HR)**
2. **Moderate Risk (MR)**
3. **Low Risk (LR)**
4. **Negligible Risk (NR)**

Covering Area States (CA):

1. **value = 1.00**
2. **value 0.75**
3. **value 0.50**
4. **value 0.25**

Based on these evaluations we can have the following state machine to manage the variables covering area and the associated risk with the outlier (figure 29). Again, each variable needs to have its own state machine.

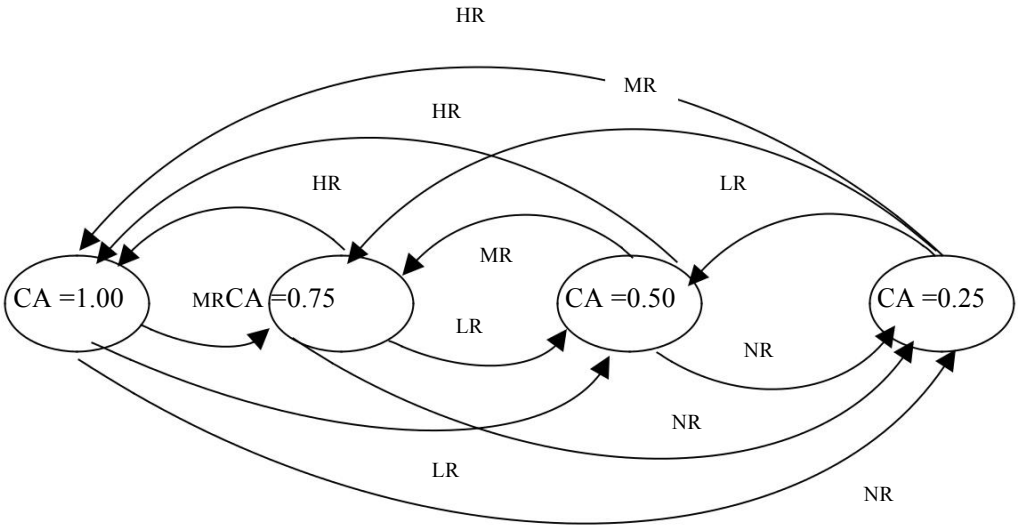


Figure 29: SFCAR State Machine.

If the current covering area state is 0.25 and an outlier's event was detected and classified as a low risk event, the current reliability state would change to the covering area state equal to 0.50. For example, imagine that all the data that come from the available sensors are in the normal range and that the user is healthy. Based on this evaluation, the current covering area state would start at the 0.25 covering area state. If no outlier is detected, it remains in the same state. But, if an outlier is detected and classified as a high risk event (e.g., a very low blood flow in the right arm), the new current covering area state would be 1.00. This means that the decision system will require a new set of sensors to achieve the new variable covering area requirement (covering area equal to 1.00).

4.6.3 Other Sensors Recommendation

The other sensors recommendation sub module is very important. It determines the influence of one variable over the others. For example, imagine that the current blood pressure value has just dropped down and the current heart rate value is inside the normal range. Based on current (HR, R) values, the HR Interpreter would decide to maintain the current HR reliability or even decrease it. However, based on the current (BP, V) values, the BP Interpreter would conclude that the blood pressure is very low, and it would increase the BP reliability and recommend an increment in the HR and O reliabilities. As a consequence, a conflict is established. Based on the principle that recommendations of increase reliability predominate over decrease recommendations, the System Fusion Reliability Requirement would decide to increase the HR reliability. The same approach should be applied to the System Covering Area Requirement. The System's Sensors Reliability & Covering Area requirement is achieved through the concatenation of the system's reliability needs with the system's covering area needs.

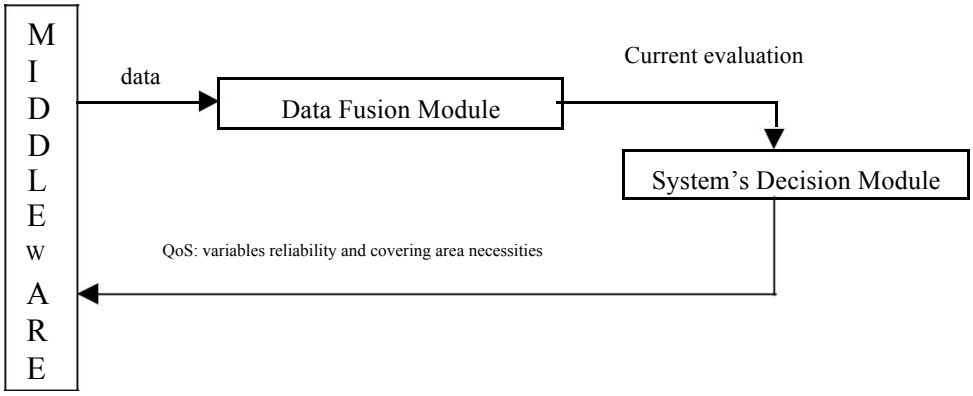


Figure 30: Data and variable management.

Figure 30 summarizes the basic principle of the system. The data fusion module receives data and evaluates the current data/variable value. Its output (current evaluation) goes to the System Decision module where, based on the current evaluation, the current QoS requirement, such as variable's covering area (CA) and reliability (R) values will be determined. This new QoS information will be sent to the middleware. The middleware

will appropriately reconfigure the network to meet the new requirements, and the new set of sensor's data will be sent to the fusion system for analysis.

4.7. The ECG Waves Recognizing System

We have developed an automatic ECG recognizer and analyzer system based on a sequence of automatons. First the signal is segmented and each wave and ECG segment are identified and classified by a five level logarithm approximation automaton. Each end state of this automaton generates a word that represents the cardiac cycle. The set of words constitute the language of the ECG. This language is interpreted by another automaton that is responsible for the identification of ECG abnormalities both in a static condition like a conventional ECG, or in continuous monitoring. The first automaton works on the ECG samples, whereas the second one works on the output of the first automaton. We have verified the algorithm by a formal verification algorithm using the Model Checking Algorithm (Verus formalism).

The developed system is able to recognize and analyze ECG signals in static and continuous (dynamic) conditions. It is based on a sequence of specialized automatons to recognize each ECG wave and segment isolated or in sequence in normal and abnormal conditions. We showed the systems' efficacy using the MIT-BIH ECG database. The automaton developed was able to recognize 96% of the "P" waves, 100% of the "QRS" waves and 98% of the "T" waves. It was also able to recognize 96% of all ECG intervals and segments (table 2).

Table 2: Comparison of the proposed algorithm and available results in the literature.

Wave Algorithm	P	QRS	T	Segment or Interval
Ours	100	100	98	96
Literature	< 96	100	100	?

? = not available data.

The fact that it is a real time algorithm and was implemented in a reconfigurable hardware permits its use in critical situations as a conventional ECG analyzer or in continuous monitoring as Holter equipment or a Web based system. Thus, the reconfigurable characteristic provides the ability to change the hardware configuration by choosing the best and most suitable algorithm to the patient in a remote manner.

The ability to recognize abnormal patterns compared to normal ECG patterns or patient baseline ECG provides the ability to store the patient baseline ECG and the different patterns recognized in the continuous monitoring. This is a very efficient compression method since it does not have the problem of signal reconstitution and stores

only what is really very important. This saves power and is very important in mobile applications of biological signals monitoring.

The proposed methodology for formal verification of the electrocardiographic waves and wave sequences recognizing system by the Verus model checker has shown good results in identifying some faults and giving more reliability to the system. The application of the methodology has permitted the system to recognize other waves in the electrocardiographic module, re-designing the actual wave recognizing automata and designing the wave sequence recognizing automata, leading to a real contribution in the biomedical system implementation.

4.8. Prototype

We have developed our first prototype based on hardware from the market. Our platform is based on the PROCOMP device (Thought Technology Ltd., Montreal, Quebec, Canada) and the PDA Zaurus (Sharp). The software was developed in C language to guarantee performance and low power. The PDA is connected through a wireless connection (IEEE 802.11b) to a base station.

4.8.1. Hardware

We have built a prototype composed by an acquisition board sold by PROCOMP (figure 31) and the PDA Zaurus 5500 (Sharp) (figure 32). The PROCOMP device is composed by multiple sensors (EEC, surface EMG, ECG, Temperature, Skin conductance, and Blood pulse) connected to a wearable board composed of an AD converter, amplifiers and filters. The Procomp device is an 8 channel, multi-modality encoder that has all the power and flexibility for real -time, computerized biofeedback and data acquisition in a clinical setting. The first two sensor channels provide ultimate signal fidelity (2048 samples per second) for viewing RAW EEG, EMG and EKG signals. The remaining six channels (256 samples/sec) can be used with any combination of sensors, including EEG, EKG, RMS EMG, skin conductance, heart rate, blood volume pulse, respiration, goniometry, force, and voltage input. ProComp offers internal, user-activated calibration to ensure that you can always obtain the highest quality signal, without the costly downtime associated with factory re-calibration. It is powered by four alkaline "AA" batteries**.** The signals are sent to a serial port in the PDA through a optical fiber cable.

Figure 31: Acquisition board

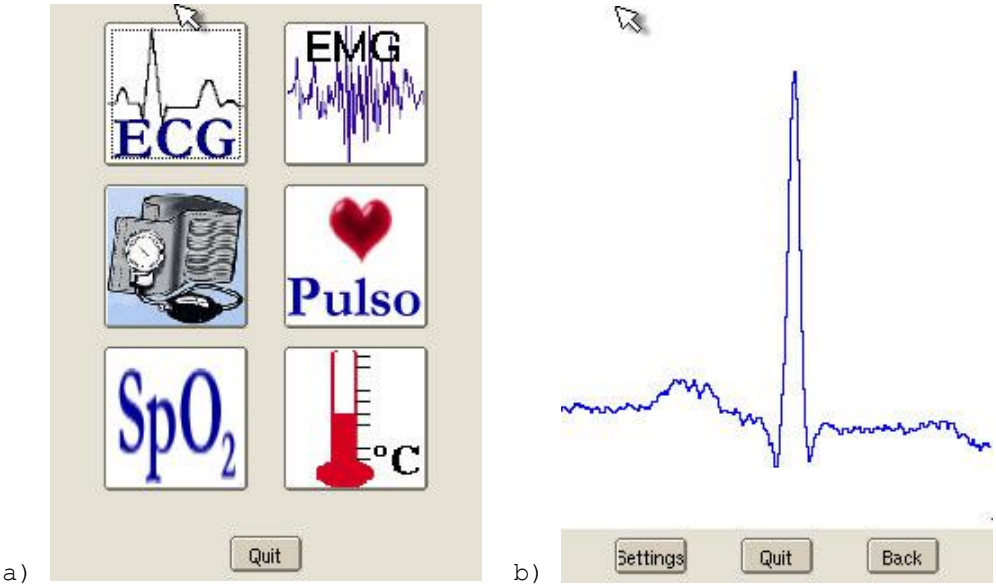
The PDA Zaurus has the following characteristics: StrongARM(1) SA-1110, 206MHz[1] , Linux[2] based embedded OS (Embedix [3]) Qtopia, Personal Java[4] , Reflective TFT LCD with Front Light (touch sensitive panel supported), 3.5" with 240x320 pixel, 65,536 colors, 64MB SDRAM 16MB FLASH ROM, Touch Panel, QWERTY keyboard with a slide cover, 1 compact Flash Card[5] slot, 1 SD card slot (no copyright protection feature), Serial/USB (via docking station port, IR port, Stereo headphone jack included, mono-audio input (via audio jack), buzzer / alarm, Calendar, Address Book, To-Do, and Memo, POP3, SMTP, IMAP4, Equiv. HTML 4.0, and is powered by a replaceable lithium-ion battery.

Figure 32: PDA Zaurus

4.8.2. Software

4.8.2.1. The PHRM software infrastructure: developed in "C" based on thread technology to represent the "parallel" processing, analysis, fusion and decision of the available data and variables.

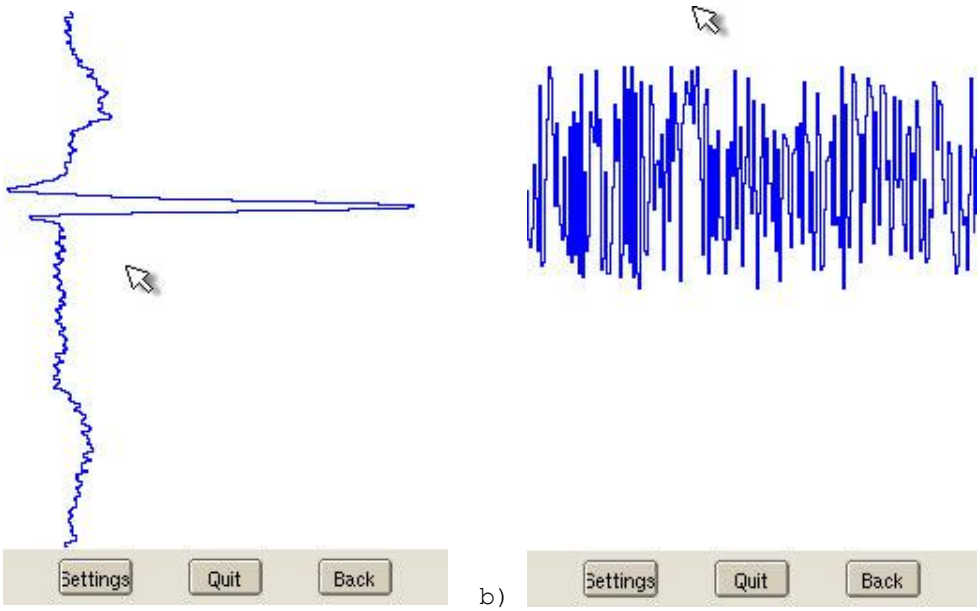
4.8.2.2. Software for signals visualization: The signals can be visualized on the PDA screen using a "C' language software based on QTopia interface. The figures below show some examples of the signals and the interface of the system. The user can use the keyboard or the touch screen to start the system and visualize the signals together or separately.



a) b)

Figure 33: The main window of the monitoring system (a) and the ECG screen (b)

Figure 33a shows the interface of the monitoring system. The user can choose to see the ECG (electrocardiogram), EMG (Electromyogram), Blood pressure, pulse frequency, SPO2 (oxygen saturation), and temperature. The user can also choose to see the signal in landscape and portrait. The settings bottom allows the user to choose some characteristics of the image, such as position and checkered background to represent the millivolts in the vertical and milliseconds in the horizontal perspective applied to the ECG analysis. Figures 33b and 34a show examples of the electrocardiogram in portrait and landscape views.



a) b)

Figure 34: The landscape view of the ECG wave (a) and the EMG screen (b).

Figure 34b shows an example of the electromyogram signal. Figures 35[a] shows the pulse oxymeter (blood oxygen saturation) view and figure 35b the blood pulse frequency screen with theirs respective data tables. Figure 36[a] shows the Blood Pressure (Systolic and diastolic measurements) view and figure 36b shows the body's temperature view with their respective data tables.

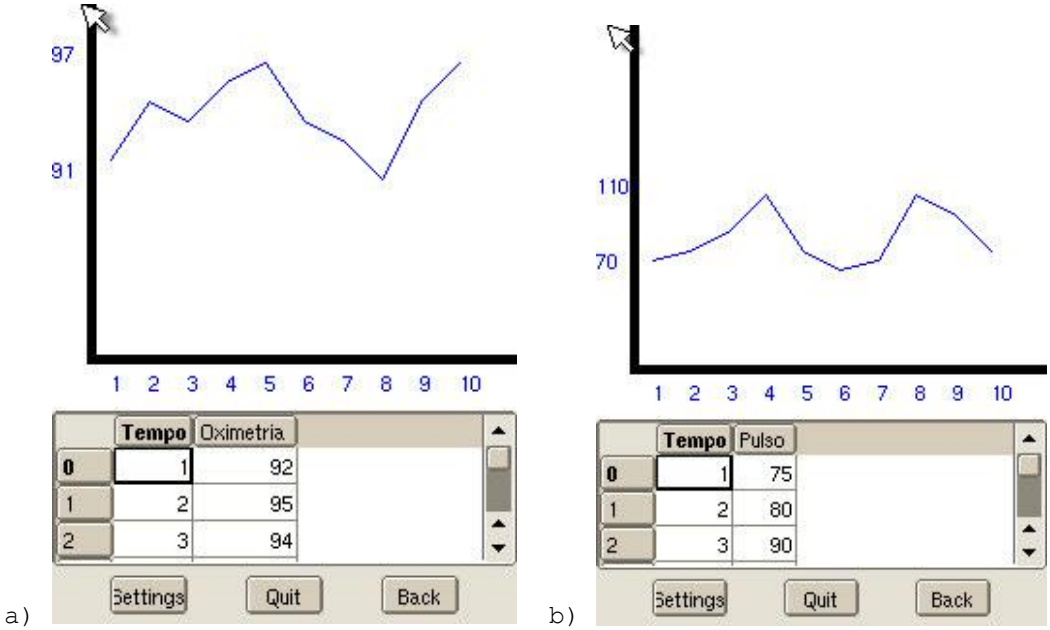


Figure 35: The Pulse oxymeter (blood oxygen saturation) view (a) and the blood pulse screen (b).

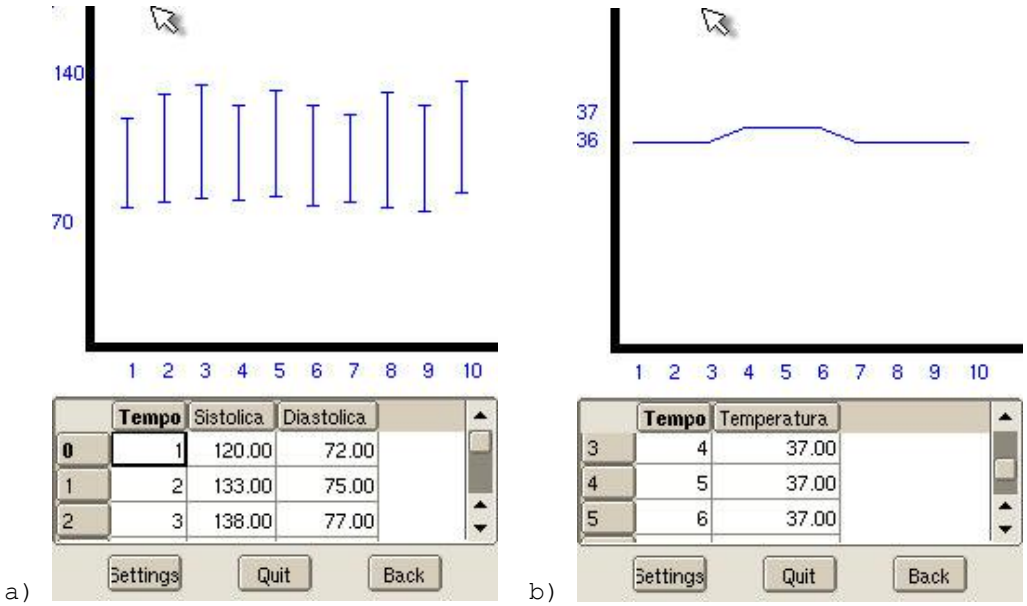


Figure 36: The Blood Pressure (systolic and diastolic measurements) view (a) and the body's temperature (b).

4.9. Experimental Results

1) Evaluation of power consumption of the System's components: measurement, processing and transmission.

Objective: to obtain real measurements about the power consumption of some parts of the system.

Methodology: we have used a prototype of a wearable computer used to monitor the ECG and pulse oxymeter signals. We could evaluate separated the different parts of the system to calculate the power consumption. These results can be used as a reference to the same hardware used, because these measurements can very from different hardware.

Results:

Table 3: Sample, and bandwidth consumption of the ECG, Pulse Oximetry and Non Invasive blood pressure signals.

	sampling	*bits*	*bandwidth*
EKG			
12	1000	16	192000
3	1000	16	48000
3	250	16	6000
1	250	16	2000
Oximetry	0.5	8	4
NIBP	0.0033	8	0.03

Table 3 shows the bandwidth necessary to run a 16 bits ECG and 8 bits Pulse oxymeter and Non invasive blood pressure. The ECG is a continuous signal, and the bandwidth consumption is greater than the other discrete signals.

Table 4: Power consumption of a multi parametric unit.

Power Consumption of Multiparametric Unit								
Component	+5V		-5V		+12V		-12V	
	Max	Avg	Max	Avg	Max	Avg	Max	Avg
Acquisition	80	76			46	46	5	5
Oximetry	187	158	4	3				
Blood Pressure	400	50						
Processing+Comm	600	360						
TOTAL	1307	644	4	3	46	46	5	5

Table 4 describes the power consumption of the acquisition, pulse oxymeter, blood pressure and processing plus communication components. The results showed that computation and communication are the most power consumption components.

Conclusion: communication is the main source of power consumption in a wearable device used for monitoring biological signals.

2) Experimental design: Based on specialist knowledge we conducted an experiment that identifies the use of the Electrocardiogram monitoring in an Intensive Care Unit (ICU). The system has the capability to monitor the electrocardiogram using one, two or three leads.

Table 5 represents this knowledge: The level of monitoring (number of leads used) is related to the patients risk of cardiovascular events. The higher the risk, the higher the necessity of improve the monitoring system. The contrary is also true, the low risk patient most of the time can be monitored by one lead ECG.

Table 5: Probability of patient's monitoring states according to his risk**:**

QoS of Monitoring / Patient State	1 channel	3 channel	12 channel
Normal patient	**97%**	**2, 9%**	**0, 1%**
Patient with low risk disease	**95%**	**4, 75%**	**0, 25%**
Patient with medium risk disease	**40%**	**59, 5%**	**0, 5%**
Patient with high risk disease	**30%**	**65%**	**5%**

Methodology: we employed the classification described bellow to choose the level of monitoring related to the patients risk.

Patient with normal EKG

1 channel / 300Hz, delayed by 30 sec

Patient with low risk abnormalities in EKG
3 channels / 300Hz, delayed by 5 sec

Patient with high risk abnormalities in EKG 12
channels / 1000Hz, delayed by 1 sec

Results: Figure 37 shows the power state machine of the experiment. It describes the power consumption in the different states (1 to 12 leads) and the transitions (patients risk). If we monitor the patient all the time with one lead ECG the power consumption is lower than monitoring all the time using an ECG 12 leads monitor. It is clear that based on clinical variables such as the patients risk we can save power and adapt the system to the patients benefit. This is a great improvement for a wearable monitoring device where power is limited.

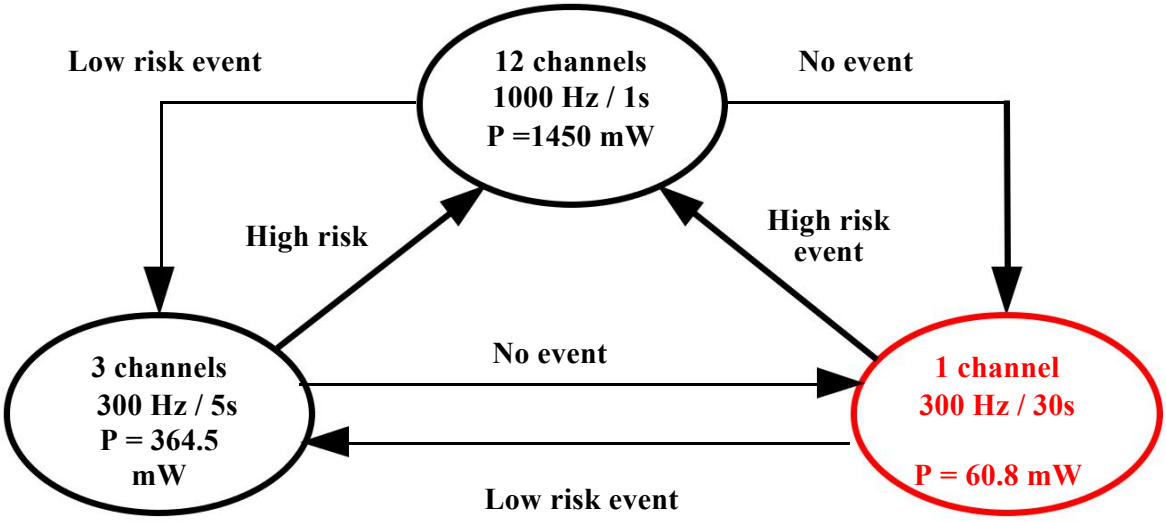


Figure 37: EPSM Wearable ECG Monitor

Conclusions: it is possible to use clinical evaluation to determine variable that can help to adapt the system to increase its lifetime.

2) System's lifetime related to the visualization of each signal in the Zaurus SL-5500

Objective: To measure the power consumption of the different biomedical signals on the screen of the PDA Zaurus SL-5500.

Procedure: The power consumption of the Zaurus SL-5500 using the ROM 3.10 version cannot be made directly. One bug in the power management system makes impossible the use of a software that directly measures the power consumption. To measure the power consumption we had to develop a software that writes the time each 30 seconds. So the last written time is the last time the system was functioning. The procedure is described below:

1. Power the battery.
2. Turn off the power supply and start the signal window that we want to measure the power consumption.
3. Run the signal monitor until the Zaurus' battery goes down.
4. To avoid interruptions in the monitoring program it was necessary to turn off the automatic stat of the PDA. It was achieved using the software qpe-suspendapplet. The use of a screen saver was also avoided.
5. As the control is made every 30 seconds, this is the precision of the measurement.

Results: The results described in table 6 shows that the backlight is the most power consuming component. It is necessary to turn it off to increase the system lifetime. On the other hand, the signal visualization is compromised whether we turn it off or not. Continuous variables such as ECG, EMG and EEG are related to lower lifetime. Discrete variables such as oxygen saturation and blood pressure are related to a longer lifetime. As a consequence, the system's lifetime is directly related to the sample frequency of the monitored signal.

