DATA FUSION IN WIRELESS SENSORS NETWORK

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

COMPUTER SCIENCE & ENGINEERING

Under the Supervision of

DR. YASHWANT SINGH

By

ANKUR NADDA (091028)

SAHIL GUPTA (091263)

to

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

ACKNOWLEDGEMENTS

We would like to show our greatest appreciation to Dr. Yashwant Singh, our esteemed project guide and teacher. Even saying thank you will not be enough for his tremendous support and help. We feel motivated and encouraged every time we attend his meeting. Without his encouragement and guidance this project would not have materialized.

The guidance and support received from all the members who contributed and who are contributing to this project was vital for the success of the project. We are grateful for their constant support and help.

We would also like to thank our project coordinator Dr Nitin for all the facilities provided to us for the working of this project and also the lab coordinators for the environment provided to us.

Dated: 16/5/13

Ankur Nadda and Sahil Gupta

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This project reviews the potential benefits that can be obtained by the implementation of data fusion in a multi-sensor environment. A thorough review of the commonly used data fusion frameworks is presented together with important factors that need to be considered during the development of an effective data fusion problem-solving strategy. A system-based approach is defined for the application of data fusion systems within engineering. Kalman filtering techniques and algorithms have been studied thoroughly along with Bayesian probabilistic techniques being used for effective data predictions. We have studied various network simulators, and NS-2 has been selected as the best appropriate simulation environment for this project. The scope of this project is not only limited to the simulation of small scale network but can also be used for large size networks using Tree Aggregation Model and Kalman filtering techniques.

CHAPTER 1- INTRODUCTION

1.1 Introduction

A Wireless Sensor Network (WSN) is a special type of *ad hoc* network composed of a large number of nodes equipped with different sensor devices. This network is supported by technological advances in low power wireless communications along with silicon integration of various functionalities such as sensing, communication, and processing. WSNs are emerging as an important computer class based on a new computing platform and networking structure that will enable novel applications that are related to different areas such as environmental monitoring, industrial and manufacturing automation, health-care, and military. Commonly, wireless sensor networks have strong constraints regarding power resources and computational capacity.

A WSN may be designed with different objectives. It may be designed to gather and process data from the environment in order to have a better understanding of the behavior of the monitored entity. It may also be designed to monitor an environment for the occurrence of a set of possible events, so that the proper action may be taken whenever necessary. A fundamental issue in WSNs is the way the collected data is processed. In this context, information fusion arises as a discipline that is concerned with how data gathered by sensors can be processed to increase the relevance of such a mass of data. In a nutshell, information fusion can be defined as the combination of multiple sources to obtain improved information (cheaper, greater quality, or greater relevance).

1. Information fusion is commonly used in detection and classification tasks in different application domains, such as robotics and military applications. Lately, these mechanisms have been used in new applications such as intrusion detection and Denial of Service (DoS) detection Within the WSN domain, simple aggregation techniques (e.g., *maximum*, *minimum*, and *average*) have been used to reduce the overall data traffic to save energy. Additionally, information fusion techniques have been applied to WSNs to improve location estimates of sensor nodes, detect routing failures [Nakamura et al. 2005b], and collect link statistics for routing protocols [Woo et al. 2003]. Given the importance of information fusion for WSNs, this work surveys the

state of-the-art related to information fusion and how it has been used in WSNs and sensor based systems in general. This background is presented in the following structure.

Data fusion arises as a discipline that is concerned with how data gathered by sensors can be processed to increase the relevance of such a mass of data. Data fusion is combination of data from multiple sensors, and related information provided by associated databases, to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone. Data fusion has established itself as an independent research area over the last decades, but a general formal theoretical framework to describe data fusion systems is still missing. One reason for this is the huge number of disparate research areas that utilize and illustrate some form of data fusion in their context of theory. For example, the concept of data or feature fusion, which forms together with classifier and decision fusion the three main divisions of fusion levels, initially occurred in multi-sensor processing. By now several other research fields found its application useful. Besides the more classical data fusion approaches in statistics, control, robotics, computer vision, geosciences and remote sensing, artificial intelligence, and digital image/signal processing, the data retrieval community discovered some years ago its power in combining multiple data sources.