Table 6: Battery lifetime related to different monitoring sets and backlight.

Signal	Backlight	FPS	Time (minutes)
None	100%	0	111
	Turn off	0	596
ECG	100%	30	73
	100%	30	86
	100 %	Maxi	71
	Turn off	mum	≈300
EMG	100%	30	81
	100%	30	83
Blood	100%	1	101
Pressure	Turn off	1	590

Conclusion: The backlight is the most power consuming component. Excluding the backlight component, the system's lifetime is directly related to the signal's sample frequency.

4.10. Discussion and Conclusions

4.10.1. Discuss the prototype: The PHRM is new system that takes into consideration that it is very difficult to analyze one body variable independently. In the case of the heart rate, without knowing the user age, underlying diseases, level of body activity, whether he is sleeping or awake, it is very difficult to conclude whether the measured heart rate is normal or abnormal. Nowadays, all of the available systems do not consider this wild view of the physiological monitoring and interpretation. Another aspect that makes the developed system new is the fact that it takes into account aspects such as fault tolerance

and ubiquitous computing determined by not only the hardware, but also determined by the application's necessity (Application's quality of Service). In this aspect, the presence of a middleware can help to manage the hardware point of view together with the application point of view, optimizing this relationship and the system life time.

The nearest system presented in the current literature is the Total Heart project developed by the Macmaster University. This system is a mobile device developed to achieve low power consumption of its components. It is composed by a sensor board for the ECG attached to a processing and analysis system based on neural networks to detect events locally in the mobile device. The PHRM is a more complete system because it can changes the number of leads according to the user necessity. Furthermore, the PHRM is able to correlate the heart rate with other signals and users characteristics, such as the level of body activity, blood pressure, body position and underlying medical conditions. Besides this, the PHRM considers the power management from the systems point of view, changing the components status (on or off) according to the quality of service of the application (heart rate monitoring).

There are other systems that only record the ECG (Holter machines) using different media such as tapes, flash memory or smart cards. None of theses systems have local processing and analysis. Although all these systems are mobile, all the data management is made offline. The same type of device is described for blood pressure, blood glucose measurement and oxygen saturation evaluations. Some systems used for biofeedback are composed of multiple sensors, but the data management is also made offline.

Based on these aspects, The PHRM is the first heart rate monitoring system that considers multiple correlations with the user's health variables to help the user in the monitoring of his health. To achieve this goal the general data fusion architecture applied showed that it is a reasonable approach to provide dynamic management of data and variables from different sources. Besides this, the developed system is fault tolerant to failures and compatible with ubiquitous availability.

It is our intention to substitute in the next generation of the prototype micro sensors by nanosensors in the direction of a body-worn sensor networks.

4.10.2. Contributions

A body-worn heart rate monitoring system;
The ECG Waves Recognizing System;
Algorithm for surface electromyogram interpretation;
Algorithm for body's activity evaluation;
Algorithm for correlation between signals and clinical medical evaluation (symptoms and signals);
Algorithm for sleep/awake evaluation;
Power state machine based on clinical variable and level of monitoring (number of leads of the ECG);

Future work: We are now starting a clinical trial described below:

4.10.3. Long Term Monitoring of Physiological Signals Clinical Trial

Institutions:

University of Brasilia Hospital, School of Medicine, University of Brasilia, DF-Brazil

Center For Future Health, University of Rochester Medical Center, Rochester, NY-U.S.A.

Motivation:

1) Chronic diseases such as cardiovascular diseases, cancer and degenerative diseases are the most common cause of mortality and morbidity in United States and Brazil. Morbidity prevention and decrease mortality of chronic diseases can have a great impact in the health care system in both countries.
2) Lack of long term physiological data studies
3) Lack of good correlation between data gathered in laboratory studies and data gathered at home.
4) Available infra structure: Smart Home and Mobile Monitor.

Hypothesis:

Long term physiological monitoring during daily activities can predict and provide early detection of morbidity conditions.

Main objectives:

Correlate physiological data with daily activities

Secondary objectives:

1) Development of machine learning tools to analyze physiological data.
2) Development of a software infrastructure to provide long term physiological monitoring compatible with all the constraints inherent to mobile and wireless monitoring.
3) Public database of physiological data

Human–machine interaction studies

Chapter 5

Data Fusion implementation in distributed sensor networks

The benefits provided by distributed data fusion of sensor data are widely accepted: enhanced targeting and area coverage accuracy; increased situational awareness; greater confidence in tracking and association; reduced false alarms and improved application space management. Some authors have been working in this field. Durrant-Whyte and Stevens [86] have described three constraints that characterize distributed data fusion:

1. There is no single central fusion center; no one node should be central to the successful operation of the network.
2. There is no common communication facility; nodes cannot broadcast results and communication must be kept on a strictly node-to-node basis.
3. Sensor nodes do not have any global knowledge of sensor network topology; nodes should only know about connections in their own neighborhood.

Based on these premises, distributed data fusion depends on three aspects: sensor role and management [92][83] , wireless communication and data fusion path or data dissemination.

1) Sensor role: and management

Different aspects should be considered at this topic:

- Sensing the environment
- Computation capability
- Relay: receive/transmission capability without processing
- Data fusion capability:
    - Different levels of data fusion, data analysis and variable interpretation
- Decision:
    - Change sensor mode:
    - Change data fusion model
    - Control actuators
- Mobility

2) Communication: all the aspects related to wireless communication

3) Information dissemination algorithms and data path (optimal node data fusion):

Wendi et all have shown that data aggregation increases the system's lifetime and decrease power consumption in power aware sensor networks [102]. Since this initial paper many different aspects related to distributed data fusion such as information dissemination, data fusion techniques, optimal data fusion node, data fusion and system's

lifetime, accuracy, coverage area, have been addressed by different authors. Some of them have considered the node mobility and code mobility [87][91], where mobile agent adopts a new computing model in which data stay at the local site, while the execution code is moved to the data sites. The problem of how to fuse data from multiple sensors in order to make a more accurate estimation of the environment has been studied by other authors [88][89][90][96][97]. Most of the time sensors can have different levels of energy, in these cases the load balance should be considered [93]. Many authors have worked on data dissemination and association [94][63][18][101], and data fusion middleware [95]. A few papers have addressed the optimal data fusion node problem. [99][100]. Our work addressed the optimal data fusion path problem, which is related to the optimal data fusion node problem.

Mohin et all [99] worked on ."the problem of choosing fusion nodes in a heterogenous wireless sensor network, operating in a terrain with blockages, where some sort of data fusion or data management can be performed to optimize some defined network performance metrics". Their main contribution is related to the definition of centroide. Centroide is the best node to perform data fusion in the local area network. The problerm is that to determine the centróide node the algorithjm used does not take into account that the data source can be in different parts of the environment (out from the local network). It also does not take into account the sensing, communication (receive and transmission), and data fusion computation cost that are very important aspects to evaluate the system´s lifetime. Furthermore, their simulation includes networks with 20 nodes. We have showed in our research that the different algorithms have different performance according to different network densities. In this aspect, our approach takes into account all these variables and the simulations varied from low network densities (network with 50 nodes) to high density networks with 1000 nodes.

Konstantinos et all [100] described approaches to solve the maximum lifetime data gathering problem in sensor networks, with and without data aggregation. Although their solution based on the maximum flow algorithm can be used as a reference for the maximum lifetime of a sensor networks application, they didn't consider the sensors individually. As a consequence, They didn't consider the tradeoff between the total cost (communication, sensing and data fusion computation) and system's lifetime. It does not solve the problem of finding data fusion path for data from different source sensing units.

We consider that the data fusion capability such as different levels of data fusion, data analysis and variable interpretation, as well as the decision (change sensor mode, change data fusion model, and actuators control) can also be part of the sensors functionalities (smart sensors). Furthermore, each tool, parametric and non parametric statistical approach [98] as well as soft computing approaches (neural networks, fuzzy logic, genetic algorithms, etc), performed to achieved pre-processing or data fusion has different computation cost. In addition, the data path can be found based on the data source nodes, possible data fusion nodes, and destination node for the fused data or information

Problem to solve: what is the best path to fuse data from data source (sensing node) to send the fused data or information to some destination node, i.e., how to find the optimal (s) data fusion node (s)?

Body-worn sensor networks distributed data fusion problem: In the body-worn sensor networks environment the sensors are distributed along the body on the surface (skin), inside organs such as lungs and heart, or inside the vascular system or gastro intestinal system involved by the blood and food bolus respectively. Considering that a network is composed of smart sensors with sensing, relay and data fusion capability, how can we determine the best path to achieve the data fusion goal, i.e. fused data should arrive in a pre-defined base station, considering sensing cost, data pre-processing and data fusion computation and transmission cost?

The approach to this problem should consider:

1. Tradeoff: total cost (sensing, computation, and communication) versus system lifetime.
2. Total cost: once the total cost of computation is the same independent of the workload balance, what determines the total cost is the communication cost plus the sensing cost plus de total computation (pre-processing and data fusion techniques). Based on this, the relaxed pairs shortest path can be employed to optimize the communications cost.
3. System´s lifetime: the DF path choice is determined by the available power in each node. In this case the system might want to use a high cost path, but that will increase the system lifetime. In this case all possible DF paths should be used.

5.1. Distributed Data Fusion Implementation

Most sensor networks applications will run in a distributed scenario. In this case, the data fusion has to be done in a distributed way. How to fuse distributed data taking into account different aspects that should be considered in this type of application, such as the available resource in terms of node's energy and sensing capability.

Data fusion is an important aspect related to the application development in a sensor networks environment. It is directly related to the application quality of service specifications and also related to different aspects of the sensor networks environment. Based on these aspects, different aspects should be considered in the data fusion implementation in sensor networks:

1) **Communication cost** (receiving and transmiting)
    f. Influence of different sizes of data. If the data is higher than the packet, we need to add the number of necessary messages to send all the data.
2) **Data fusion cost**:
3) **Data fusion precedence**:
4) **Sensing cost**
5) **Node capability**: power, computation capability and functionalities capability
    g. All the nodes have the same capabilities.
    h. Heterogeneous nodes
6) **Different data rates:** it can determine the use of shortest or longest paths.
7) **Real time versus not real time**: real time applications need a short path and smaller delays.
8) **Security**: Adds overhead in the communication cost.

9) **System lifetime**: how many paths can we use to provide a larger system's lifetime? From the lifetime point of view, the use of as many paths as possible is useful.
10) **Data throughput**: the shortest paths are related to a higher throughput than the larger paths.
11) **Transmission delay**: the transmission delay is higher in the longer paths than in the shortest paths.
12) **Time synchronization:** it is easier to obtain synchronization on the shortest paths than on the longest paths.
13) **Location integration**:
    i. Scenario 1: any source, anywhere.
    j. Scenario 2: sources to be fused are near from each other.
14) **Scalability:** We can apply the algorithm to the entire network or divide the network in smaller domains represented by different destination nodes. We can then integrate the domain's destination nodes to scale to the entire network.

We have developed some algorithms to find the best optimal node to fuse the data from different sensors considering some of the aspects described above.

5.2. Methodology

5.2.1. Objective: Given a data fusion expression, determine the best data fusion nodes (path) based on communication cost, sensing capability and system's lifetime to fuse the data from the data source nodes to the destination node.

5.2.2. Data Fusion expression definition

- All the operations are inside a parenthesis;
- All opened parenthesis have a corresponding closed parenthesis;
- The operator + means fusion of data;
- The operands represent the data sources;
- The operations inside parenthesis have precedence over non parenthesis operations.

Ex: DF expression = ((A + B) + (C + D)) = (AB + CD) = ABCD;
DF expression = ((A1 + A2 + A3 +A4 +A5) + B + C) = (A + B + C) = ABC

5.2.3. Algorithms

Brute force algorithm: Test all the ways to get the data to each possible DF node and combine the cost of each combination. This is an exponential problem (NP complete).

Optimal solution: Use the **shortest path algorithm** to get the data to each possible DF node and combine the cost of each combination. This is an approximation of the brute

force algorithm but still a very hard problem (quadratic as a function of the number of nodes in the network).

Approximate solutions: We are going to employ different approximate heuristics based algorithms to find a sub optimal solution with a lower computation cost than the optimal solution.

- **Greedy solution:**
- **Approximate Greedy solution:**
- **Optimal Node:**
- **Optimal node plus intermediate nodes (Hervaldo)**

Algorithms for the tests:

Notation: A = 1, B = 2, C = 3, D = 4 and destination node = dst.

```
//////Calculate all pairs shortest path and store in a table//////

for dst = 1:num_sensors,
   %disp(dst);
   [paths, total_cost] = find_shortest_path(dst, cost);
   cost_table(dst, :) = total_cost;
   path_table(dst, 1:size(paths,1), 1:size(paths,2)) =
paths; end;
////////////////////////////////////////////////////////////
```

1) Optimal Algorithm: evaluate all the shortest path

```
solutions for fusion_point12 = 1:num_sensors,
      cost12 = cost_table(src1, fusion_point12) + cost_table(src2,
      fusion_point12); for fusion_point34 = 1:num_sensors,
         cost34 = cost_table(src3, fusion_point34) + cost_table(src4,
         fusion_point34); for fusion_point1234 = 1:num_sensors,
cost1234 = cost_table(fusion_point12, fusion_point1234) + cost_table(fusion_point34,
fusion_point1234) + cost_table(fusion_point1234, dst);
                  cost_final = cost12 + cost34 + cost1234;
                  if (cost_final < optimal_cost),
                           optimal_fusion_point12 = fusion_point12;
                           optimal_fusion_point34 = fusion_point34;
                           optimal_fusion_point1234 = fusion_point1234;
                           optimal_cost = cost_final;
                 end;
```

```
        end;
      end;
   end;
```

2) Calculate the greed solution: evaluate the best fusion nodes by dividing the evaluation in parts (A + B) , (C + D), (AB + CD + destination).

```
/// Evaluate the fusion point for A and B//
for fusion_point12 = 1:num_sensors,
      cost12 = cost_table(src1, fusion_point12) + cost_table(src2,
      fusion_point12); if (cost12 < greed_cost12),
         greed_fusion_point12 =
         fusion_point12; greed_cost12 = cost12;
      end;
 end;
### Evaluate the fusion point for C and D####
for fusion_point34 = 1:num_sensors,
      cost34 = cost_table(src3, fusion_point34) + cost_table(src4,
      fusion_point34); if (cost34 < greed_cost34),
         greed_fusion_point34 =
         fusion_point34; greed_cost34 = cost34;
      end;
end;
 ####Evaluate the fusion point for AB and CD and destination node####
for fusion_point1234 = 1:num_sensors,
cost1234 = cost_table(greed_fusion_point12, fusion_point1234) +
cost_table(greed_fusion_point34, fusion_point1234) + cost_table(fusion_point1234, dst);
      if (cost1234 < greed_cost1234),
         greed_fusion_point1234 = fusion_point1234;
         greed_cost1234 = cost1234;
      end;
  end;

   greed_total_cost = greed_cost12 + greed_cost34 + greed_cost1234;
```

3) Calculate the modified greed solution (considering the destination node): evaluate the best fusion nodes by dividing the evaluation in parts (A + B) , (C + D), (AB + CD + destination). The difference from this algorithm to the greedy solution is that this one considers the destination node to calculate de (A + B) and (C + D) fusion nodes.

```
 /// Evaluate the fusion point for A and B//
for fusion_point12 = 1:num_sensors,
cost12 = cost_table(src1, fusion_point12) + cost_table(src2, fusion_point12) +
```

```
cost_table(fusion_point12, dst);
      if (cost12 < part_cost12),
         part_fusion_point12 = fusion_point12;
         part_costdest = cost_table(fusion_point12, dst);
         part_cost12 = cost12;
      end;
end;
part_cost12 = part_cost12 - part_costdest;
part_costdst = 0;

### Evaluate the fusion point for C and D####
for fusion_point34 = 1:num_sensors,
cost34 = cost_table(src3, fusion_point34) + cost_table(src4, fusion_point34) +
cost_table(fusion_point34, dst);
      if (cost34 < part_cost34), part_fusion_point34 =
         fusion_point34; part_costdest =
         cost_table(fusion_point34, dst); part_cost34 =
         cost34;
      end;
end;
part_cost34 = part_cost34 - part_costdest;

####Evaluate the fusion point for AB and CD and destination node####
for fusion_point1234 = 1:num_sensors,
cost1234 = cost_table(part_fusion_point12, fusion_point1234) +
cost_table(part_fusion_point34, fusion_point1234) + cost_table(fusion_point1234, dst);
      if (cost1234 < part_cost1234),
         part_fusion_point1234 = fusion_point1234;
         part_cost1234 = cost1234;
      end;
end;

part_total_cost = part_cost12 + part_cost34 + part_cost1234;
```

4) Approximate solution Hervaldo: calculate the best fusion point (ABCDdst node) for all the involved nodes (A, B, C, D, and destination). Now calculate the best fusion node to fuse A and B to ABCDdst node and C and D to the ABCDdst node.

```
####Evaluate the fusion point for A, B, C, D and destination node###
for fusion_point1234 = 1:num_sensors,
cost1234 = cost_table(src1, fusion_point1234) + cost_table(src2, fusion_point1234) +
cost_table(src3, fusion_point1234) + cost_table(src4, fusion_point1234) +
cost_table(fusion_point1234, dst);
      if (cost1234 < Aprox_cost1234),
         Aprox_fusion_point1234 = fusion_point1234;
```

```
        Aprox_cost1234 = cost1234;
    end;
end;
 ###  Evaluate the fusion point for A and B to the fusion point ABCD and
destination####
for fusion_point12 = 1:num_sensors,
cost12 = cost_table(src1, fusion_point12) + cost_table(src2, fusion_point12) +
cost_table(fusion_point12, Aprox_fusion_point1234);
    if (cost12 < Aprox_cost12),
        Aprox_fusion_point12 =
        fusion_point12; Aprox_cost12 = cost12;
    end;
end;
###  Evaluate the fusion point for C and D to the fusion point ABCD and
destination####
for fusion_point34 = 1:num_sensors,
cost34 = cost_table(src3, fusion_point34) + cost_table(src4, fusion_point34) +
cost_table(fusion_point34, Aprox_fusion_point1234);
    if (cost34 < Aprox_cost34),
        Aprox_fusion_point34 = fusion_point34;
        Aprox_cost34 = cost34;
    end;
end;
Aprox_total_cost = Aprox_cost12 + Aprox_cost34 +
cost_table(Aprox_fusion_point1234, dst);
```

Software for simulations: We have used the MATLAB and the MATLAB network toolbox.

Simulations assumptions:

1. Network of different number of nodes: networks simulated from 50 nodes to 1000 nodes.
2. Randomized nodes distribution (each trial): for each trial, the nodes distribution in the field was randomized.
3. Randomized data sources (sensors): for each trial, the nodes responsible for the data sources (sensing units) in the field were randomized.
1. Randomized destiny nodes: for each trial, the node assigned as the node destiny for the fused data (destiny node) in the field was randomized.

Figure notations:

1. data source (sensor) represented by a blue triangle
2. Destiny node: blue circle
3. Data fusion nodes A + B and C + D: small red circles
4. Data Fusion node: (A+B) + (C+D): large red circle

5. OBS: each sensor node must be a different randomized one. This means that each sensor can only sense one variable each time. There are no restrictions about the other nodes (data fusion and destiny nodes). They are defined by the algorithm employed.

Example:

The following figures 38, 39, 40 e 41 represent examples of the different solutions obtained from the employment of the different algorithms to capture the data from the data sources (sensors) and to find the best routes to perform data fusion and send the resulted data to the destination node. Note that all the solutions are based on the same set of sensors and destination node positions, but presenting different solutions to perform data fusion.

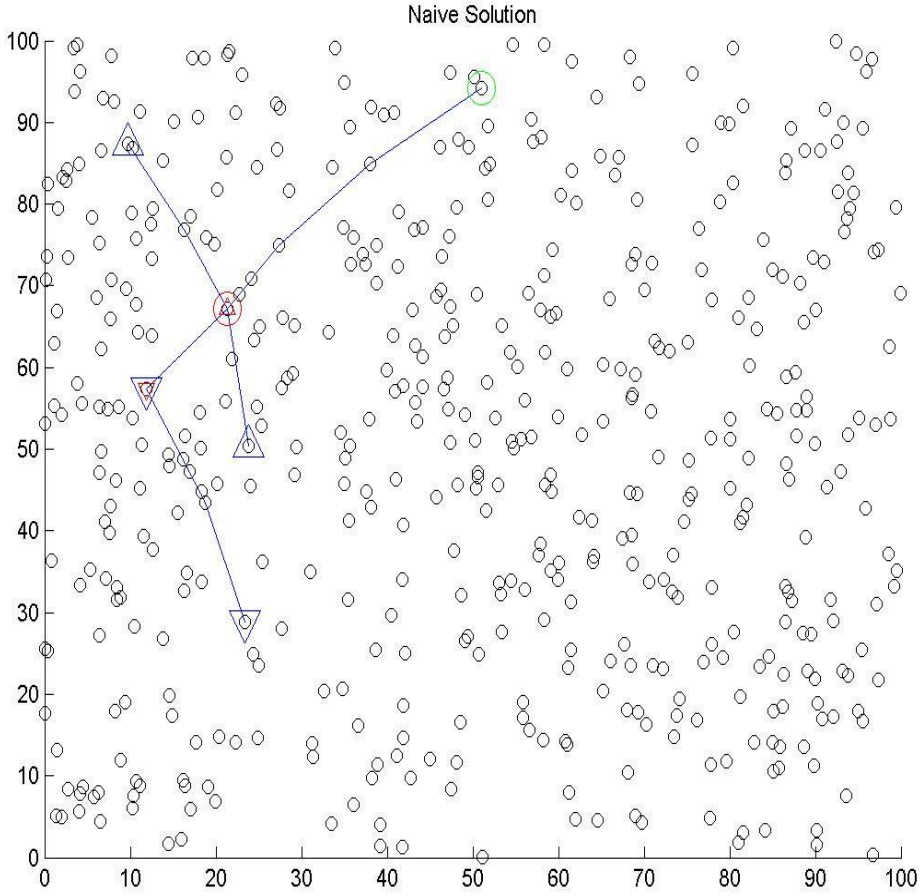


Figure 38: Shortest path algorithm solution

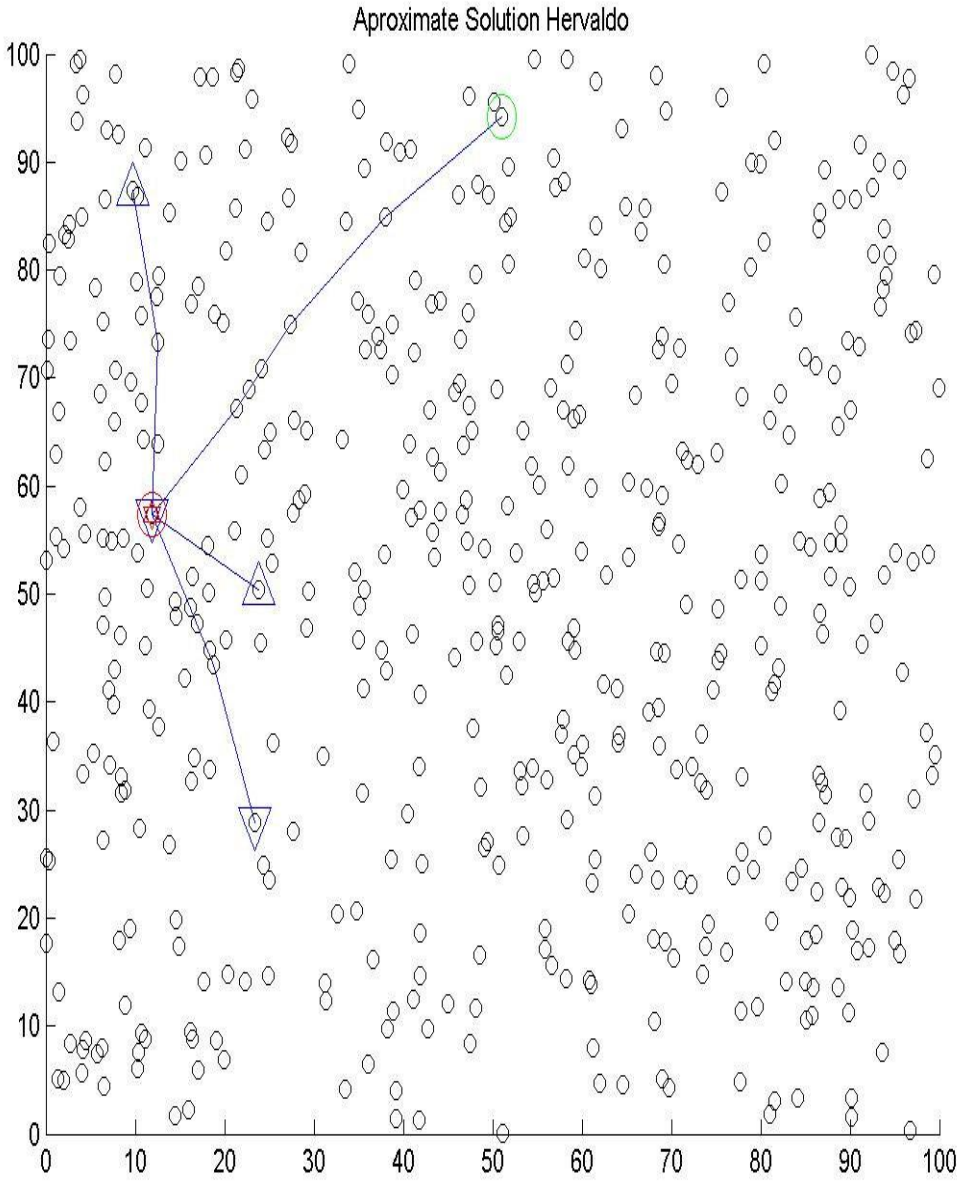


Figure 39: Approximate solution Hervaldo and Optimal node algorithms solutions (same resulted graph)

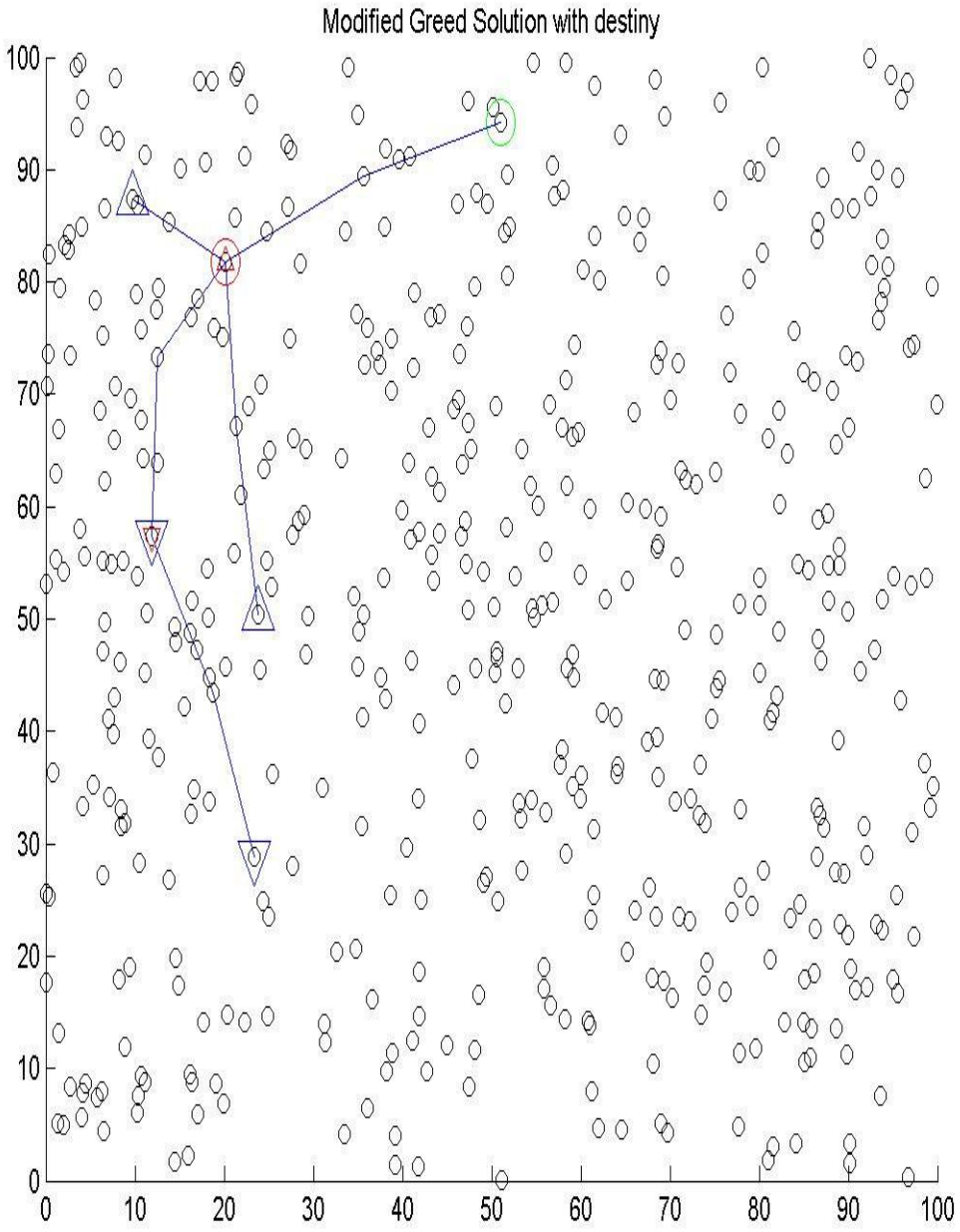


Figure 40: Approximate solution Greedy

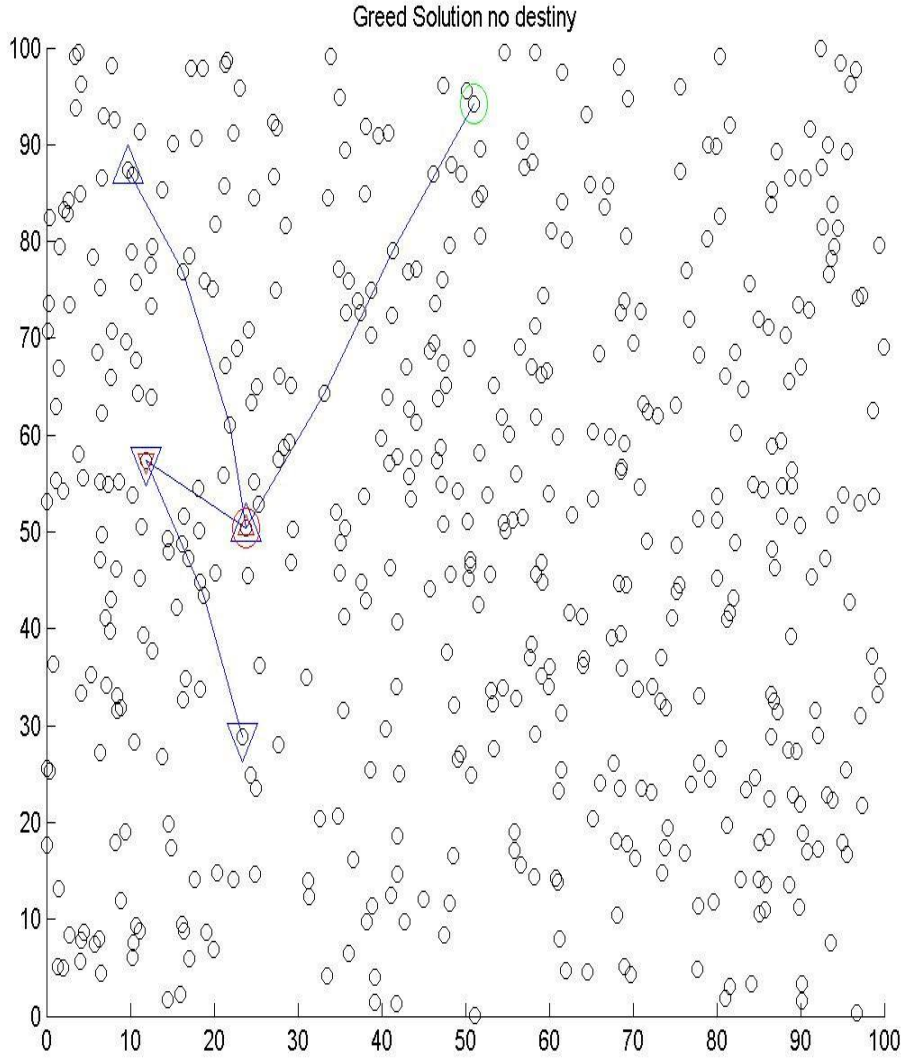


Figure 41: Approximate solution Modified Greedy

4) Results of Simulations:

1) We have compared the optimal solution against the approximate solutions to find the optimal data fusion node (s) to fuse the data from 4 different sources and to send the fused data to a destination node. We have employed the shortest path algorithm described by Dijkstra.

    a. Comparison among the different algorithms based on communication cost: optimal, approximate based on greedy solution, modified greedy, Approximate Hervaldo , and one optimal node algorithm.

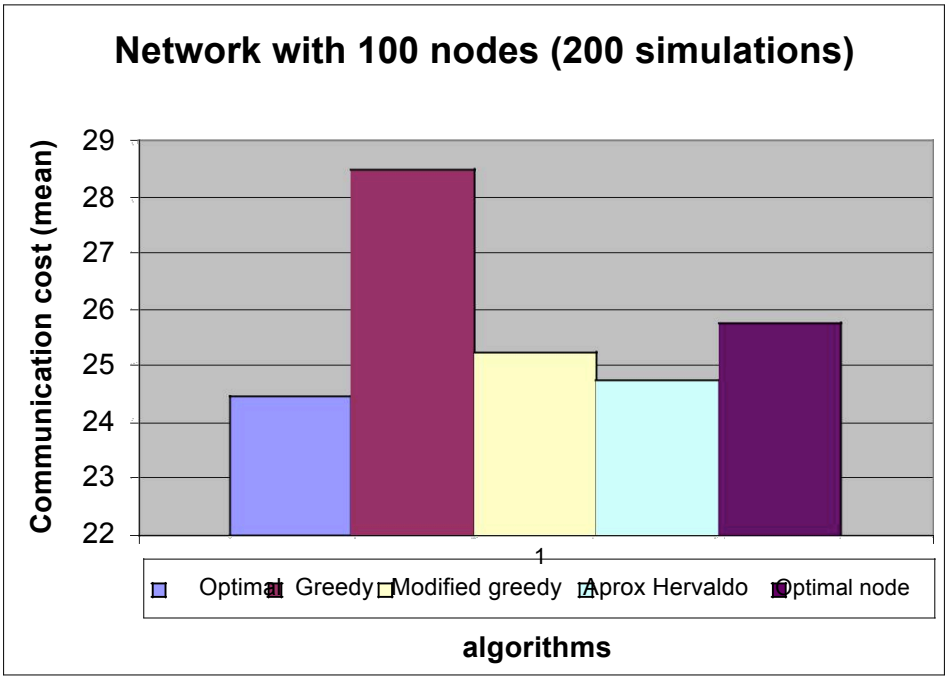


Figure 42: Results obtained from a network of 100 nodes

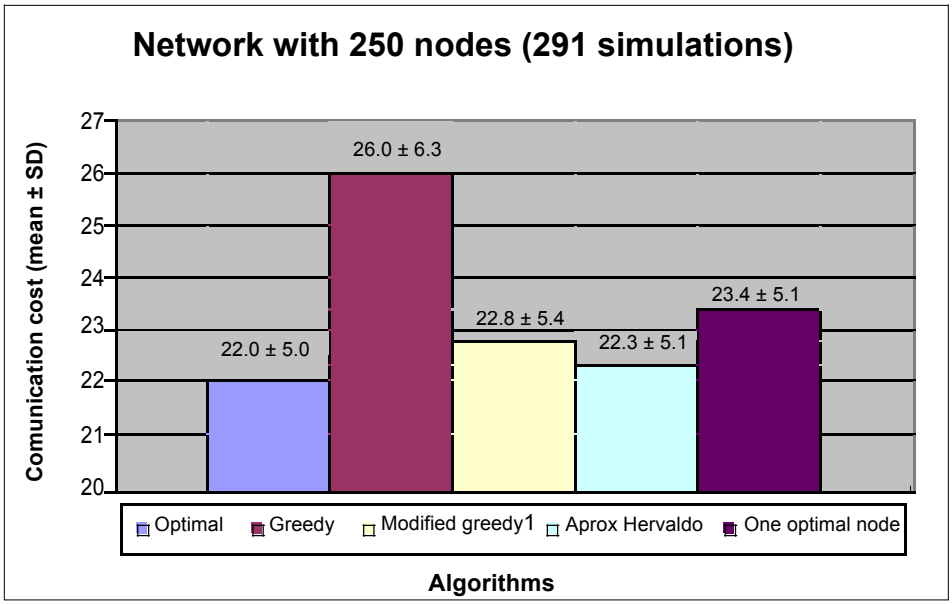


Figure 43: Results obtained from a network of 250 nodes

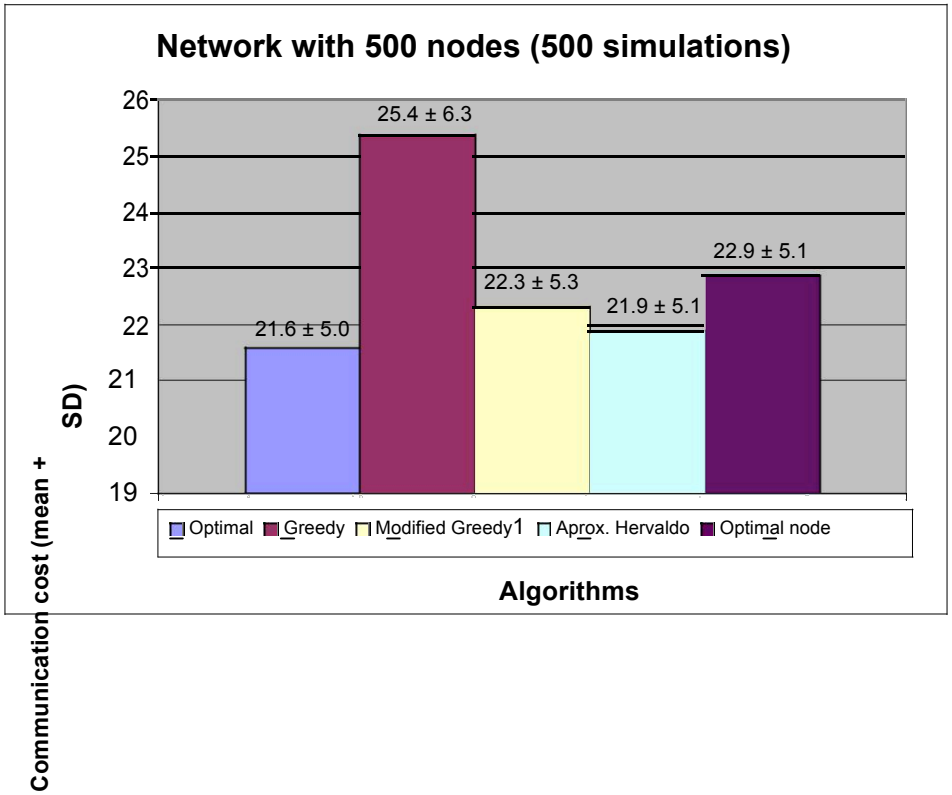


Figure 44: Results obtained from a network of 500 nodes

The results obtained from the simulations in different network densities (100 to 500 nodes) showed that the approximate solution named Hervaldo, Modified Greedy, Optimal node and Greedy are the best alternatives to the optimal solution (figures 42, 43 e 44). As a conclusion, The approximate solution Hervaldo is the best approximate solution and as a consequence, the nearest solution to the optimal solution.

b. Comparison among the different algorithms based on computation cost (running time): optimal, approximate based on greedy solution, modified greedy, approximate Hervaldo, and one optimal node.

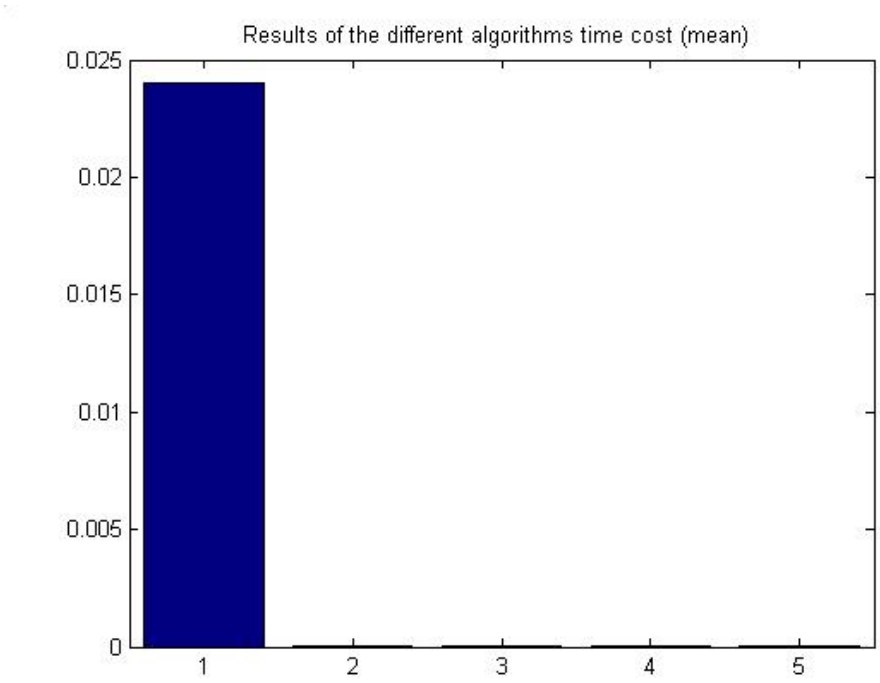


Figure 45: Computation cost of the optimal solution (1), Greedy (2), Modified Greedy (3), approximate Hervaldo (4) and Optimal Node (5).

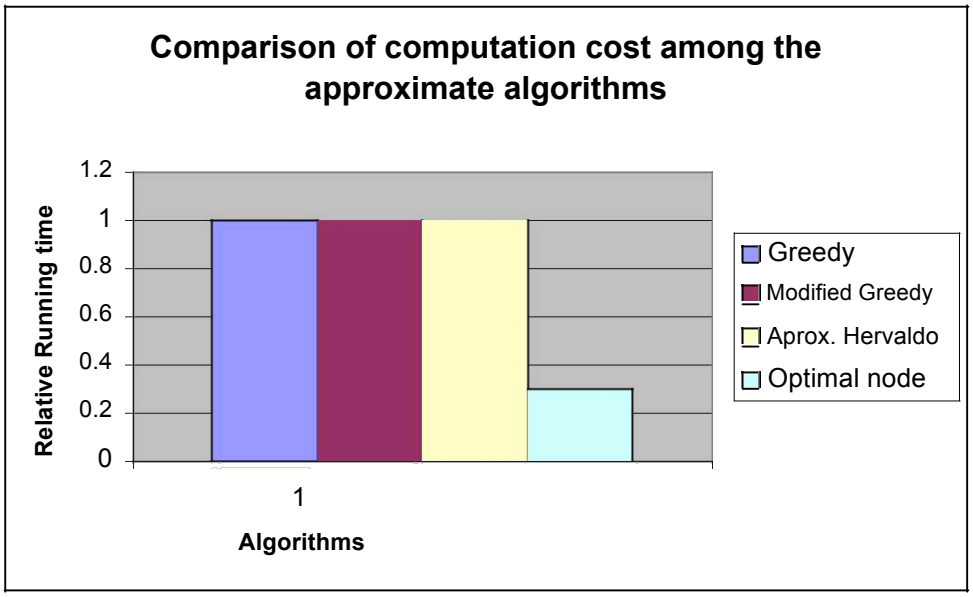


Figure 46: Computation cost of the approximate solution Greedy, Modified Greedy, approximate Hervaldo and Optimal Node.

The results obtained from the computation cost simulations showed that the approximate solutions are less expensive than the optimal solution (figure 45). The comparison of the computation cost among the aproximate solutions showed that the Greedy, modified Greedy and Hervaldo have the same computation cost. The computation cost of the optimal node is only 20% (1/5) of the other approximate solutions (figure 46). As a conclusion, The optimal node solution has the lowest computation cost. .

As a result of the simulations considering the communication cost and the computation cost, the best alternatives to the optimal solution are the approximate Hervaldo and the optimal node solution.

c. Comparison among the different approximate algorithms based on DF path lifetime: approximate based on greedy solution, modified greedy, one optimal node, and Approximate Hervaldo.

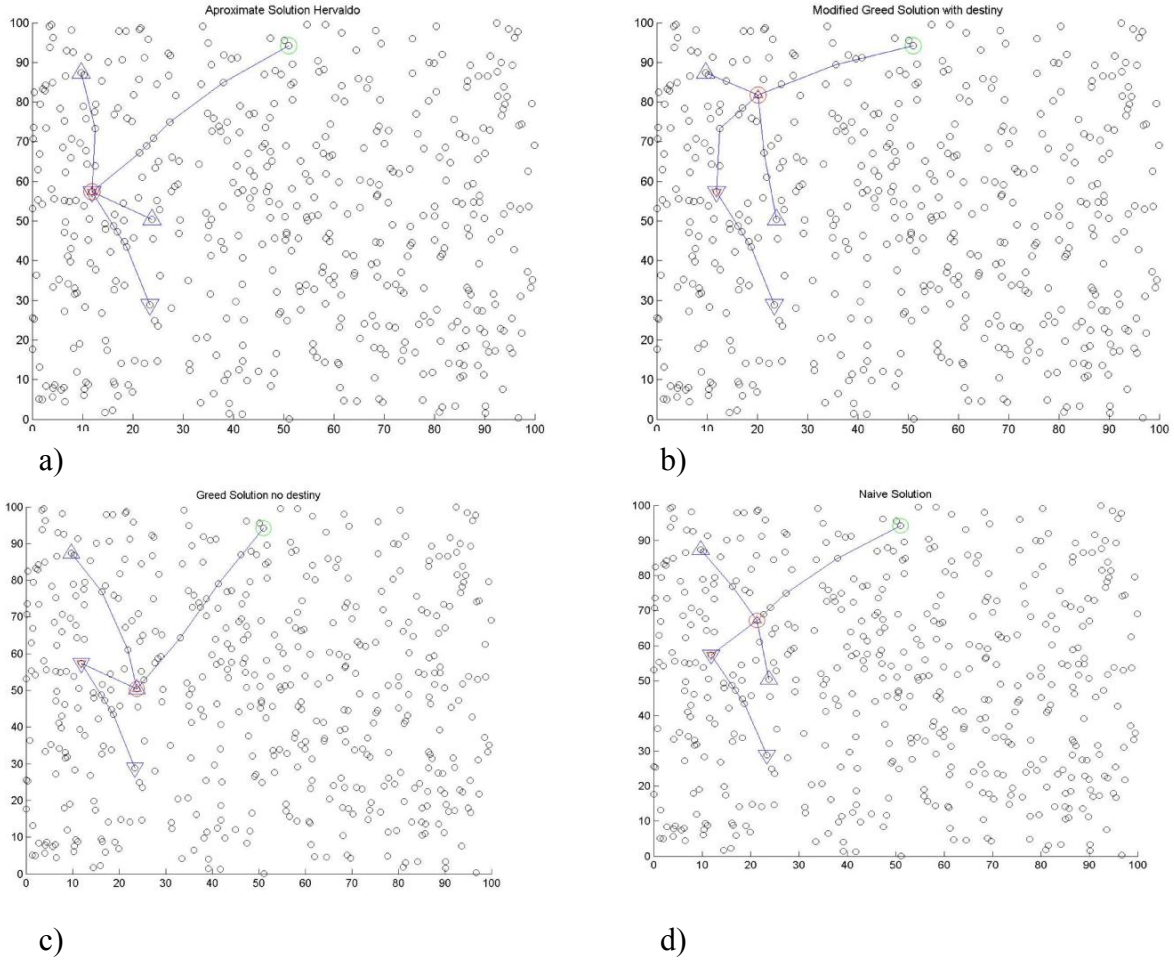


Figure 47: Example of data fusion path in different algorithms: approximate Hervaldo (a), Modified Greedy (b), Greedy (c), and Optimal Node (d).

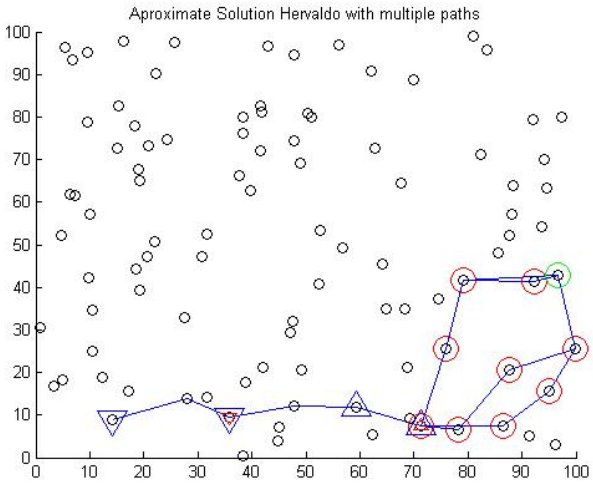


Figure 48: Example of the worst case scenario in data fusion multiple paths.

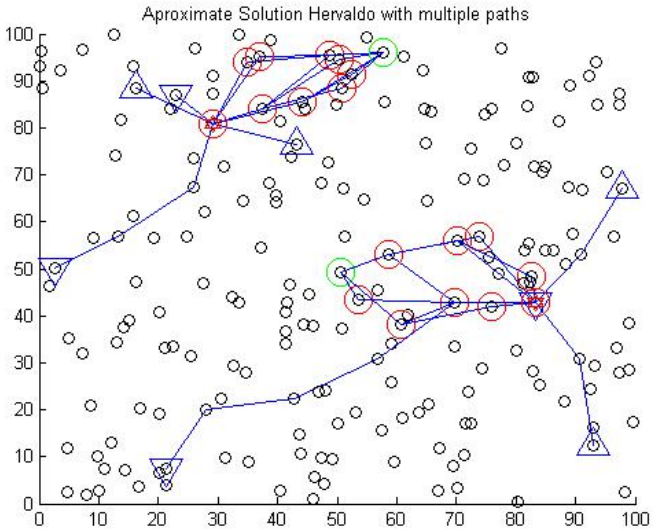


Figure 49: Examples of the worst and best case scenarios in data fusion multiple paths.

Figure 47 shows examples of data fusion paths from different algorithms. Notice that all the algorithms have different data fusion paths to solve the same data fusion expression ((A+B) + (C+D) and to send the data fusion result to the destination node. Figures 48 and 49 show that multiple data fusion paths can have good or bad results. If a node is a sensing unit and data fusion node simultaneously, its energy will drain earlier. So, the worst case scenario is the situation where the same nodes act as sensing units and data fusion node or destination node. .The best data fusion multiple paths solutions is

characterized by the use of different nodes to act as sensing units, data fusion nodes and destination nodes. If the system uses more nodes with approximately the same communication cost, the system's lifetime will be greater.

As a conclusion, the sensor networks application designer should balance minimum cost and maximum lifetime: The goal is to achieve the minimum communication and data fusion cost, and to maximize the system's lifetime.

So, the question is how to increase the number of paths. One of the ways is to vary the data fusion nodes (figure 50). As we are employing the shortest path, whether we choose the best 10 shortest paths, we might be varying the data fusion node. Another way is to use multiple DF nodes between the optimal DF node and destination node (figure 51).

Multiple optimal DF nodes (best 10 shortest paths)

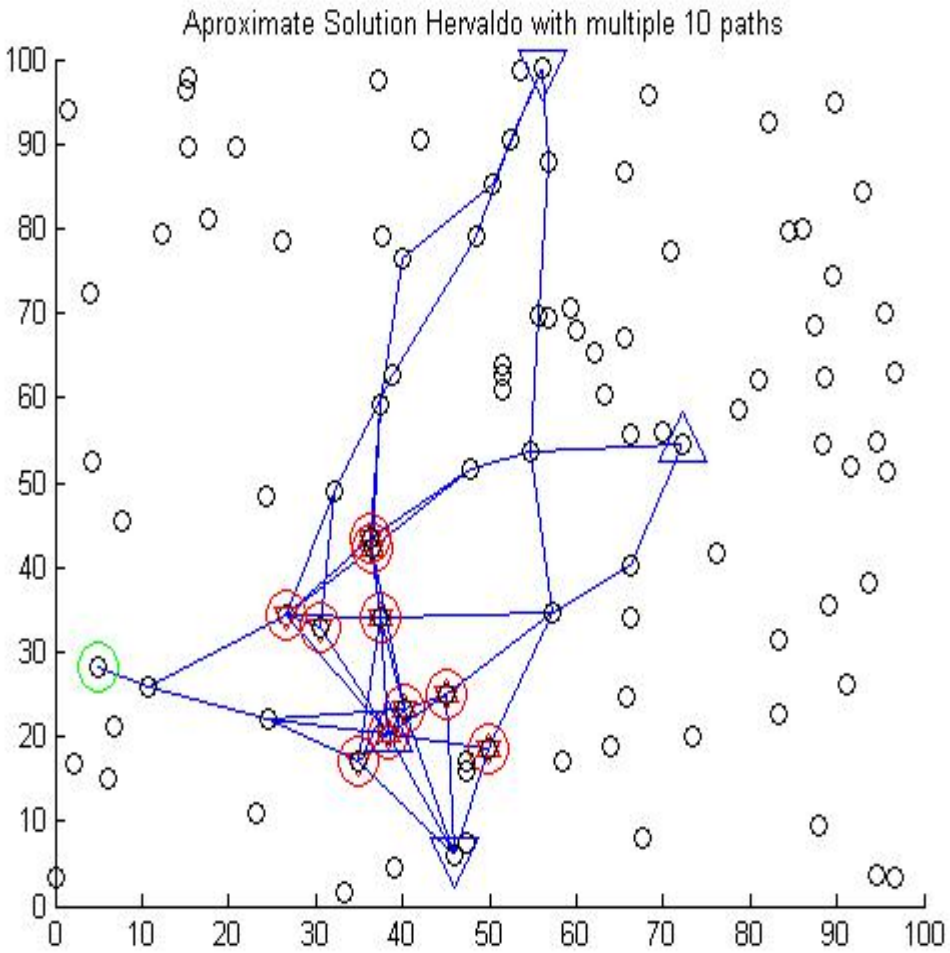


Figure 50: Example of the 10 best shortest paths of the approximate solution Hervaldo.

Multiple DF nodes between the optimal DF node and destination node

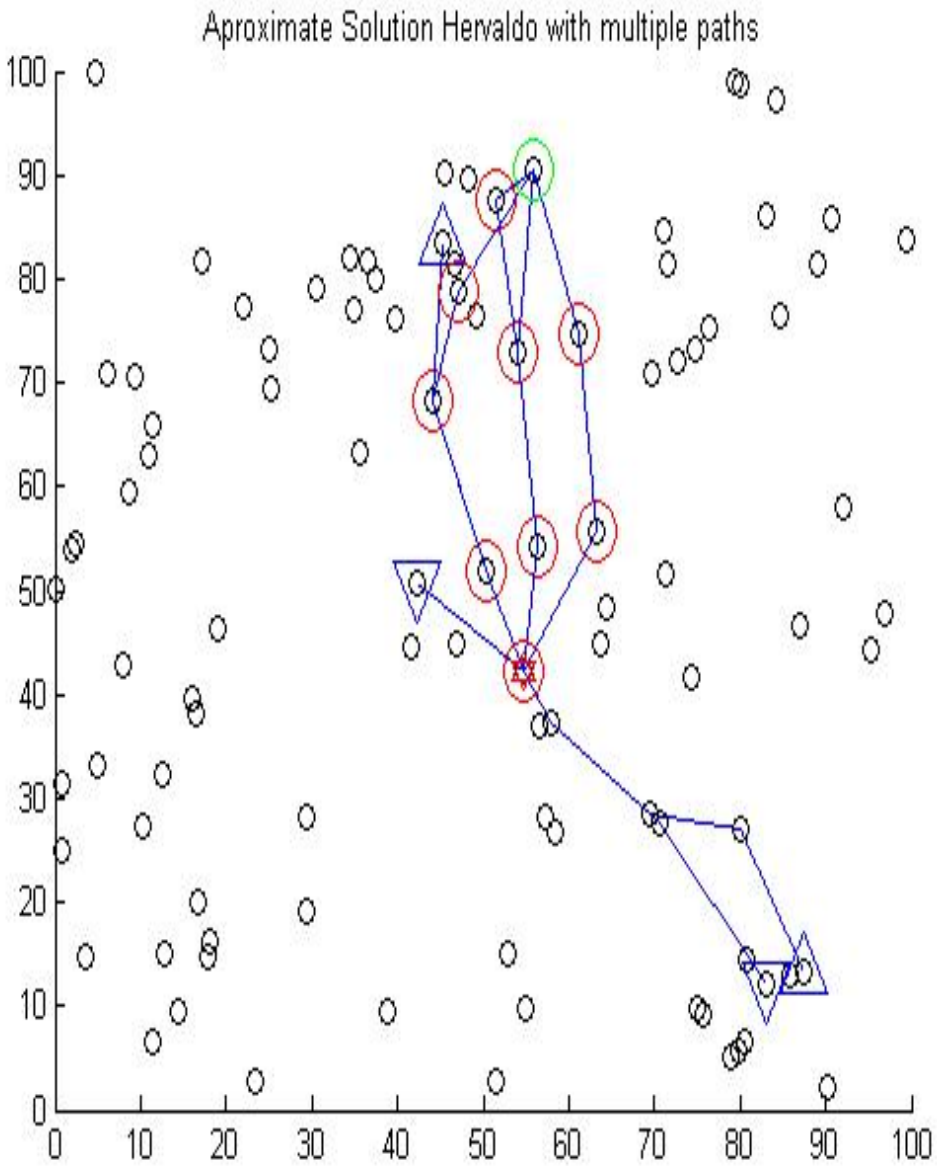


Figure 51: Example of the 10 best shortest paths between the (AB+CD) data fusion node and destination node of the approximate solution Hervaldo.

Network with 100 nodes, 10 different set of sensors, for each set of sensors 10 paths.

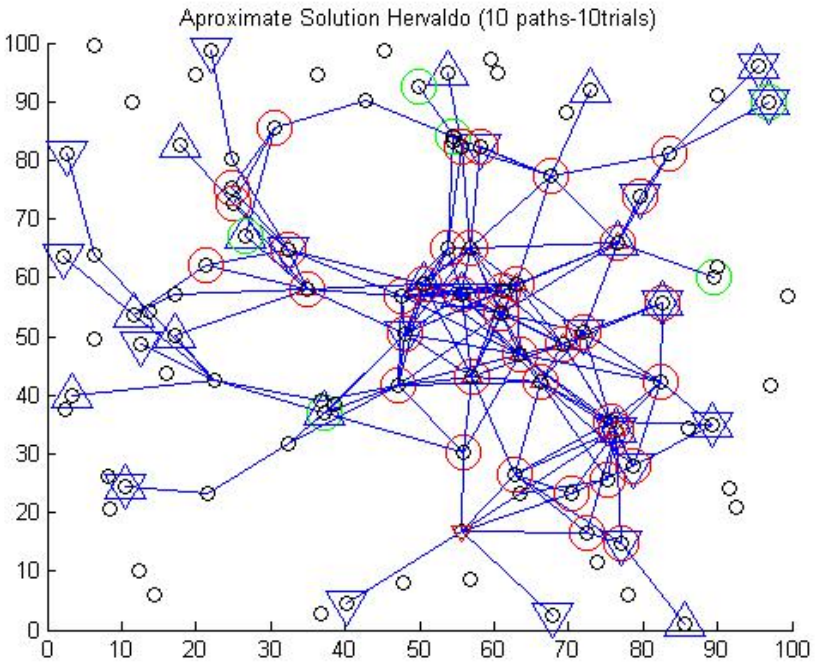


Figure 52: Example of a Network with 100 nodes and 10 different set of sensors. For each set of sensors, we considered the 10 best paths (lowest communication cost) of the approximate solution Hervaldo.

Figure 52 shows that network density should be considered in the design of sensor networks applications. A network with a hundred nodes and ten sensors as source of data will use almost all nodes to perform 10 distributed data fusion paths to each combination of sensors and destination node. This result shows that the number of network's nodes is a limiting factor to increasing the system's lifetime.

Another aspect that should be considered is the communication data fusion cost tradeoff. If the data fusion computation cost is higher than the communication cost, in general it is better to increase the number of paths and data fusion nodes. On the contrary, it is better to use the fewest number of paths.

In the next simulations we are going to perform the balance of the communication cost and data fusion cost in the different algorithms. As the optimal node and Hervaldo's solution were considered the best approximate solutions, from this point to the end we are going to consider only these algorithms in comparison to the optimal solution (shortest path).

d. Comparison of system's lifetime among the best different approximate algorithms based on multiple DF paths: One optimal node considering destination node and not considering destination node (aprox. Optimal DF node no sensor), and Approximate Hervaldo.

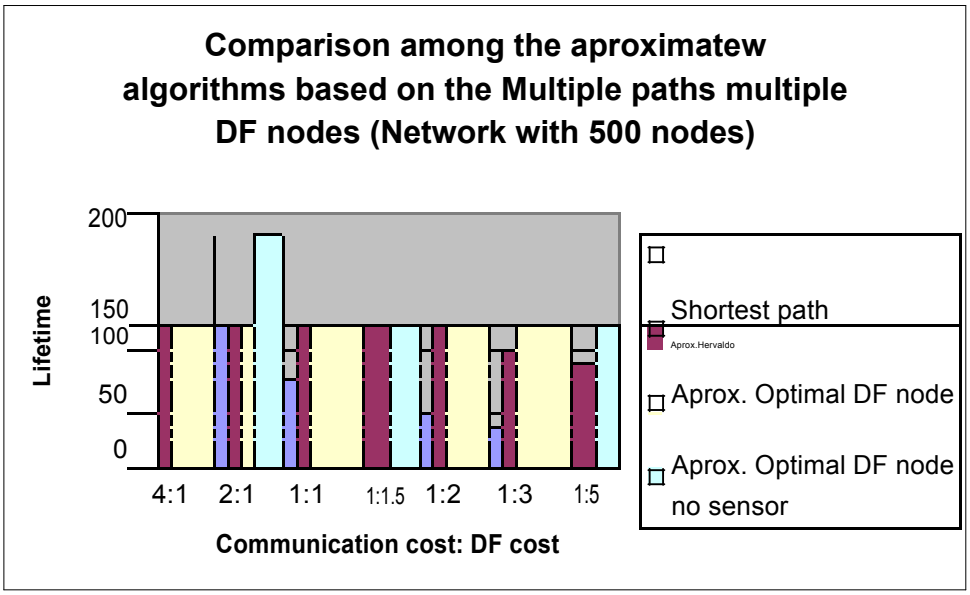


Figure 53: Network with 500 nodes, 10 multiple paths combined to 10 multiple data fusion nodes, for the shortest path, approximate Hervaldo, Approximate Optimal node and approximate optimal node no sensor.

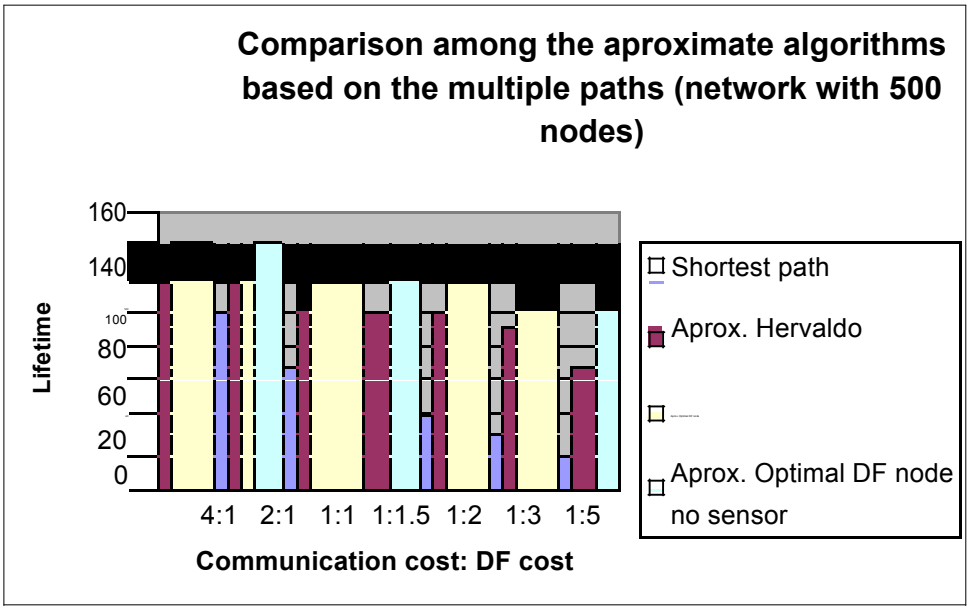


Figure 54: Network with 500 nodes, and 10 multiple paths, for the shortest path, approximate Hervaldo, Approximate Optimal node and approximate optimal node no sensor.

Figures 53 and 54 try to answer the question. Is it good to increase the number of nodes between the last data fusion node (AB+CD) and the destination node? The hypothesis considers that whether the data fusion cost is greater than the communication cost, it would be good to increase the number of DF nodes. The results showed that the use of the combination multiple paths and multiple nodes increase the system's lifetime. This increase is greater in the algorithms that vary the nodes in the different paths. The 10 shortest paths in the shortest path algorithm use almost the same nodes. On the other hand, the approximate solution based on the best node not considering the destination node achieves the best system's lifetime because it creates more paths with different nodes. So-, perhaps considering only the system's lifetime, the shortest path is not the best solution to achieve the optimal data fusion nodes to solve the data fusion expression.

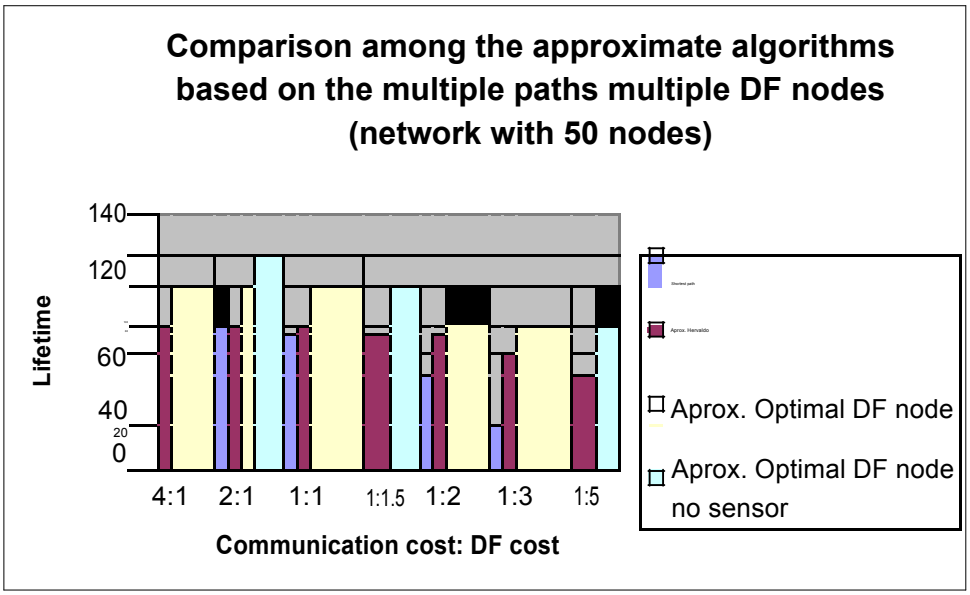


Figure 55: Network with 50 nodes, and 10 multiple paths combined to multiple DF nodes between the last DF node and destination node for the shortest path, approximate Hervaldo, Approximate Optimal node and approximate optimal node no sensor.

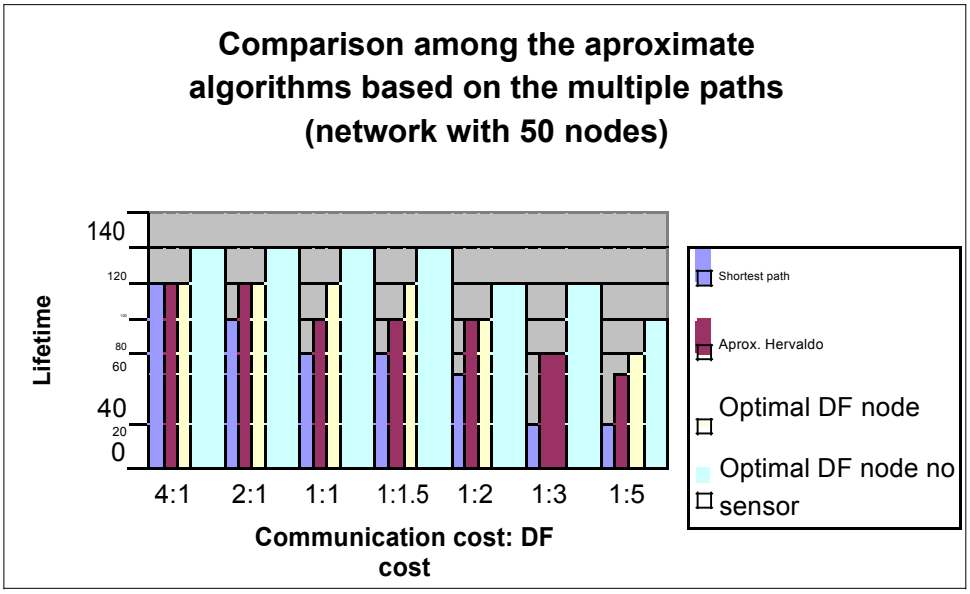


Figure 56: Network with 50 nodes, and 10 multiple paths for the shortest path, approximate Hervaldo, Approximate Optimal node and approximate optimal node no sensor.

The result of figures 55 and 56 show the same results as the figures 53 and 54. The use of the combination multiple paths and multiple nodes increase the system's lifetime. This increase is greater in the algorithms that vary the nodes in the different paths. It also shows that the network density is proportional to system's lifetime if the number of sensing units and DF nodes are constant. A network with 50 nodes will have a lower system's lifetime than a network with 500 nodes

e. Comparison of lifetime among the different approximate algorithms based on multiple DF paths considering 10, 50 and 100 paths: approximate based on greedy solution, modified greedy, one optimal node, and Approximate Hervaldo.

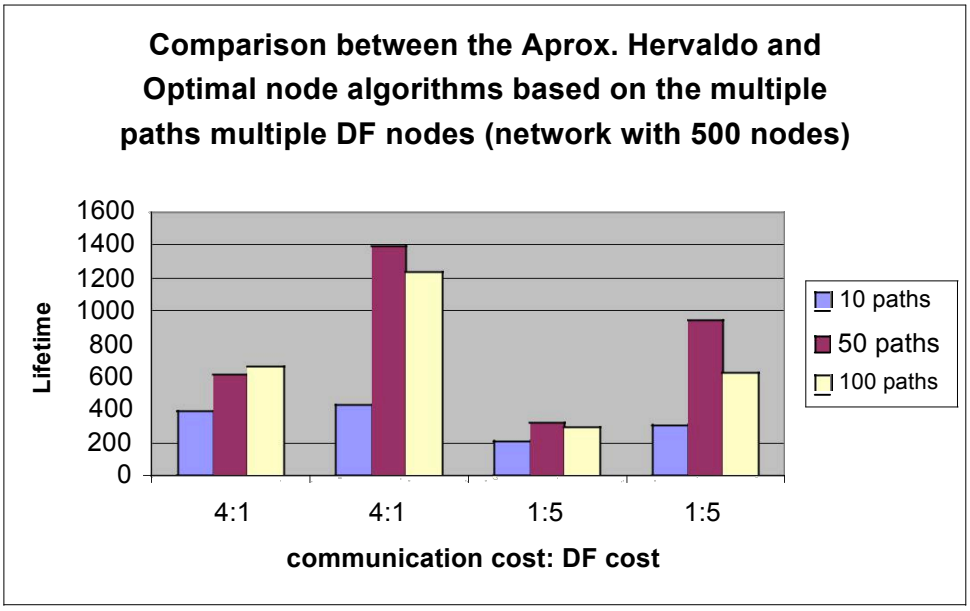


Figure 57: Comparison between Aprox Hervaldo algorithm and Optimal Node algorithm in a network with 500 nodes, multiple paths and multiple DF nodes.

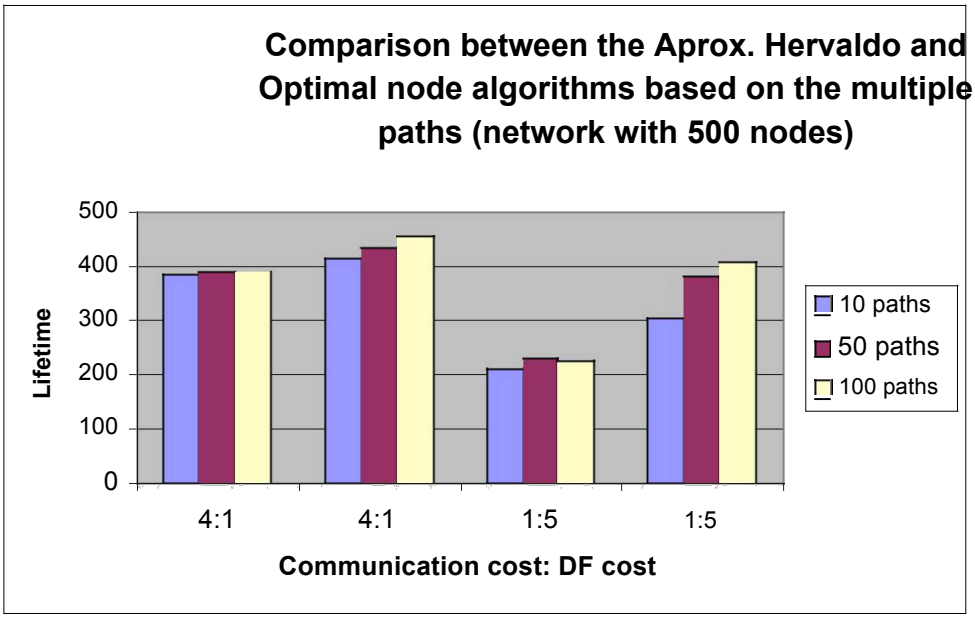


Figure 58: Comparison between Aprox Hervaldo algorithm and Optimal Node algorithm in a network with 500 nodes with multiple paths.

Figures 57 and 58 show the comparison between Approximate Hervaldo and Optimal node algorithms in network of 500 nodes, but varying the communication DF

cost and the number of paths. In figure 58 the simulation applied multiple paths multiple DF nodes between the AB+CD DF node and destination node, while the figure 59 only considered multiple paths. The results showed that the optimal node algorithm is better than Hervaldo's algorithm. Therefore, there is a great increase in system's lifetime to add DF nodes between the AB+CD DF node and destination node. The increment in the number of paths also increases the system's lifetime. When the communication cost is higher than the DF cost, the system's lifetime is higher.

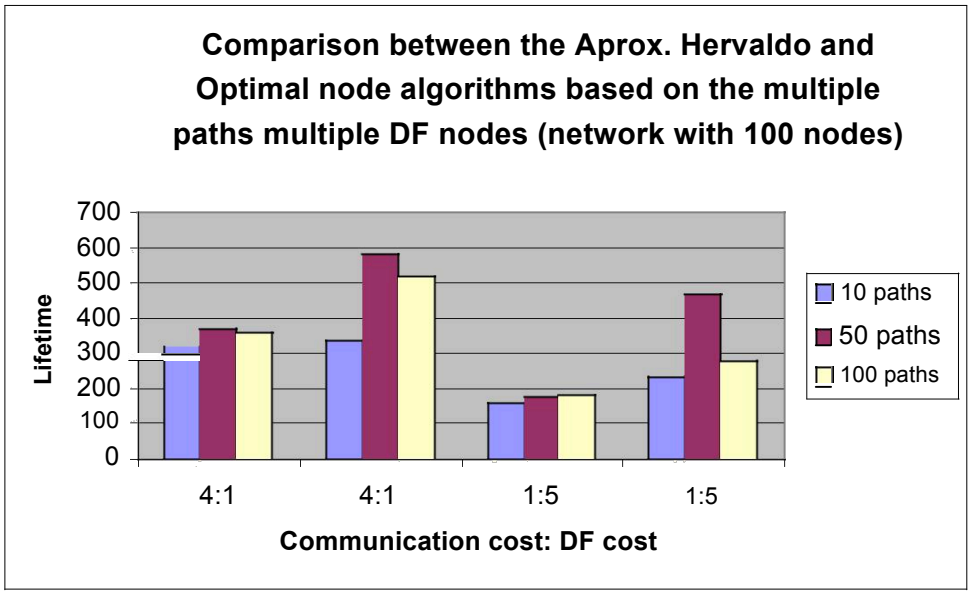


Figure 59: Comparison between Aprox Hervaldo algorithm and Optimal Node algorithm in a network with 100 nodes with multiple paths multiple DF nodes.

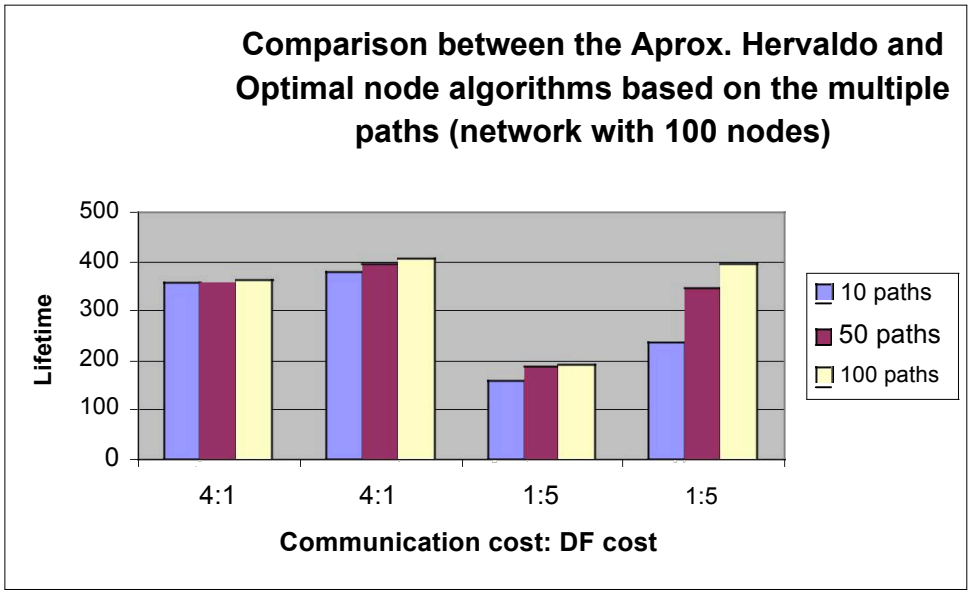


Figure 60: Comparison between Aprox Hervaldo algorithm and Optimal Node algorithm in a network with 500 nodes with multiple paths.

Figures 59 and 60 show that the difference between the approximate algorithm remain the same in a network with 100 nodes. It also shows that to decrease the network density decrease the system's lifetime.

f. Comparison of the system's lifetime among the different algorithms based on multiple DF paths considering the communication cost and Data fusion cost ratio and varying the number of paths: approximate solutions based on Approximate Hervaldo, one optimal node, and one optimal node without sensors.

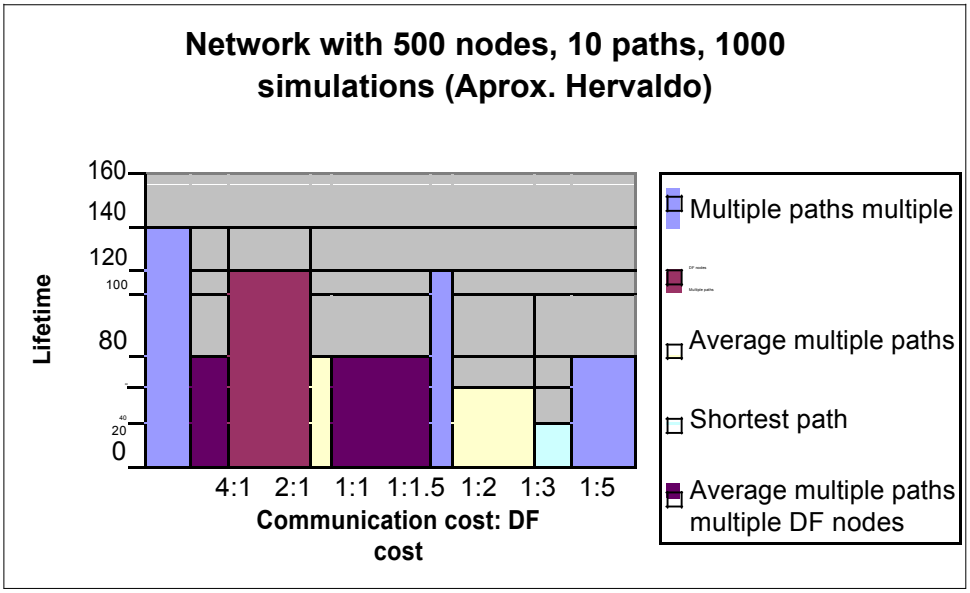


Figure 61: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Hervaldo algorithm in a network with 500 nodes with 10 paths.

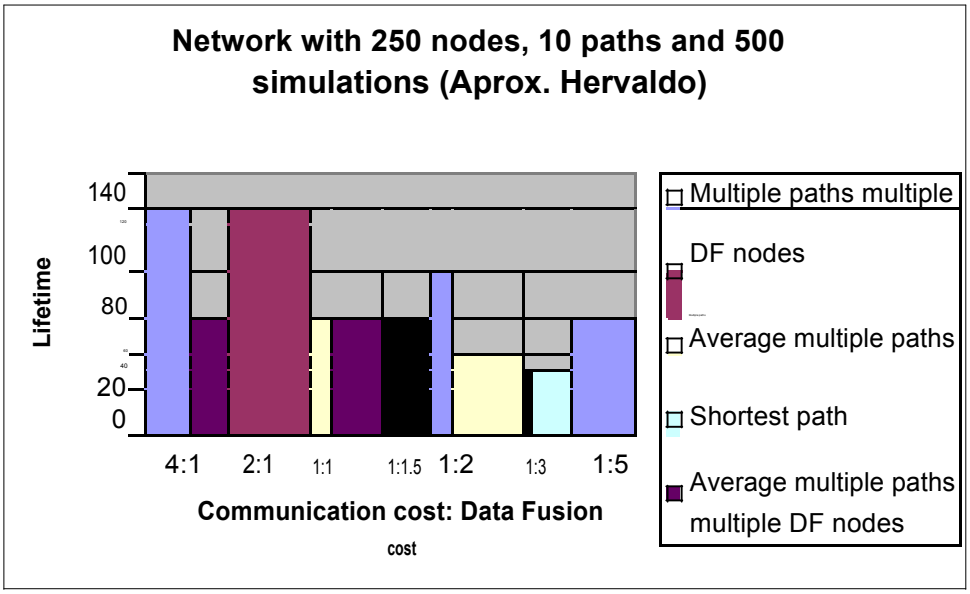


Figure 62: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Hervaldo algorithm in a network with 250 nodes with 10 paths.

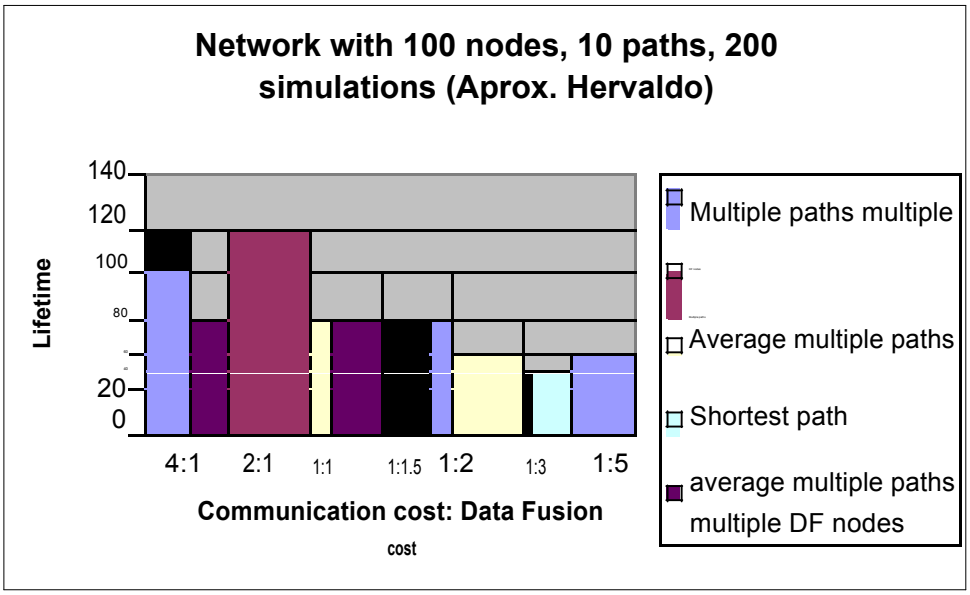


Figure 63: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Hervaldo algorithm in a network with 100 nodes with 10 paths.

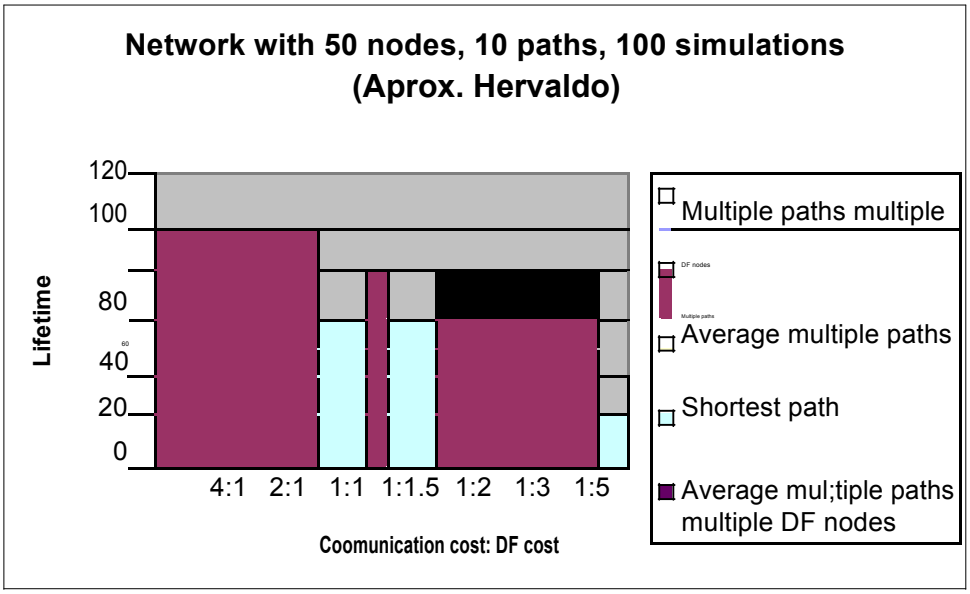


Figure 64: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Hervaldo algorithm in a network with 50 nodes with 10 paths.

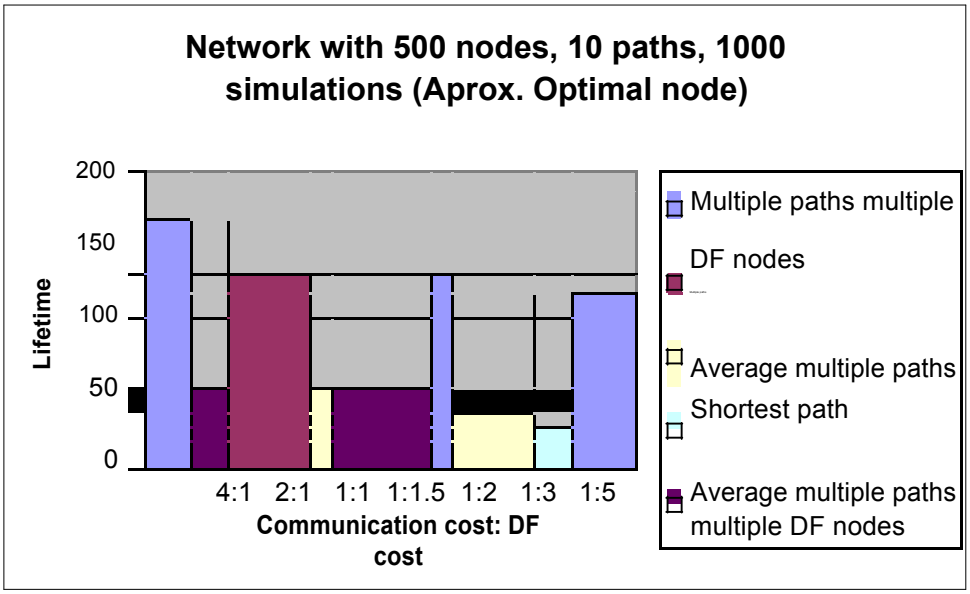


Figure 65: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node algorithm in a network with 500 nodes and 10 paths.

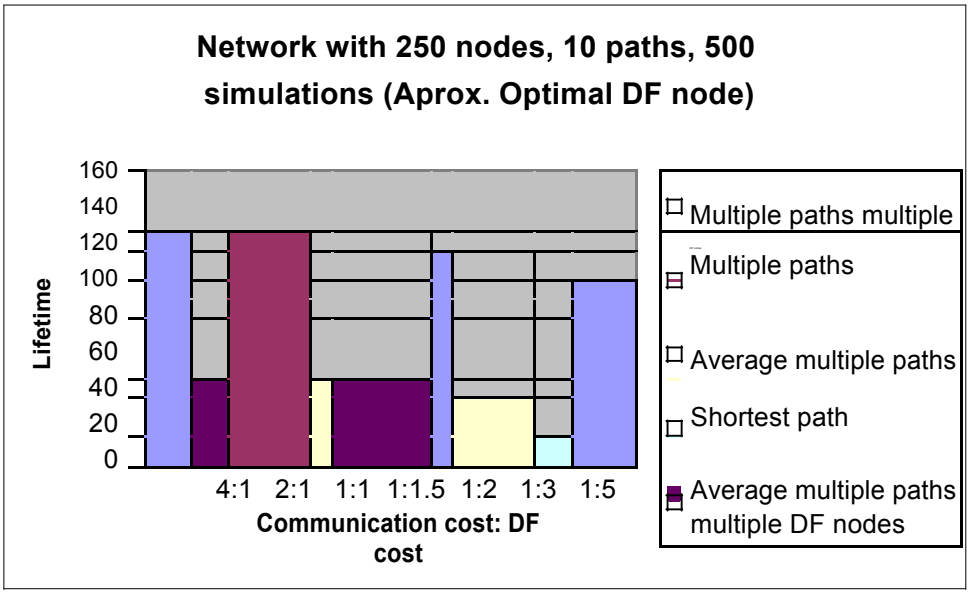


Figure 66: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node algorithm in a network with 250 nodes and 10 paths.

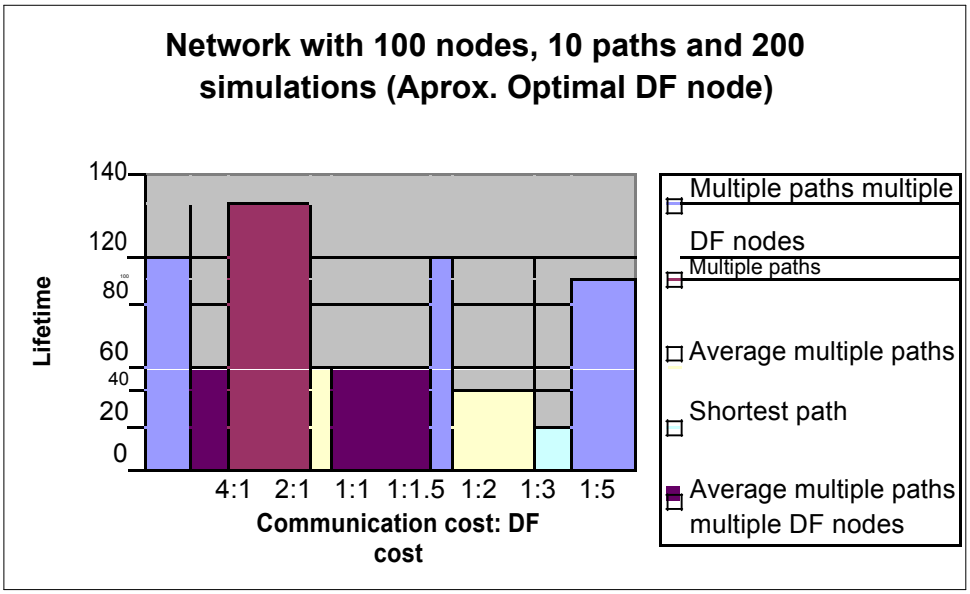


Figure 67: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node algorithm in a network with 100 nodes and 10 paths.

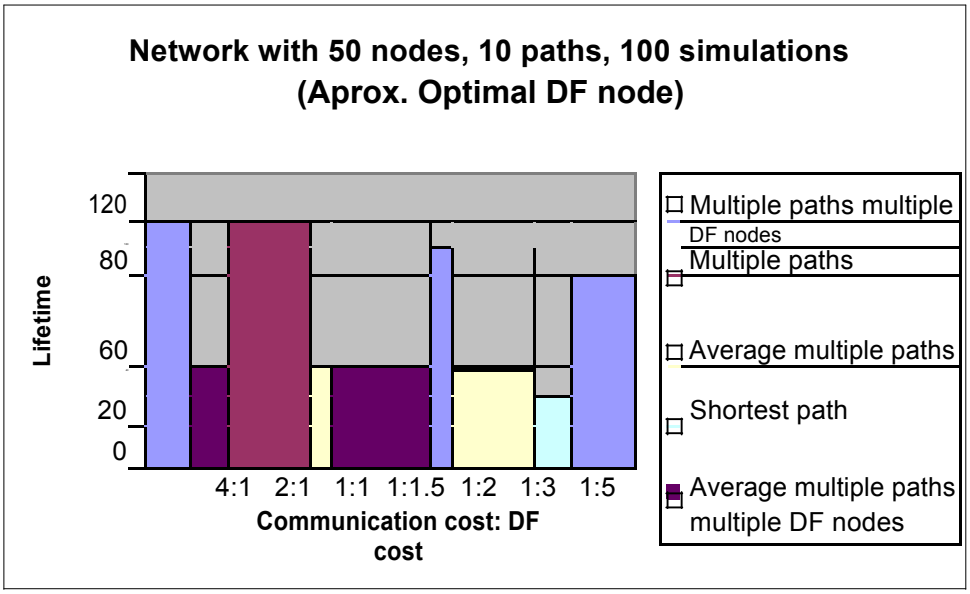


Figure 68: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node algorithm in a network with 50 nodes and 10 paths.

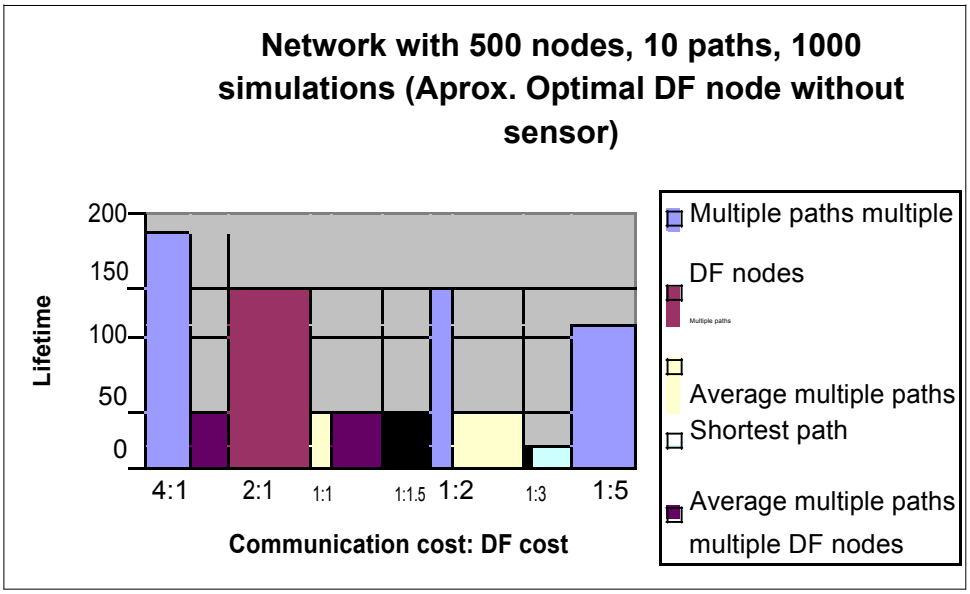


Figure 69: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node no sensor algorithm in a network with 500 nodes and 10 paths.

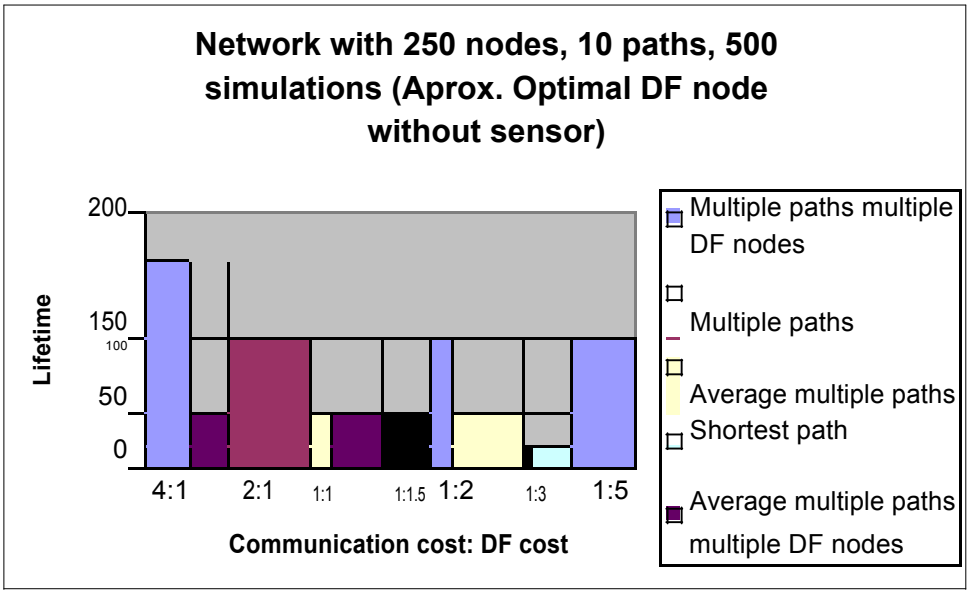


Figure 70: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node no sensor algorithm in a network with 250 nodes and 10 paths.

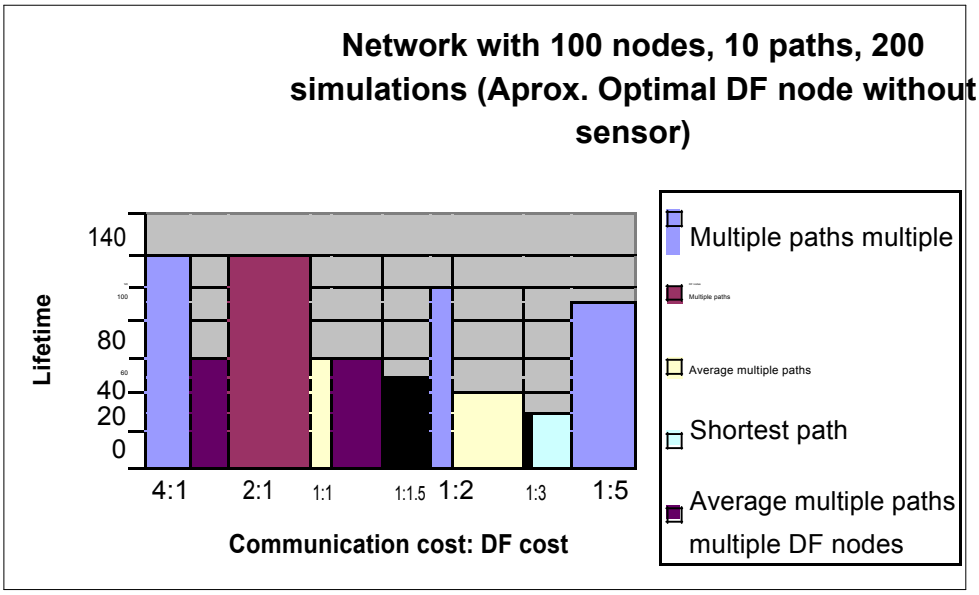


Figure 71: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node no sensor algorithm in a network with 100 nodes and 10 paths.

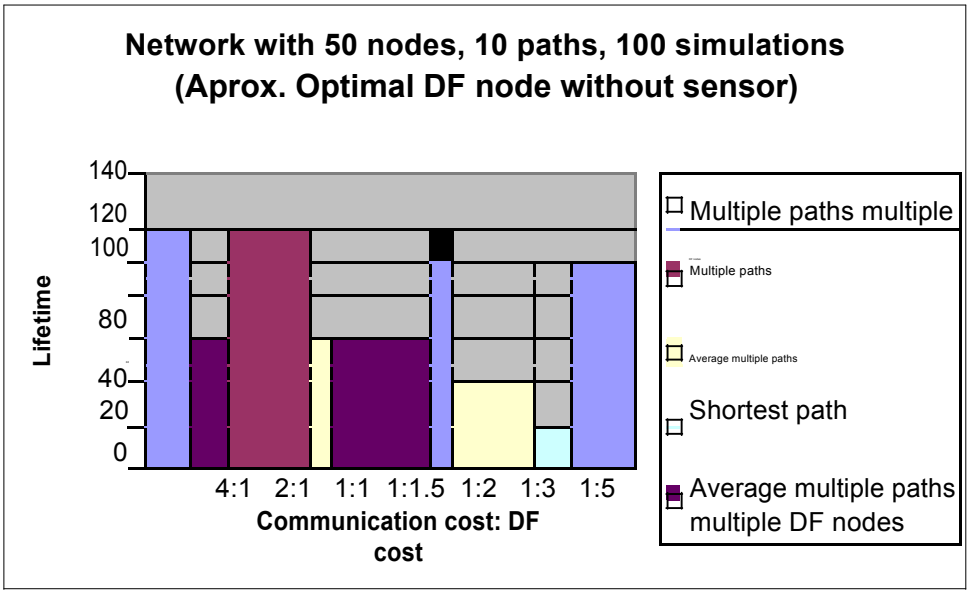


Figure 72: Comparison between multiple paths multiple DF nodes, multiple paths, average multiple paths, shortest path and average multiple paths multiple DF nodes using Aprox Optimal Node no sensor algorithm in a network with 50 nodes and 10 paths.

Figures 61 to 72 show that in all algorithms tested the network based on multiple paths multiple DF nodes is much better than the options multiple paths, average multiple paths, shortest path (better communication cost) and average multiple paths multiple DF nodes. These differences are greater when we increase the communication data fusion cost ratio. The comparison among the algorithms showed that the Approximate Optimal DF Node without sensor is slighter better than the Optimal DF node and much better than the Hervaldo's algorithm.

g. Compare the lifetime of first 10 paths, 20 to 30, 30 to 40, 50 to 60, 60 to 70, 70 to 80, 80 to 90 and 90 to 100 in different network densities.

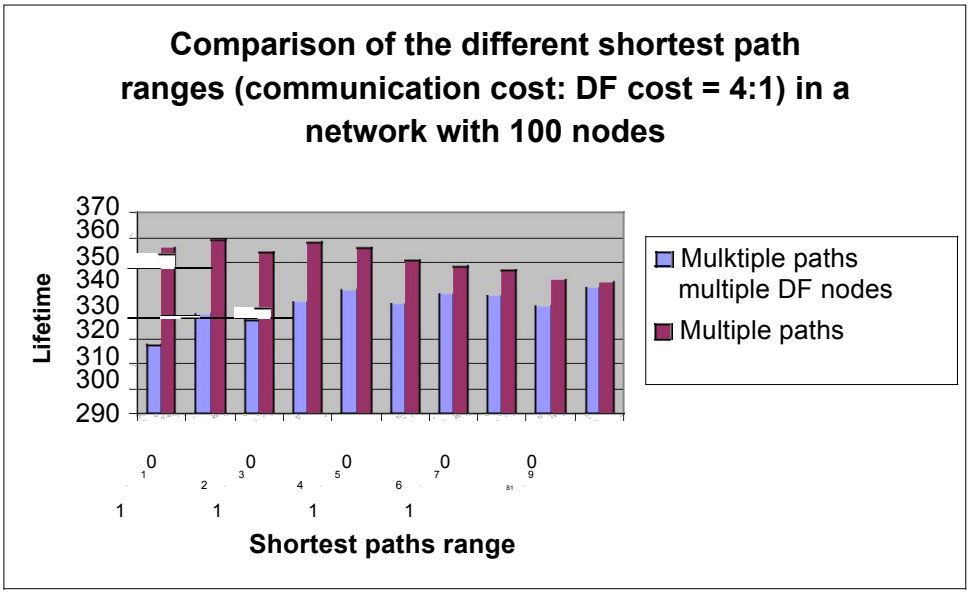


Figure 73: Comparison between multiple paths multiple DF nodes and multiple paths, in a network with 100 nodes and communication cost DF cost of 4:1 using optimal node algorithm.

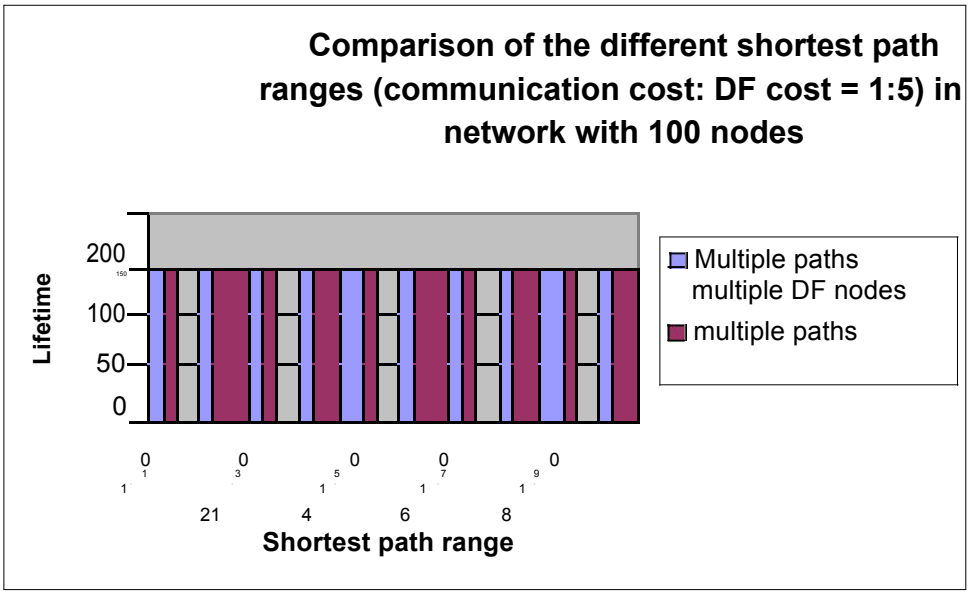


Figure 74: Comparison between multiple paths multiple DF nodes and multiple paths, in a network with 100 nodes and communication cost DF cost of 1:5 using optimal node algorithm.

The hypothesis tested and showed in figures 73 and 74 is that to use long paths can be better than to use the shortest paths. To test this hypothesis we evaluated the best

100 paths divided in ranges of ten. The results showed that it not favorable to use longest paths to increase system's lifetime in low density networks. It should be tested in high density networks.

How to employ the algorithms discussed in this chapter in real problems? In the next section we describe how to employ the Distributed Data Fusion Algorithm in a sensor networks application.

Distributed Data Fusion Algorithm: With the algorithm described below we can employ anyone of the approximate algorithms tested (greedy, modified greedy, one optimal node, and approximate Hervaldo). The algorithms will be different in the number of fusion nodes and how to calculate them (step 5 of the algorithm).

1) Destination node or destination nodes broadcast the data fusion expression.
2) Sensors read the expression and broadcast their positions (GPS?) whether they are participating in the expression. .
3) Find the optimal fusion node: each node of the network evaluate the distance (number of hopes) from the participating sensors and destination sensor to itself.
4) Each node broadcast its distance cost.
5) Each node builds an ordered vector of shortest paths.
6) Two options:
    a. Based on its data rates each participating sensor sends its data to the fusion point. It controls the data rate, the rotating of the fusion points (multiple paths), but it cannot control the rotation of redundant sensors. The fusion node can control the fusion point (multiple data fusion point's paths).
    b. Each destination node pulls the data fusion node and the data fusion node pulls the participating sensors. This approach allows the rotation of fusion nodes, participating sensors, sensors mode, and control the sensors' data rates, but the approach is related to an overhead of communication and control.

Conclusions:

To achieve the goal of performing data fusion in an optimal node in distributed sensor networks, we have implemented different algorithms already described in the literature (shortest path) and greedy algorithm. The other algorithms are adaptations of these algorithms. The suboptimal´s algorithms (hervaldo's algorithm and optimal node algorithm) showed to be the algorithms with best performance and acceptable cost.

The results obtained from the simulations in different network densities (100 to 500 nodes) showed that the approximate solution named Hervaldo, Modified Greedy, Optimal node and Greddy as the best alternatives to the optimal solution. In conclusion, the approximate solution Hervaldo is the best approximate solution and as a consequence, the nearest solution to the optimal solution.

The results obtained from the computation cost simulations showed that the approximate solutions are less expensive than the optimal solution. The comparison of the computation cost among the aproximate solutions showed that the Greedy, modified

Greedy and Hervaldo have the same computation cost. The computation cost of the optimal node is only 20% (1/5) of the other approximate solutions. As a conclusion, the optimal node solution has the lowest computation cost. As a result of the simulations considering the communication cost and the computation cost, the best alternatives to the optimal solution are the approximate Hervaldo and the optimal node solution.

Multiple data fusion paths can have good or bad results. If a node is a sensing unit and data fusion node simultaneously, its energy will drain earlier and the system´s lifetime will be shorter. So, the worst case scenario is the situation where the same nodes act as sensing units and data fusion node or destination node. The best data fusion multiple paths solutions is characterized by the use of different node to act as sensing units, data fusion nodes and destination nodes. If the system uses more nodes with approximately the same communication cost, the system's lifetime will be greater. As a conclusion, the sensor networks application designer should balance minimum cost and maximum lifetime: The system´s goal is to achieve the minimum communication and data fusion cost, and to maximize the system's lifetime.

Network density should be considered in the design of sensor networks applications. A network with a hundred nodes and ten sensors as source of data will use almost all nodes to perform 10 distributed data fusion paths to each combination of sensors and destination node. This result shows that the number of network nodes is a limiting factor to increase the system's lifetime.

Another aspect that should be considered is the communication data fusion cost tradeoff. If the data fusion computation cost is higher than the communication cost, in general it is better to increase the number of paths and data fusion nodes. To the contrary is better to use the fewest number of paths.

The use of the combination multiple paths and multiple nodes increase the system's lifetime. This increase is greater in the algorithms that vary the nodes in the different paths. The 10 shortest paths in the shortest path algorithm use almost the same nodes. On the other hand, the approximate solution based on the best node not considering the destination node achieves the best system's lifetime because it creates more paths with different nodes. So, considering the system's lifetime, the shortest path is not the best solution to achieve the optimal data fusion nodes to solve the data fusion expression. Besides this, network density is proportional to the system's lifetime if the number of sensing units and DF nodes are constant. A network with 50 nodes will have a lower system's lifetime than a network with 500 nodes.

The optimal node algorithm is better than Hervaldo's algorithm. Therefore, there is a great increase in system's lifetime to add DF nodes between the (AB+CD) DF node and destination node. The increment in the number of paths also increases the system's lifetime. When the communication cost is higher than the DF cost, the system's lifetime is higher.

In all algorithms tested, the network based on multiple paths multiple DF nodes is much better than the options multiple paths, average multiple paths, shortest path (better communication cost) and average multiple paths multiple DF nodes. These differences are greater when we increase the communication data fusion cost ratio. The comparison among the algorithms showed that the Approximate Optimal DF Node without sensor is slighter better than the Optimal DF node and much better than the Hervaldo's algorithm.

The hypothesis that to use long paths can be better than using shortest paths was evaluated by testing the performance of the Optimal Node algorithm in the best 100 paths divided in ranges of ten. The results showed that it not favorable to use longest paths to increase system's lifetime in low density networks. It should be tested in high density networks.

As a future work we are planning to compare the number of hopes of first 10 paths, 20 to 30, 30 to 40, 50 to 60, 60 to 70, 70 to 80, 80 to 90 and 90 to 100 in different network densities; compare the result using heterogeneous nodes; simulate a network with multiple sensors of the same type and rotation among them; simulate a network with multiple sensors of the same type and rotation among them. Compare the number of paths influence on system's lifetime; compare between shortest paths and longest paths: number of hopes, system's lifetime, delay, throughput, data rates and time synchronization.

Chapter 6

Middleware Application relationship in sensor networks

6.1 Introduction

For several decades, distributed computing has been both an enabling and a challenging environment in which to build applications. Initially, the major difficulty in implementing such systems was simply exchanging data across distances and among heterogeneous components. Today these problems are essentially solved, and research is turning its focus to higher level concerns, such as improved fault tolerance through replication, optimal data access via distributed object placement, and methods of enabling high level communication abstractions such as event dispatching and remote invocation. The end result of this research into distributed systems is an expanding set of middleware platforms that reside above the operating system and below the application, abstracting lower level functionality such as network connectivity and providing a high-level coordination interface to the application programmer.

Often the combination of characteristics from the environment and application drive the design of the middleware. For example, consider the new class of applications for sensor networks with the following features:

Inherent distribution. The sensors are distributed throughout a physical space, and are primarily connected wirelessly;
Dynamic availability of data sources. Either mobility through space, addition of new sensors, or loss of existing sensors causes the set of available sensors to change over time;
Constrained application quality of service demands. Sensor network applications require a minimum quality of service (QoS), and this level must be maintained over an extended period of time. There may be many ways to achieve the QoS (e.g., different sensors may offer data or services that meet the applications©QoS requirements). Furthermore, the required QoS and the means of meeting this QoS can change over time, as the state of the application or the availability of sensors changes;
Resource limitations. Both network bandwidth and sensor energy are constrained. This is especially true when considering battery powered sensors and wireless networks;
Cooperative applications. Sensor network applications share available resources (e.g., sensor energy, channel bandwidth, etc.) and either cooperate to achieve a single goal, or, at the very least, do not compete for these limited resources.

One unique feature of sensor network applications with these properties is that simply responding to the changing environment is insufficient to achieve the required QoS over time. Instead, the applications must be proactive, actively affecting the network. Most existing middleware systems do not support such a proactive approach with respect to the network, leaving reactivity as the only choice and sacrificing

application quality over time. We believe that a middleware that enables applications to affect the network and the sensors themselves is needed to support this new and growing class of applications for sensor networks.

This chapter presents an overview of the related research in the areas of sensor networks and middleware, highlighting how existing approaches to the management of sensor networks could benefit from a middleware abstraction and showing that existing middleware does not meet the specification needs of all sensor network applications. Based on this observation, we propose a new middleware for sensor networks called Milan (Middleware Linking Applications and Networks). Milan allows sensor network applications to specify their quality needs and adjusts the network characteristics to increase application lifetime while still meeting those quality needs. Specifically, Milan receives information from the individual applications about their QoS requirements over time and how to meet these QoS requirements using different combinations of sensors, the overall system about the relative importance of the different applications, and the network about available sensors and resources such as sensor energy and channel bandwidth. Combining this information, Milan continuously adapts the network configuration (e.g., specifying which sensors should send data, which sensors should be routers in multi -hop networks, which sensors should play special roles in the network, etc.) to meet the applications©needs while maximizing application lifetime.

Next we will describe several sensor network applications that could benefit from a middleware like Milan that proactively affects different characteristics of the network, and in section 6.3 we will discuss existing sensor network management and middleware approaches. In section 6.4 we will describe Milan and show how the design of a health monitor sensor application can be simplified using Milan.

6.2 Sensor Network Applications

As stated in the introduction, sensor network applications represent a new class of applications that are:

> data driven, meaning that the applications collect and analyze data from the environment, and depending on redundancy, noise, and properties of the sensors themselves, the applications can assign a quality level to the data, and
>
> state based, meaning that the application©s needs with respect to sensor data can change over time based on previously received data. Typically sensors are battery-operated, meaning they have a limited lifetime during which they provide data to the application. A challenge of the design of sensor networks is how to maximize network lifetime while meeting application quality of service (QoS) requirements.

For these types of applications, the needs of the application should dictate which sensors are active and the role they play in the network topology. To further illustrate this point, we discuss some specific sensor network applications and how they can benefit from this form of interaction. Next we will give a brief description of the test bed application for the middleware application relationship proposed.

6.2.1. Medical Monitoring

As an example of a sensor networks application, consider a personal health monitor application running on a PDA that receives and analyzes data from a number of sensors (e.g., ECG, EMG, blood pressure, blood flow, pulse oxymeter). The monitor reacts to potential health risks and records health information in a local database. Considering that most sensors used by the personal health monitor will be battery operated and use wireless communication, it is clear that this application can benefit from intelligent sensor management that provides energy-efficiency as well as a way to manage QoS requirements, which may change over time with changes in the patient©s state. For example, higher quality might be required for certain health-related variables during high stress situations such as a medical emergency, and lower quality during low stress situations such as sleep. Next we will present a review of the related literature (section 6.3) and describe the middleware application relationship approach employed in the Milan architecture (section 6.4).

6.3 Sensor Network Management and Middleware Application Relationship Approaches

There has been considerable research into the development of low-level protocols to support sensor networks as well as high-level middleware systems to support the development of distributed computing applications by hiding environmental complexities. A recent trend includes the combination of these into middleware designed for sensor networks. In this section, we describe these developments and explain why they are insufficient for the unique style of many sensor network applications.

6.3.1. Sensor Networks

One of the distinguishing characteristics of sensor networks is their reliance on non-renewable batteries, despite their simultaneous need to remain active as long as possible. Therefore, initial work has been done to create network protocols tailored to sensor networks that extend network lifetime considering the energy constraints of the individual sensors. Some protocols make use of low-level node collaboration to reduce the energy cost of data transfer by aggregating data locally rather than sending all raw data to the application. For example, with LEACH [52], nodes form local clusters and all data within a cluster are aggregated by the cluster-head node before being transmitted to the base station. This limited form of low-level collaboration is also found in the query-based technique of Directed Diffusion [53], in which nodes collaborate to set up routes as interests for particular data are disseminated through the network. Another approach to reducing energy dissipation is to turn nodes o_ whenever possible. As idle power can often be significant, this approach can greatly extend application lifetime. MAC-level protocols, such as PAMAS [54] and S -MAC [55] use this technique to reduce energy dissipation in the MAC protocol, often trading o_ latency in packet delivery for energy efficiency. Topology control protocols such as ASCENT [56], Span [57], and STEM [58] use a similar technique of turning on and off sensors to maximize network lifetime while

keeping the network fully connected. Other topology control protocols such as Lint [59] aim to determine the minimum transmitting power necessary for a fully connected network, whereas protocols such as those described in [60, 61] determine the optimal transmitting power to minimize overall energy dissipation. In addition to the above two techniques, considerable energy can be saved by tailoring the routing protocol to the characteristics of sensor networks, including the energy constraints of the sensors, the data-driven nature of these networks, and the many-to-one, many-to-some, or many- to-many collection of the data. Sensor network routing protocols such as Rumor Routing [62], Directed Diffusion [53] and SPIN [63] provide lightweight, data-centric solutions tailored to typical sensor network traffic patterns. Although these protocols are effective in extending the lifetime of sensor networks, the gap between the protocol and the application is often too large to allow the protocols to be effectively used by application developers.

6.3.2. Middleware

Middleware has often been useful in traditional systems for bridging the gap between the operating system (a low-level component) and the application, easing the development of distributed applications. Because wireless sensor networks share many properties with traditional distributed systems, it is natural to consider distributed computing middleware for use in sensor networks. One of the most common middleware systems, Corba [64], hides the location of remote objects, simplifying the application©s interactions with these remote objects by allowing all operations to appear local. Although this could be applied to sensor networks to provide access to the sensor data, by hiding the location of the object (e.g., the sensor), the context information (e.g., the location) of the sensor is similarly lost. Additionally, by providing individual sensor access through objects, the potential energy savings by aggregation is lost. Jini©s [65] service discovery protocol and leasing mechanisms allow client applications to discover services and manage client-server connections as the set of available services changes. Service discovery is useful for dynamic sensor networks to know what sensors and/or services are available; however, access to services remains object-based, similar to Corba. The Lime middleware [66] focuses on a different API (application programming interface), namely a shared memory scheme for mobile ad hoc components through a Linda-like tuple space [67]. Neither Jini nor Lime considers the limited energy constraints of sensor networks, and their supporting protocols are heavyweight when compared to protocols tailored to sensor networks.

Some middleware acknowledge the changing properties of wireless networks and attempt to modify their own behavior to match the conditions detected within the network. For example, both Limbo [68] and FarGo [69] reorder data exchanges or relocate components to respond to changing network conditions such as bandwidth availability or link reliability. At a lower level, Mobiware [70] supports various levels of quality of service by adapting streams within the network with active filters deployed in the routers. Other middleware systems provide hooks to allow the applications to adapt. For example, applications built on the Odyssey platform [71] can register for notification of changes in the underlying network data rate. Similarly, the Spectra [72] component of Aura [73] monitors the network conditions and the accessible computation resources,

deciding where computation should be performed based on the network transmission required to complete them as well as the expense of the computation on mobile versus fixed nodes. These advances are applicable to wireless sensor networks; however, they do not integrate any of the specific data aggregation protocols of sensor networks, nor do they consider the details of the low-level wireless protocols.

Among existing distributed computing middleware, QoS-Aware Middleware [74] provides the closest example of a middleware that can support sensor network applications. This middleware is responsible for managing local operating system resources based on application requirements specified to the middleware. The application©s QoS information is compiled into a QoS profile to guide the middleware in making resource use decisions.

6.3.3 Middleware for Sensor Networks

Recently, much work has targeted the development of middleware specifically designed to meet the challenges of wireless sensor networks, focusing on the long-lived and resource-constrained aspects of these systems. Both the Cougar [85] and SINA [75] systems provide a distributed database interface to the information from a sensor network with database-style queries. Power is managed in Cougar by distributing the query among the sensor nodes to minimize the energy consumed to collect the data and calculate the query result. To support the database queries, SINA incorporates low-level mechanisms for hierarchical clustering of sensors for efficient data aggregation as well as protocols that limit the re-transmission of similar information from geographically proximate sensor nodes. AutoSec [76], Automatic Service Composition, manages resources in a sensor network by providing access control for applications so that qualities of service requests are maintained. This approach is similar to middleware for standard networks because resource constraints are met on a per-sensor basis, but the techniques for collecting the current resource utilization are tailored to the sensor network. DSWare [77] provides a similar kind of data service abstraction as AutoSec, but instead of the service being provided by a single sensor, it can be provided by a group of geographically close sensors. Therefore, DSWare can transparently manage sensor failures as long as enough sensors remain in an area to provide a valid measurement. While these middleware for sensor networks focus on the form of the data presented to the user applications, Impala [78], designed for use in the ZebraNet project, considers the application itself, exploiting mobile code techniques to change the functionality of the middleware executing at a remote sensor. The key to energy efficiency for Impala is for the sensor node applications to be as modular as possible, enabling small updates that require little transmission energy.

Although each of these middlewares is designed for efficient use of the wireless sensor network, they largely ignore the properties of the network itself. In other words, most of these approaches do not attempt to change the properties of the network in order to manage energy, and they are not flexible enough to support different protocol stacks or different applications©QoS requirements.

6.4. MILAN Project

As the summary of related work in the previous section shows, most sensor network research has focused on designing new network-level protocols (e.g., MAC layer, routing layer, topology control, etc.), without considering existing standards or how applications use the protocols. We argue that sensor network applications may be built on top of existing protocols, and thus some coordination framework is needed to leverage the exibility that exists in both standardized and non-standardized network protocols. However, to make these protocols more useful, application designers would benefit from a middleware that encapsulates the protocols, providing a high-level interface. Although the middleware discussed provide reasonable APIs, they either invent their own energy management protocols or provide limited mechanisms to adapt to the constraints of the wireless network. We argue that additional savings can be achieved if the middleware varies the actual parameters of the network over time while simultaneously meeting the requirements of the application, thereby increasing the lifetime of the network.

We are developing a new middleware named MILAN (Middleware Linking Applications and Networks) that receives a description of application requirements, monitors network conditions, and optimizes sensor and network configurations to maximize application lifetime. To accomplish these goals, applications represent their requirements to Milan through specialized graphs that incorporate state-based changes in application needs. Based on this information, Milan makes decisions about how to control the network, as well as the sensors themselves, to balance application QoS and energy efficiency, lengthening the lifetime of the application.

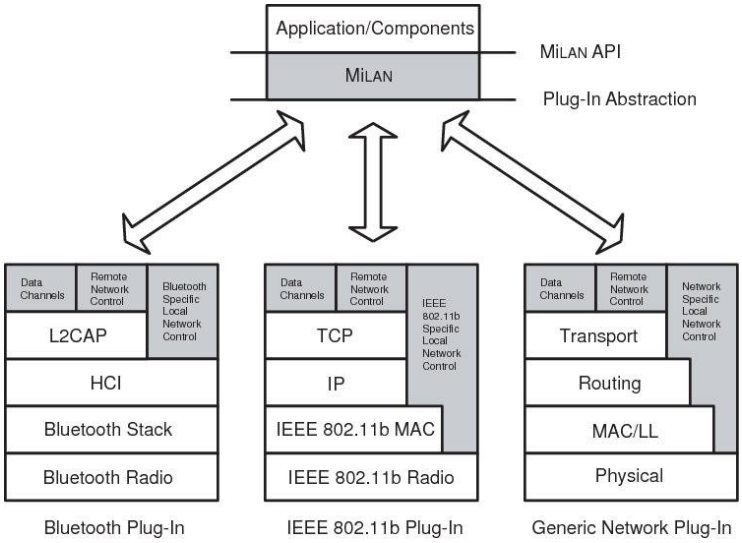


Figure 75: Overview of the interactions among Milan, the applications, and the sensors, together with a partial API.

Figure 75 shows an overview of the interactions among Milan, the applications, and the sensors, together with a partial API. This figure makes a distinction between the network plug-ins and the core of Milan, emphasizing the separation of computation that is specific to the selected network type, versus the computation that always occurs, but the API specifies only the application and sensor level operations. To make the description of the Milan API and the network plug-in abstraction more concrete, we use the personal health monitor application from section 6.2 as a running example. For more details about the MILAN project see [84].

6.5 Middleware Application Relationship proposed

The middleware application relationship can be achieved through the employment of two graphs. The first graph represents the sensors' accuracy to determine a certain variable and/or the accuracy resulting from the data fusion process. The second graph is based on the different states that an application can assume over time. These two graphs represent the knowledge-based application's Quality of Service necessity in different situations (states). This QoS system is named application performance. The middleware should manage the tradeoff among the different components of the entire system (network bandwidth, sensor management, application performance and life time, fault tolerance, among other factors).

6.5.1 Application Performance

Many sensor network applications are designed to receive data input from multiple sensors and to adapt as the available sensors change over time, either as new sensors come within range, or as sensors go offline when they move away or run out of energy. We assume that application performance can be described by the QoS of different variables of interest to the application, where the QoS of the different variables depends on which sensors provide data to the application. Sometimes two or more sensors can constitute a virtual sensor to provide a specific QoS to the application. A generic component QoS graph is showed in figure 76. We can have n sensors related to n virtual sensors with a cardinality of zero to n. So, the QoS can be provided to the application directly from a sensor or from the virtual sensor that is a set of two or more sensors. As an example, in the personal health monitor, variables such as blood pressure, respiratory rate, and heart rate may be determined based on measurements obtained from any of several sensors [79]. Each sensor has a certain QoS in characterizing each of the application©s variables. For example, a blood pressure sensor directly measures blood pressure, so it provides a quality (accuracy) of 1.0 in determining this variable. Quality is mapped to a specific reliability in determining the variable from the sensor©s data, with 1.0 corresponding to 100% reliability. In addition, the blood pressure sensor can indirectly measure other variables such as heart rate, so it provides some quality,

although less than 1.0, in determining these variables. The component QoS graph of the heart monitor is showed in figure 77.

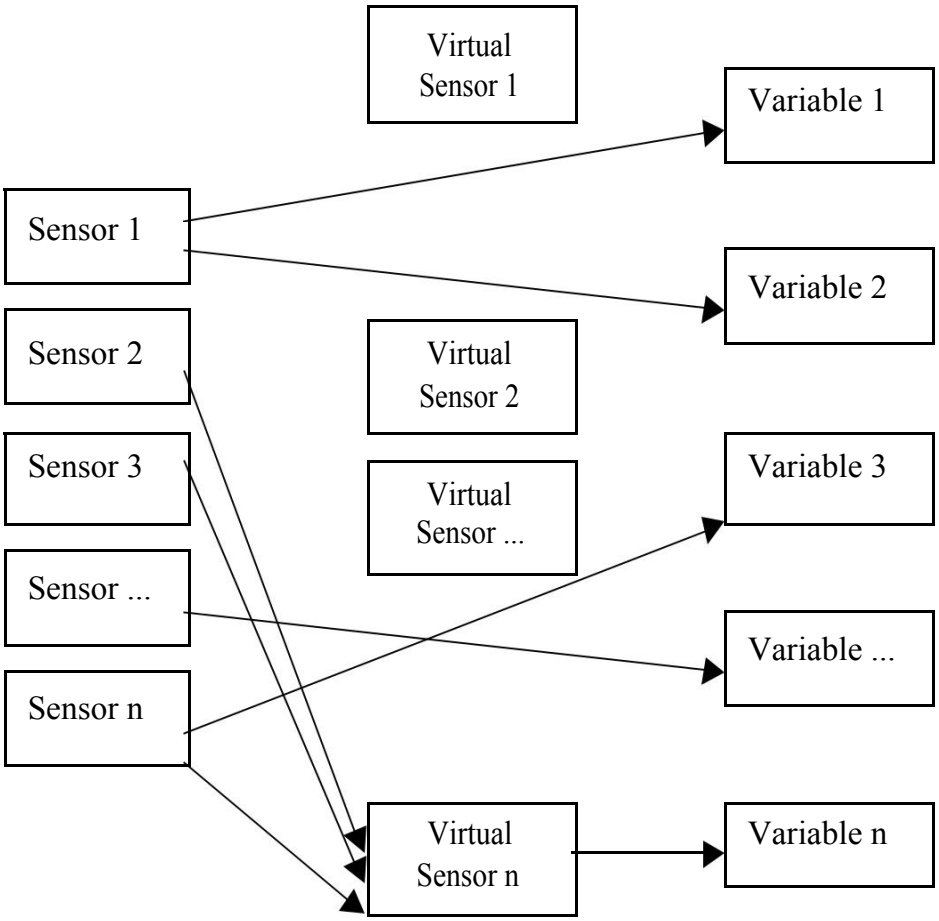


Figure 76: Generic Component

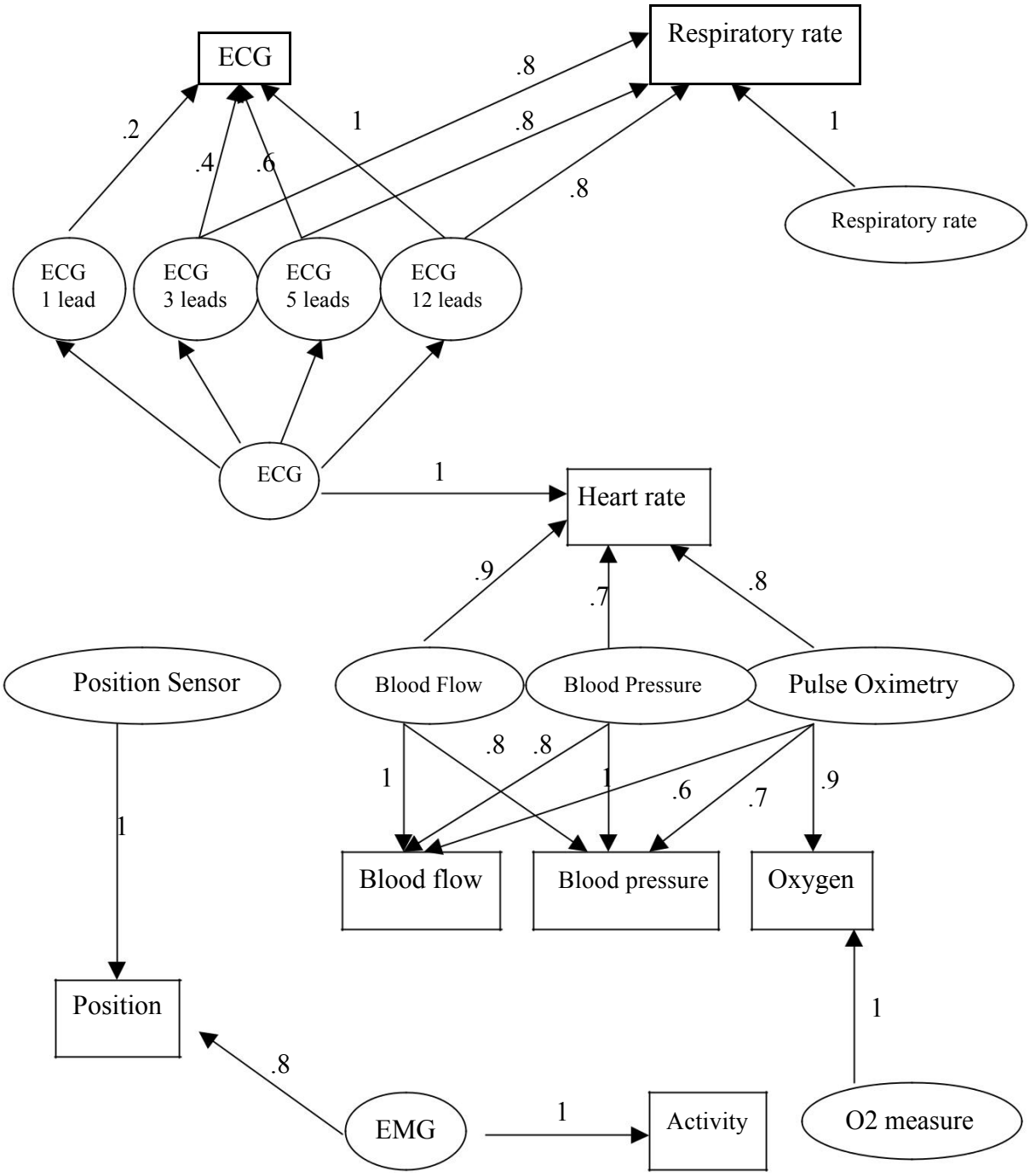


Figure 77: Heart Monitor Application Component QoS graph.

The quality of the heart rate measurement would be improved through high-level fusion of the blood pressure measurements with data from additional sensors such as a blood flow sensor. In order to determine how to best serve the application, Milan must know (1) the variables of interest to the application, (2) the required QoS for each variable, and (3) the level of QoS that data from each sensor or set of sensors can provide for each variable. Note that all of these may change based on the application©s current state. During initialization of the application, this information is conveyed from the application to Milan via \State-based Variable Requirements" and \Sensor QoS" graphs. Examples of these graphs are shown in figures 78 and 79, respectively. Figure 78, an abstract State-based Variable Requirements Graph, shows the required QoS for each variable of interest based on the current state of the system and the variables of interest to the application, where these states are based on the application©s analysis of previously received data. For a particular state (a combination of system state (level A) and variable state (level B)), the State-based Variable Requirements Graph defines the required QoS for each relevant variable. Because variables (level C) can be named in multiple variable states (level B), Milan must extract the maximum QoS for each selected variable to satisfy the requirements for all variable states. Figure 79 shows the State-based Variable Requirements Graph for the personal health monitor. This application has two states - a system state that includes the patient©s overall stress level, as well as multiple states for each variable that can be monitored.

The State-based Variable Requirements Graph specifies to Milan the application©s minimum acceptable QoS for each variable (e.g., blood pressure, respiratory rate, etc.) based on the current state of the patient. For example, the figure shows that when a patient is in a medium stress state and the blood pressure is low, the blood oxygen level must be monitored with a quality level of :7 and the blood pressure must be monitored with a quality level of :8. For a given application, the QoS for each variable can be satisfied using data from one or more sensors. The application specifies this information to Milan through the Sensor QoS Graph, figure 76. When multiple sensors are combined to provide a certain quality level to the variable, we refer to this as a single “virtual sensor". Figure 77 shows the Sensor QoS Graph for the personal health monitor. This graph illustrates the important variables to monitor when determining a patient©s condition and indicates the sensors that can provide at least some quality to the measurement of these variables. Each line between a sensor (or virtual sensor) and a variable is labeled with the quality that the sensor (or virtual sensor) can provide to the measurement of that variable. For example, using data from a blood pressure sensor, the heart rate can be determined with a .7 quality level, but combining this with data from a blood flow sensor increases the quality level to 1:0. Given the information from these graphs as well as the current application state, Milan can determine which sets of sensors satisfy all of the application©s QoS requirements for each variable. These sets of sensors define the application feasible set FA, where each element in FA is a set of sensors that provides QoS greater than or equal to the application-specified minimum acceptable QoS for each specified variable. For example, in the personal health monitor, for a patient in medium stress with a high heart rate, normal respiratory rate, and low blood pressure, the application feasible sets in FA that Milan should choose to meet the specified application QoS are shown in Table 7. Milan must choose which element of FA should be provided to the application. This decision depends on network-level information.

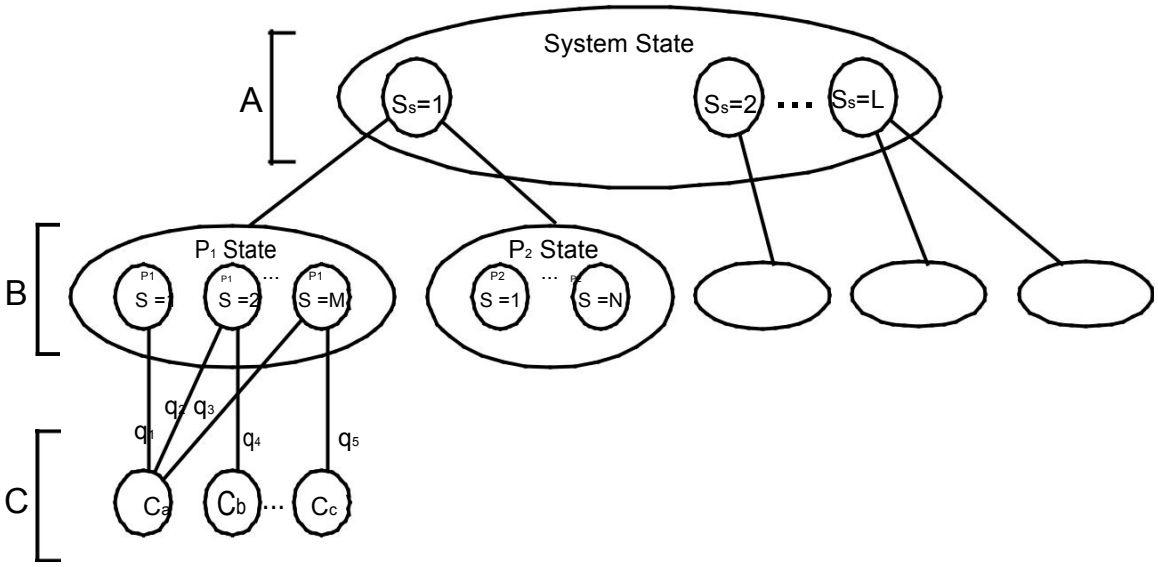


Figure 78: Generic Performance graph

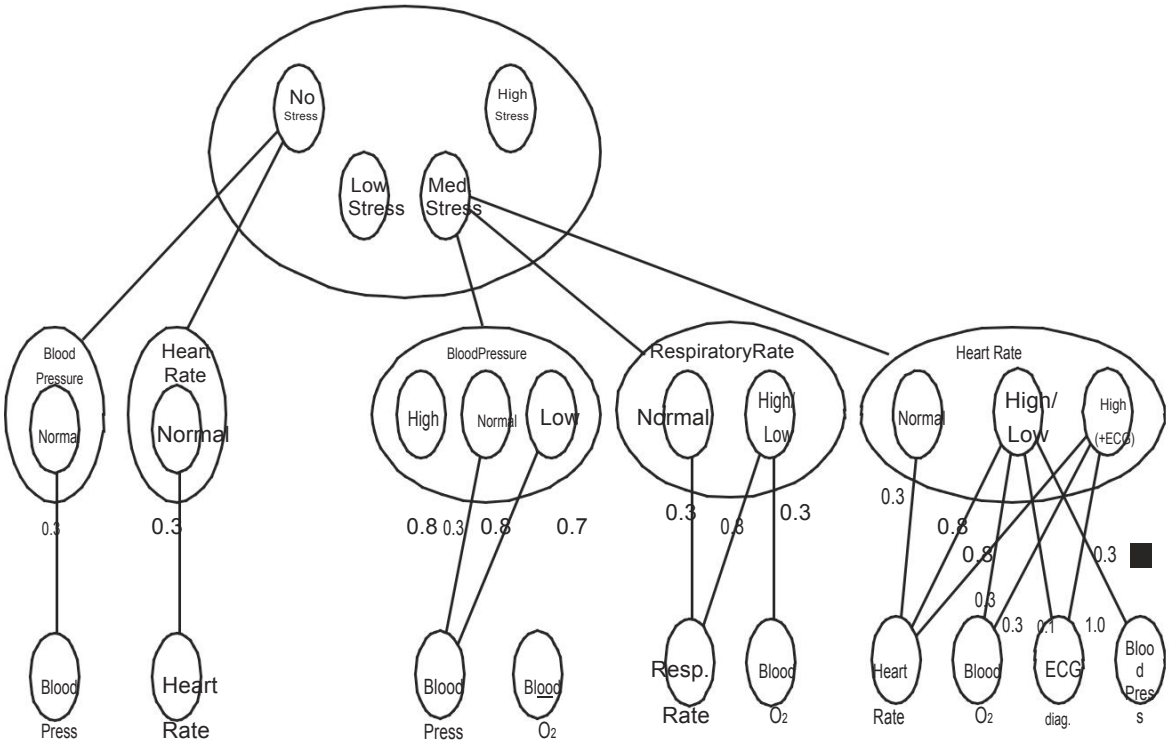


Figure 79: Heart Monitor application Performance graph

Table 7: Specified application QoS

Set	Sensors
1	**Blood Flow and Respiratory Rate**
2	**Blood Flow and ECG 3 leads**
3	**Pulse Oxymeter, Blood Pressure, ECG (1 lead), Respiratory Rate**
4	**Pulse Oxymeter, Blood Pressure, ECG (3 lead),**
5	**Oxygen Measurement, Blood Pressure, ECG (1 lead), Respiratory Rate**
6	**Oxygen Measurement, Blood Pressure, ECG (3 lead)**

6.5.2 Tradeoffs

Among the elements in F, Milan chooses an element fi that represents the best performance/cost tradeoff. How should “best" be defined? This depends on the application. Milan framework supports any method of deciding how to choose an element of F. In most sensor network applications, we want to allow the application to last as long as possible using the limited energy of each of the sensors. Simple approaches to choosing sensor sets may yield the set fi that consumes the least power or that will run for the maximum lifetime before the first sensor dies. However, if we want to ensure that the application can run at the required QoS level as long as possible, we should instead optimize the total lifetime by intelligently choosing how long to use each feasible sensor set [83]. In some cases, there are multiple ways to schedule sensors so that the same total network lifetime is achieved. In these cases, we may want to maximize the average quality of the sensor sets over time. For some applications, the goal may be to maximize some combination of lifetime and quality. Milan is flexible enough to incorporate any of these or other optimization criteria. It is performed the tradeoff computation occurring in the core Milan component. After the computation is complete and the first set of sensors is chosen, the Milan core informs the plug-in of the selection, and the plug-in configures the network accordingly, using information about the role each sensor should play.

6.6. Discussion

Fault tolerance and resilience: The middleware application relationship is a powerful tool to provide fault tolerance and resilience to fails and adaptation of the system to new states of the application. If the application provides to middleware more than one possible way to provide one service, the system can

choose which one is better. Furthermore, if one way fails, the system can provide the same service by the other set of sensors.

Ubiquitous computing aspect: The approach that provides fault tolerance and resilience to the system allows to build a ubiquitous computing environment. The application specifies to the system the way it accepts a graceful degradation along the decrement in available resources. This concept allows the application to run until the QoS can be provided by the entire system. It is part of the application designer job to specify an application that is compatible with graceful degradation. We have specified a language represented by the different graphs to represent the resilience and graceful degradation of the application.

6.7. Conclusions

Current research trends suggest the power of middleware to ease the application development task in complex environments. While conventional middleware operates above the networking layer, for sensor network applications that rely on multiple and varying sensors, it is not a viable approach to manage the network completely independently of the needs of the application. We have argued that the needs of the application should be integrated with the management of the network into a single, unified middleware system. Through this tight coupling, the middleware can trade application performance for network cost, while still retaining the separation between the policy specifying how to react to a dynamic environment (obtained from the application) and the mechanisms to implement the policy (performed in the middleware). We have shown that Milan, a sensor network middleware that we are developing to meet these goals, can aid the development of sensor network applications.

Conclusion

We have presented a general data fusion architecture described in a formal language of object representations (UML) that tries to represent different scenarios, specifications and features of a general data fusion system. It also allows a dynamic modification of the system according to different states of the environment or of the system.

We have also described general body-worn sensor networks built in a network of micro sensors, agents and actuators. The system was designed considering and optimizing power management and wireless network problems to achieve the resilience and ubiquitous computing goals. Furthermore, some specific aspects related to the health and body-worn sensor network application were considered. We have also introduced the concept of a resilience agent. This type of agent was more complex than a fault tolerant agent, and it included the fault tolerant agent's functionalities added adaptability to new circumstances without failures. Moreover, it considered clinical aspects to decide about power consumption and network optimization.

The Personal Heart Rate Monitor is from the class of network-based mobile dynamic systems powered by battery, where an application should adapt itself to different configurations of the system (data sources moving in and moving out), different states of the environment, and consider power and bandwidth constraints. We propose a solution for the problem of developing an application framework to manage data from different types of sensors to perform a Heart Rate Monitoring application in a Ubiquitous Computing environment. In this paper, we have focused on the application's framework (data fusion and decision modules) while also considering the necessary middleware and network facilities to ensure resilience to changes in available resources and in the environment at state. As a consequence, the solution for these problems may solve the same problems in the class of related systems. Controlling Robots based on the environment and home security systems are two examples of this class of systems to which our framework should also be applicable.

The PHRM is a new system that takes into consideration that it is very difficult to analyze one body variable independently. In the case of the heart rate, without knowing the user age, underlying diseases, level of body activity, sleeping state and alert, it is very difficult to conclude whether the measured heart rate is normal or abnormal. Nowadays, available systems do not consider this wild view of physiological monitoring and interpretation. Another aspect that makes the developed system new is the fact that it takes into account aspects such as fault tolerance and ubiquitous computing determined by not only the hardware, but also by the necessity of the application (Application's Quality of Service). In this aspect, the presence of a middleware can help to manage the hardware point of view together with the application point of view, optimizing this relationship and the system life time.

Based on these aspects, the PHRM is the first heart rate monitoring system that considers multiple correlations with the user's health variables to help the user in the monitoring of his health. To achieve this goal, the general data fusion architecture applied showed that it is a reasonable approach to provide dynamic management of data and

variables from different sources. Besides the developed system is fault tolerant to failures and compatible with ubiquitous availability.

To achieve the goal of performing data fusion in an optimal node in distributed sensor networks, we have implemented different algorithms already described in the literature (shortest path) and greedy algorithm. The other approximate algorithms are adaptations of the greedy algorithm. The suboptimal´s algorithms (Hervaldo's algorithm and optimal node algorithm) showed to be the algorithms with better performance and acceptable cost. The results obtained from the computation cost simulations showed that the approximate solutions are less expensive than the optimal solution. The comparison of the computation cost among the aproximate solutions showed that the Greedy, modified Greedy and Hervaldo's algorithms have the same computation cost. The computation cost of the optimal node is only 20% (1/5) of the other approximate solutions. As a conclusion, the optimal node solution has the lowest computation cost. As a result of the simulations considering the communication cost and the computation cost, the best alternatives to the optimal solution are the approximate Hervaldo's algorithms and the optimal node solution.

Multiple data fusion paths can have positive or negative results. If a node is a sensing unit and data fusion node simultaneously, its energy will drain earlier and the system´s lifetime will be shorter. So, the worst case scenario is the situation where the same nodes act as sensing units and data fusion or destination node. The best data fusion multiple paths solutions are characterized by the use of different nodes to act as sensing units, data fusion nodes and destination nodes. If the system uses more nodes at approximately the same communication cost, the system's lifetime will be longer. Consequently, the sensor network application designer should balance minimum cost and maximum lifetime. The system´s goal is to achieve the minimum communication and data fusion cost, and to maximize the system's lifetime.

Network density should be considered in the design of sensor network applications. A network with a hundred nodes and ten sensors as source of data will use almost all nodes to perform 10 distributed data fusion paths to each combination of sensors and destination node. This result shows that the number of network nodes is a limiting factor to the increase of the system's lifetime. Another aspect that should be considered is the communication data fusion cost tradeoff. If the data fusion computation cost is higher than the communication cost, in general it is better to increase the number of paths and data fusion nodes. On the other hand, it is better to use the fewest number of paths.

The use of combination multiple paths and multiple nodes increases the system's lifetime. This increase is greater in the algorithms that vary the nodes in different paths. The 10 shortest paths in the shortest path algorithm use almost the same nodes. In the other hand, the approximate solution based on the best node not considering the destination node achieves the best system's lifetime because it creates more paths with different nodes. So, considering the system's lifetime, the shortest path is not the best solution to achieve the optimal data fusion nodes to solve the data fusion expression. Furthermore, network density is proportional to system's lifetime if the number of sensing units and DF nodes are constant. A network with 50 nodes will have a shorter system's lifetime than a network with 500 nodes.

In all algorithms tested the network based on multiple paths and multiple DF nodes is much better than the options of multiple paths, average multiple paths, shortest path (better communication cost) and average multiple paths and multiple DF nodes. These differences are greater when we increase the communication data fusion cost ratio. The comparison among the algorithms showed that the Approximate Optimal DF Node without sensor is slightly better than the Optimal DF node and much better than Hervaldo's algorithm.

The hypothesis of using long paths can be better than the one of using shortest paths this was evaluated by testing the performance of the Optimal Node algorithm in the best 100 paths divided in ranges of ten. The results showed that it is not favorable to use longest paths to increase system's lifetime in low density networks. It should be tested in high density networks.

Current research trends suggest the power of middleware to ease the application development task in complex environments. While conventional middleware operates above the networking layer, for sensor network applications that rely on multiple and varying sensors, it is not a viable approach to manage the network completely regardless of the needs of the application. We have discussed that the needs of the application should be integrated with the management of the network into a single, unified middleware system. Through this tight coupling, the middleware can trade application performance for network cost, while still retaining the separation between the policy specifying how to react to a dynamic environment (obtained from the application) and the mechanisms to implement the policy (performed in the middleware). We have shown that Milan, a sensor network middleware that we are developing to meet these goals, can aid the development of sensor network applications.

The general data fusion architecture proposed can be represented at middleware level in different ways. We have shown a solution represented by graph theory that seems to be a reasonable approach to guarantee resilience of the system's functions to achieve the ubiquitous computing goal. Our model also considers the possibility of integration of the different aspects from different applications running at the same time.

The data fusion architecture, data fusion techniques such as Kalman Filter, power state machine, and middleware application relationship are powerful tools to provide fault tolerance and resilience to failure and adaptation of the system to new states of the application. If the application provides to middleware in more than one possible way (set of sensors) to provide one service, the system can choose which one is better. Furthermore, if one way fails, the system can provide the same service through the other set of sensors.

The approach that provides fault tolerance and resilience to the system allows to build a ubiquitous computing environment. The application specifies to the system the way it accepts a graceful degradation along the decrease in available resources. This concept allows the application to run until the application´s QoS can be provided by the entire system. It is part of the application designer job to specify an application that is compatible with graceful degradation. We have specified a language represented by different graphs to demonstrate the resilience and graceful degradation of the application.

Contributions (scientific production related to the thesis)

Patent:

1. Coelho Jr, Claudionor J. N.; Andrade, Luiz Cláudio Gil; Conway, Júlio C. D. ; Carvalho, Hervaldo Sampaio; Fernandes, Antônio O.; Silva Jr, Diógenes C. da; Pinho, Antônio L. Monitor Biológico Multiparamétrico Usável, 2000. Patente: Modelo Industrial n. PI0001075-8, "Monitor Biológico Multiparamétrrico Usável", 17 de abril de 2000 Brasil (depósito).

Journal Articles:

1. Wendi B. Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho, Mark A. Perillo. Middleware to Support Sensor Networks Applications. IEEE Network 18(1): 6-14, 2004.

International Conferences:

1. H.S. Carvalho, Wendi B. Heinzelman, Amy L. Murphy, Coelho Jr, Claudionor J. N. A General Data Fusion Architecture. IEEE Information Fusion 2003. Proceedings Of the 6th International Conference of Information Fusion, Australia, July 8-11, 2003, volume 2, pages 1465-1472.
2. H.S. Carvalho, Amy L. Murphy,Wendi B. Heinzelman, Coelho Jr, Claudionor J. N. Network Based Distributed Systems Middleware. Proceedings of the International Conference on IEEE/ACM Middleware 2003 (1st International Workshop on Middleware for Pervasive and Ad-Hoc Computing), Rio de Janeiro, June, 2003, pg 22-30.
3. H.S. Carvalho, Coelho Jr, Claudionor J. N., Wendi B. Heinzelman, Amy L. Murphy. Body-Worn Sensor Networks Applied for Health Monitoring. Proceedings of the IEEE 30th International Conference on Computers in Cardiology, Greece, September, 2003.
4. Ana Luiza de Almeida Pereira Zuquim, Antônio Alfredo F. Loureiro, Claudionor J. N. Coelho Jr, Marcos Augusto M. Vieira, Luiz Felipe Menezes, Alex Borges Vieira, Antonio O. Fernandes, Diógenes Cecílio da Silva Jr, José M. da Mata, José Augusto Nacif, Hervaldo S. Carvalho. Efficient Power Management in Real-Time Embedded Systems. ETFA 2003, 16-19 September 2003, Lisbon, Portugal.
5. Hervaldo Sampaio Carvalho, Berthier Ribeiro-Neto, Claudionor J. N. Coelho Jr. Evidence Based Cardiovascular Information Retrieval. 14th World Congress of Cardiology, Austrália, 05 to 09 May, 2002. (session Computers in Cardiology). Abstract Published in the Journal of the American College of Cardiology supplement 2002.
6. Julio C. D. Conway, Claudionor José Nunes Coelho Jr, Diógenes C. da Siolva, Antônio O. Fernandes, Luis C. G. Andrade, Hervaldo S. Carvalho. Wearable

Computer as a Multiparametric Monitor for Physiological Signals. IEEE BIBE 2000, 236-242.

7. Claudionor José Nunes Coelho Jr, Antônio O. Fernandes, Julio C. D. Conway, Fabio L. Correa Jr Hervaldo S. Carvalho, Jose M. Mata. A Biomedical Wearable Device for Remote Monitoring of Physiological Signals. SPG 2003.
8. Hervaldo S. Carvalho, Claudionor José Nunes Coelho Jr, Antonio Otávio Fernandes, José Augusto Nacif. FPGA Based Electrocardiographic Diagnosis. APCMBE, Singapore, 2002 (paper identification: conf1a-345).
9. Claudionor José Nunes Coelho Jr, Hervaldo S. Carvalho, Julio C. D. Conway, Diógenes C. da Siolva, Antônio O. Fernandes, Jose M. da Mata. Mobile Monitoring of Physiological Signals Technologies and Applications. Punta Arenas, Chile, 2001.
10. Hervaldo S. Carvalho, Claudionor José Nunes Coelho Jr, Diógenes C. da Silva Jr, Antonio Otávio Fernandes. A Multiparametric Cardiac Controller System. Proceedings of the II Latin American Congress of Artificial Organs and Biomaterials, Belo Horizonte – Brazil, 5-8 December, 2001 (paper identification: BRMG051). Best paper of the conference.

Brazilian Conferences:

1. R. S. Ortis, H. S. Carvalho, A. F. Rocha, Coelho Jr., C. J. N., Nascimento, F. A.O. Monitorização de Sinais Biomédicos em Assistentes Pessoais Digitais. VIII Congresso Brasileiro de Informática em Saúde. Ribeirão Preto, SP, Brazil, Novembro de 2004 (aceppted paper for oral presentation). Paper will be published in the conference proceedings.
2. Hervaldo S. Carvalho, Claudionor J. N. Coelho Jr, Wendi B. Heinzelman. Gerenciamento de Informações Médicas do Paciente. Proceedings of the VIII Congresso Brasileiro de Informática em Saúde. Natal, RN, Brazil, Novembro de 2002 (Paper identification: D1S 1440).
3. Hervaldo S. Carvalho , Julio C. D. Conway, Luis C. G. Andrade, Diógenes C. da Siolva, Antônio O. Fernandes, Claudionor José Nunes Coelho Jr. Computadores Usáveis Dedicados a Monitoração Multiparamétrica de Sinais Biológicos. Proceedings of the VI Congresso Brasileiro de Informática em Saúde. São Paulo, SP, Brazil, 14-18 de Outubro de 2000 (Second best Paper conference).

International Research (Visiting Faculty at The Center For Future Health)

Institution: Center For Future Heath, University of Rochester Medical Center, University of Rochester, Rochester, NY, USA

Advisor: Wendi B. Heinzelman. Dept of Electrical and Computer Engineer, University of Rochester, Rochester, NY, USA

Co-advisor: Amy Murphy. Dept of Computer Science, University of Rochester, Rochester, NY, USA.

References

[1] Wald L., A European proposal for terms of reference in data fusion. *International Archives of Photogrammetry and Remote Sensing*, Vol. XXXII, Part 7, 651-654, 1998, or Wald L., Some terms of reference in data fusion. *IEEE Transactions on Geosciences and Remote Sensing*, 37, 3, 1190-1193, 1999.

[2] U.S. Department of defense, data Fusion Sub panel of the Joint Directors of Laboratories, Technical Panel for C3, "Data Fusion lexicon,"1991.

[3] Wald L., The present achievements of the EARSeL - SIG "Data Fusion". In Proceedings of the EARSeL Symposium, held in Dresden, Germany, June 2000.

[4] Ankur Jaim, Edward Y. Chang. Adaptive Sampling for Sensor Networks. ww.cs.ucsb.edu/~ankurj/directory/jcdmsn04.pdf at 22/01/2005.

[5] A. Singhal and C. Brown. *Dynamic Bayes net approach to multimodal sensor fusion*. Proceedings of the SPIE - The International Society for Optical Engineering, 3209:2--10, October, 1997.

[6] Chen T. M. & Luo, R.C. Multilevel Multiagent Based Team Decision Fusion for Autonomous Tracking System. Machine Intelligence & Robotic Control, 1(2), 63-69 (1999).

[7] M. M. Kokar, J. A. Tomasik and J. Weyman. A Formal Approach to Information Fusion. Proceedings of the Second International Conference on Information Fusion (Fusion©99*),* Vol.I, pp.133-140, July 1999.

[8] Myers J.W., Laskey K.B., DeJong, K.A. Learning Bayesian Networks from Incomplete Data using Evolutionary Algorithms. In Proceedings of The Genetic and Evolutionary Computation Conference. 1999. Orlando, Fl.

[9] http://www.data-fusion.org/ at 22/01/2005.

[10] Durrant, H.F. W. Integration. Coordination and Control Multi-sensor Robot Systems. Kluwer Academic Publishers, 1988.

[11] Clement, V., Giraudon, G., Houzelle, S., Sadakly, F., 1993. Interpretation of Remotely Sensed Images in a Context of Multisensor Fusion Using a Multispecialist Architecture. IEEE Trans. On Geoscience and remote Sensing, vol. 31, No 4, pp. 779-791.

[12] Matsuyama, T., Hwang, V. S.-S., 1990. Sigma: A Knowledge-Based Aerial Image Understanding System, Plenum Press, New York, 277p.

[13] McKeown, D., Wilson, A., McDermott, J., 1985. Rule Based Interpretation of Aerial Imagery, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 22(2), pp. 231-243.

[14] Growe, S. Knowledge Based Interpretation of Multisensor and Multitemporal Remote Sensing Images. http://www.data-fusion.org/ps/sig/meeting/Spain99ps/growe.pdf at 01/22/2005

[15] Dailey, D.J., Ham, P., Lin, P-J. It's Data Fusion. Technical Report. Washington State Department of Transportation (USA), September, 1996.

[16] Laskey, K. B., Mahoney, S. M. Knowledge and Data Fusion in Probabilistic Networks. PhD thesis. http://ite.gmu.edu/~klaskey/papers/KDFML_Laskey_Mahoney.pdf at 01/22/2005.

[17] Carvalho, H.S. Computerized System based on Fuzzy Logic to Evaluate Autonomic Nervous System Based on Multiple Tests. Master Degree Thesis. University of Brasilia, Brazil, 1996.

[18] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks,©*Wireless Networks*, Vol. 8, 2002, pp. 169-185.

[19] L. Schwiebert, S.K.S. Gupta, J. Weinmann. Research Challenges in Wireless Networks of Biomedical Sensors. *ACM Sigmobile* 2001.

[20] H. L. Younes. Current Tools for Assisting Intelligent Agents in Real-Time Decision Making. *Master Degree* Thesis. Royal Institute of Technology, School of Electrical Engineering and Information Technology1998.

[21] U. Chajewska et al. Utility Elicitation as a Classification Problem. http://smi-web.stanford.edu/projects/panda/ at 01/22/2005.

[22] O. Buffet, A. Dutech, F. Charpillet. Incremental Reinforcement Learning for Designing Multi-Agent Systems. *Agents'01 ACM 2001.*

[23] P. Sarkar. A Brief History of Cellular Automata. *ACM Computing Surveys. Vol 32, No 1:83-107, march 2000.*

[24] K. Inoue, S. E. Chick, C. Chen. An Empirical Evaluation of Several Methods to Select the Best System. *ACM Transactions on Modeling and Computer Simulation*, vol. 9, no 4: 381-407, october 1999.

[25] G. J. Pottie and W. J. Kaiser. Wireless Integrated Network Sensors. *Communications of the ACM*, vol. 43 No 5: 51-58, May 2000.

[26] G. Borriello and R. Want. Embedded Computation Meets the World Wide Web. *Communications of the ACM*, vol. 43 No 5: 59-66, May 2000.

[27] R. R. Kampfner. Dynamics and Information processing in adaptive systems. *BioSystems* 46 (1998) 153-162.

[28] Z. Duan, M. Holcombe, A. Bell. A logic for biological systems. *BioSystems* 55 (2000) 91-105.

[29] M. He and H. Leung. An Agent Bidding Strategy Based on Fuzzy Logic in a Continuous Double Auction. *Agents'01*, ACM 2001.

[30] W. B. Heinzelman, A. P. Chandrakasan, H. Balakrihman. An Application-Specific Protocol Architecture for Wireless Micro sensor Networks.

[31] L. Chen, K. Bechkoun and G. Clapworthy. A Logical Approach to High-Level Agent Control. *Agents'01* ACM 2001.

[32] B. B. Werger and M. J. Mataric. From Insect to Internet: Situated Control for Networked Robot Teams.

[33] MEMS and Micromachining in the new Millennium: Biomedical Applications AidSurgery,Diagnosis,DrugDeliveryandMore. http://www.manufacturingcenter.com/dfx/archives/0899/899mem.asp at 01/22/2005.

[34] R. Jain. *The Art of Computer Systems Performance analysis, techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley & Sons, INC, 1991.

[35] M.Bhardwaj, T. Garnett and A. P. Chandrakasan. Upper Bounds on the Lifetime of Sensor Networks. *http://www-mtl.mit.edu/research/icsystems/uamps at 01/22/2005.*

[36] Report 4 of the Council on Scientific Affairs. Use of Wireless Radio-Frequency Devices in Hospital. http://www.ama-assn.org/ama/pub/article/2036-2918.html at 01/22/2005

[37] F. Gemperle, C. Kasabach, J. Stivoric, M. Bauer and R. Martin. Design of Wearability. Second International Symposium on Wearable Computers, ISWC 98, Pittsburgh, PA, 1998.

[38] W. Shen, Y. Lu and P. Will. Hormone-Based Control for Self-Reconfigurable Robots. Agents 2000.

[39] L. F. Lago-Fernández, M. ª Sánchez-Montanés and S. López-Buedo. A Biologically Inspired Autonomous Robot that Learns Approach-Avoidance Behaviors. Agents 2000.

[40] A. Billard and M. J. Mataric. *A biologically inspired robotic model for learning by imitation.* Agents 2000.

[41] Julio C. D. Conway, Claudionor José Nunes Coelho Jr, Diógenes C. da Siolva, Antônio O. Fernandes, Luis C. G. Andrade, Hervaldo S. Carvalho. Wearable Computer as a Multiparametric Monitor for Physiological Signals. IEEE BIBE 2000, 236-242.

[42] Hu, Y.H.; Palreddy, S.; Tompkins, W.J. A patient-adaptable ECG beat classifier using mixture of experts approach. IEEE Trans Biomed Eng 1997 Sep; 44(9):891-900.

[43] H.S. Carvalho, Amy L. Murphy,Wendi B. Heinzelman, Coelho Jr, Claudionor J. N. Network Based Distributed Systems Middleware. Proceedings of the International Conference on IEEE/ACM Middleware 2003 (1st International Workshop on Middleware for Pervasive and Ad-Hoc Computing), Rio de Janeiro, June, 2003, pg 22-30.

[44] Wendi B. Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho, Mark A. Perillo. Middleware to Support Sensor network Applications. IEEE Network 18(1): 6-14, 2004.

[45] HS. Carvalho, W. Heinzelman, A. Murphy and C. Coelho, "A General Data Fusion Architecture," Proceedings of the 6th International Conference on Information Fusion (Fusion 2003), July 2003.

[46] H.S. Carvalho, Coelho Jr, Claudionor J. N., Wendi B. Heinzelman, Amy L. Murphy. Body-Worn Sensor Networks Applied for Health Monitoring. Proceedings of the IEEE 30th International Conference on Computers in Cardiology, Greece, September, 2003.

[47] Hervaldo Sampaio Carvalho, Berthier Ribeiro-Neto, Claudionor J. N. Coelho Jr. Evidence Based Cardiovascular Information Retrieval. 14th World Congress of Cardiology, Australia, 05 to 09 May, 2002. (session Computers in Cardiology). Abstract Published in the Journal of the American College of Cardiology supplement 2002.

[48] Claudionor José Nunes Coelho Jr, Antônio O. Fernandes, Julio C. D. Conway, Fabio L. Correa Jr Hervaldo S. Carvalho, Jose M. Mata. A Biomedical Wearable Device for Remote Monitoring of Physiological Signals. SPG 2003.

[49] Claudionor José Nunes Coelho Jr, Hervaldo S. Carvalho, Julio C. D. Conway, Diógenes C. da Siolva, Antônio O. Fernandes, Jose M. da Mata. Mobile Monitoring of Physiological Signals Technologies and Applications. Punta Arenas, Chile, 2001.

[50] Hervaldo S. Carvalho, Claudionor José Nunes Coelho Jr, Diógenes C. da Silva Jr, Antonio Otávio Fernandes. A Multiparametric Cardiac Controller System. Proceedings of the II Latin American Congress of Artificial Organs and Biomaterials, Belo Horizonte – Brazil, 5-8 December, 2001 (paper identification: BRMG051).

[51] Hervaldo S. Carvalho, Claudionor J. N. Coelho Jr, Wendi B. Heinzelman. Gerenciamento de Informações Médicas do Paciente. Proceedings of the VIII Congresso Brasileiro de Informática em Saúde. Natal, RN, Brazil, Novembro de 2002 (Paper identification: D1S 1440).

[52] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-E_cient Communication Protocol for Wireless Micro sensor Networks. IEEE Transactions on Wireless Communication, 1(4):660{670, Oct. 2002.

[53] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Di_usion: A Scalable and Robust Communication Paradigm for Sensor Networks. Proceedings of ACM Mobicom ©00), Aug. 2000.

[54] S. Singh and C. Raghavendra. PAMAS: Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. ACM Computer Communication Review, 28(3):5{26, July 1998.

[55] Y. Wei, J. Heidemann, and D. Estrin. An Energy-E_cient MAC Protocol for Wireless Sensor Networks. In Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), June 2002.

[56] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring Sensor Network Topologies. In Twenty-First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), June 2002.

[57] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-E_cient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. ACM Wireless Networks, 8(5), September 2002.

[58] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing Sensor Networks in the Energy-Latency-Density Design Space. IEEE Transactions on Mobile Computing, 1(1):70{80, January 2002.

[59] R. Ramanathan and R. Rosales-Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000), pages 404{413, March 2000.

[60] E. Lloyd, R. Liu, M. Marathe, R. Ramanathan, and S. Ravi. Algorithmic Aspects of Topology Control Problems for Ad Hoc Networks. In Processings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002), pages 123{134, June 2002.

[61] V. Rodoplu and T. Meng. Minimum Energy Mobile Wireless Networks. IEEE Journal on Selected Areas in Communications, 17(8):1333{1344, August 1999.

[62] D. Braginsky and D. Estrin. Rumor Routing Algorithm for Sensor Networks. In Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications, 2002.

[63] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In Proc. of the 5th Annual ACM/IEEE Int. Conference on Mobile Computing and Networking (MobiCom ©99), pages 174{185, August 1999.

[64] Object Management Group. The Common Object Request Broker: Architecture and Specification Revision 2.2. 492 Old Connecticut Path, Framingham, MA 01701, USA, 1998.

[65] K. Edwards. Core JINI. Prentice Hall, 1999.

[66] A.L. Murphy, G.P. Picco, and G.-C. Roman. Lime: A Middleware for Physical and Logical Mobility. In Proceedings of the 21st International Conference on Distributed Computing Systems, pages 524{533, April 2001.

[67] D. Gelernter. Generative Communication in Linda. ACM Transactions on Programming Languages and Systems, 7(1):80{112, January 1985.

[68] N. Davies, S. Wade, A. Friday, and G. Blair. Limbo: A tuple space based platform for adaptive mobile applications. In Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms (ICODP/ICDP ©97), Toronto, Canada, May 1997.

[69] O. Holder, I. Ben-Shaul, and H. Gazit. System Support for Dynamic Layout of Distributed Applications. In Proceedings of the 19th International Conference on Distributed Conputing, pages 403{411, 1999.

[70] A.T. Campbell. Mobiware: QOS aware middleware for mobile multimedia communications. In Proceedings of the 7th IFIP International Conference on High Performance Networking (HPN), White Plains, New York, USA, April 1997.

[71] B. D. Noble and M. Satyanarayanan. Experience with Adaptive Mobile Applications in Odyssey. Mobile Networks and Applications, 4(4):245{254, 1999.

[72] J. Flinn, D. Narayanan, and M. Satyanarayanan. Self-tuned remote execution for pervasive computing. In Proceedings of the Eighth IEEE HotOs Conference, Elmau/Oberbayern, Germany, May 2001.

[73] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Toward distractionfree pervasive computing. IEEE Pervasive Computing, April {June 2002.

[74] K. Nahrstedt, D. Xu, D. Wichadakul, and B. Li. QoS-Aware Middleware for Ubiquitous and Heterogeneous Environments. IEEE Communications Magazine, 39(11), 2001.

[75] C. Jaikaeo, C. Srisathapornphat, and C.-C. Shen. Querying and Tasking in Sensor Networks. In SPIE©s 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V), Orlando, Florida, April 24{28 2000.

[76] Q. Han and N. Venkatasubramanian. Autosec: An integrated middleware framework for dynamic service brokering. IEEE Distributed Systems Online, 2(7), 2001.

[77] S. Li, S. Son, and J. Stankovic. Event detection services using data service middleware in distributed sensor networks. In Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks, April 2003.

[78] T. Liu and M. Martonosi. Impala: A middleware system for managing autonomic, parallel sensor systems. In ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP©03), June 2003.

[79] J.C.D. Conway, C.J.N. Coelho, D.C. da Silva, A.O. Fernandes, L.C.G. Andrade, and H.S. Carvalho. Wearable Computer as a Multi-parametric Monitor for Physiological Signals. In Proceedings of the IEEE International Symposium on Bioinformatics and Bioengineering (BIBE), pages 236{242, 2000.

[80] S. Avancha, A. Joshi, and T. Finin. Enhanced Service Discovery in Bluetooth. IEEE Computer, 35(6):96{99, June 2002.

[81] Service Location Protocol (SLP). http://www.ietf.org/html.charters/svrloc-charter.html at 01/22/2005.

[82] S. L. Dockstader and A. M. Tekalp. Multiple Camera Tracking of Interacting and Occluded Human Motion. Proceedings of the IEEE, 89(10):1441{1455, Oct. 2001.

[83] Mark Perillo and Wendi Heinzelman. Simple Approaches for Providing Application QoS Through Intelligent Sensor Management. Elsevier Ad Hoc Networks Journal, 1(2-3):235{246, 2003).

[84] W. Heinzelman, A. Murphy, H. Carvalho and M. Perillo, "Middleware to Support Sensor Network Applications," IEEE Network Magazine Special Issue, Jan. 2004.

[85] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. IEEE Personal Communication, 7:10{15, October 2000.

[86] Data Fusion in Decentralised Sensing Networks. Hugh Durrant-Whyte and Mike Stevens. www.arofe.army.mil/Conferences/ Intelligent_Abstract/5DDurrant-Whyte.pdf at 01/22/2005.

[87] H. Qi, X. Wang, S. S. Iyengar, K. Chakrabarty, "Multisensor data fusion in distributed sensor networks using mobile agents," *Information Fusion*, TuC2-11-16, Canada, August, 2001.

[88] D. N. Jayasimha, S. S. Iyengar, and R. L. Kashyap. Information integration and synchronization in distributed sensor networks. *IEEE Trans. Syst., Man Cybern.*, SMC-21(21):1032–1043, Sept./Oct. 1991.

[89] A. Knoll and J. Meinkoehn. Data fusion using large multi-agent networks: an analysis of network structure and performance. In *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 113– 120, Las Vegas, NV, Oct. 2-5 1994. IEEE.

[90] L. Prasad, S. S. Iyengar, R. L. Kashyap, and R. N Madan. Functional characterization of sensor integration in distributed sensor networks. IEEE Trans. Syst., Man, Cybern., SMC-21, Sept./Oct. 1991.

[91] H. Qi, S. S. Iyengar, and K. Chakrabarty. Multiresolution data integration using mobile agents in distributed sensor networks. IEEE Transactions on SMC: C, 2000.

[92] Improved data fusion through intelligent sensor management. .I.Smith, C.R.Angell, M.L.Hernandez,W.J.Oxford. *Proceedings of SPIE* Vol. #5096, April 2003.

[93] J. Watts, S. Taylor, "A Practical Approach to Dynamic Load Balancing," IEEE Transactions on Parallel and Distributed System, Vol. 9, pp. 235-248, 1998.

[94] K.C. Chang and Y. Bar-Shalom, " Joint Probabilistic Data Association in Distributed Sensor Networks," IEEE Trans. Autom. Contr. AC-31, pp. 889-897, 1986.

[95] Andy Franz, Radek Mista, David Bakken, Curtis Dyreson, Murali Medidi2. Mr. Fusion1: A Programmable Data Fusion Middleware Subsystem with a Tunable Statistical Profiling Service. Proceedings of the International Conference on

Dependable Systems and Networks (DSN-2002) IEEE/IFIP, June 23-26, 2002, Washington, DC.

[96] Hall, D. and Llinas, J. “Handbook of Multidata Sensor Fusion”, CRC Press, 2001.

[97] Walz, E. and Llinas, J. “Multisensor Data Fusion”, Artech House, Boston, 1990.

[98] Bing Ma. Approaches for Multisensors Data Fusion. Doctor of Philosophy dissertation. Electrical Engineering: Systems, The University of Michigan, 2001.

[99] Mohin Ahmeda, Son Daoa, Gregory Pottieb, Srikanth Krishnamurthyc, and Randy Katzd. On the Optimal Selection of Nodes to Perform Data Fusion in Wireless Sensor Networks. Proceedings of SPIE -- Volume 4396 Battlespace Digitization and Network-Centric Warfare, Raja Suresh, Editor, August 2001, pp. 53-64.

[100] Konstantinos Kalpakis, Koustuv Dasgupta, and Parag Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks". In the Proceedings of the 2002 IEEE International Conference on Networking (ICN©02), Atlanta, Georgia, August 26-29, 2002. pp. 685-696.

[101] Maurice Chu, Horst Haussecker, and Feng Zhao. Scalable Information-Driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Networks. Xerox Palo Alto Research Center Technical Report P2001-10113, May 2001.

[102] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks,©IEEE Transactions on Wireless Communications, Vol. 1, No. 4, October 2002, pp. 660-670.