Methods, techniques and algorithms used to fuse data can be classified based on several criteria, such as the data abstraction level, purpose, parameters, type of data and mathematical foundation. The classification presented is based on the method's purpose. According to this criterion, information fusion can be performed with different objectives such as inference, estimation, classification, feature maps, abstract sensors, aggregation and compression.

Simulation is an essential tool to study Wireless Sensor Networks due to the unfeasibility of analysis and the difficulties of setting up real experiments. Ns or Network Simulator is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks

.

1.2 Problem Statement

The task is to simulate wireless sensors on a simulator like Network Simulator-2, from those simulated nodes collect some virtual data and which may be redundant and simulate real time data collection and apply data fusion techniques and algorithms for collection of optimal and correct data and use that data for future predictions using probabilistic techniques.

1.3 Objectives

- Study of algorithms related to data fusion and information fusion.
- A data fusion node collects the results from multiple nodes. It fuses the results with its own based decision criterion.
- Integrate uncertain information using multiple tree aggregation.
- Simulation of Network using Network Simulator 2.
- Use one of the techniques of Complementary Fusion and data fusion models.
- Applying Probabilistic techniques and algorithms for future predictions from old data.
- Make GPS locator android application

1.4 Methodology

The process wherein, sensors detect an event, and the data related to the event is eventually fused at the sink via multiple levels of fusion en route,is called a ' round˜ of aggregation. In this project, we propose a protocol that allows a node to determine when to start and finish the fusion process during a round of aggregation in order to ensure the desired trade-off between the achieved credibility and the incurred latency. The rest of this project is organized as follows in sections. In section II, we describe related work on data aggregation in sensor networks. In section III, we describe the system model that we use in our simulation experiments. In section IV, we look at a simple but unrealistic network to understand the impact of various parameters. We will describe the proposed protocol Is section V. Our simulation results will be

presented in section VI. All the sections will be implemented systemically during the course of this project and the final results will be shown by the end of the term.

1.5 Organization of Report

Here we present the overview of contents and study of the report and its organisation

1.5.1 Literature Survey

In the literature survey, thorough study was done in relation to wireless sensors, its applications in the real world, the fusion techniques and its advantages and some of the detailed algorithms of implementation. The main topics being discussed are Data fusion and related algorithms, the fusion models, the algorithms and simulation techniques and predictions.

1.5.2 Algorithms and Architecture

In this part of the project report, we have given details about the architecture model and algorithms. Kalman filtering is the major study of the project which is one of the key techniques of Data Fusion and aggregation. Bayesian Inference and techniques if another field of study being done in detail. The algorithm and the tree aggregation model is the key aspect of the study.

1.5.3 The GPS Locator Application

In this part of the study, we have designed an application for android platform which is a practical implementation of the wireless sensors networks and is used for finding the location and position of sensor which we are interested in and its characteristics. The application is useful in fault assessment and prediction.

1.5.4 Simulation and Results

Here we are simulating a network on a simulator known as Network Simulator 2 on linux based environment. The simulation is the direct implementation of the Tree Aggregation Model.

CHAPTER 2 LITERATURE SURVEY

2.1 Fundamentals

Several different terms (e.g. data fusion, sensor fusion, and information fusion) have been used to describe aspects of the fusion subject (including theories, processes, systems, frameworks, tools, and methods). Consequently, there is a terminology confusion. This section discusses common terms and factors that motivate and encourage the practical use of information fusion in WSNs.

2.1.1 Terms

The terminology related to systems, architectures, applications, methods, and theories about the fusion of data from multiple sources is not unified. Different terms have been adopted, usually associated with specific aspects that characterize the fusion. For example, Sensor/Multisensor Fusion is commonly used to specify that sensors provide the data being fused. Despite the philosophical issues about the difference between data and information, the terms Data Fusion and Information Fusion are usually accepted as overall terms. Data fusion as "the combination of data from multiple sensors, and related information provided by associated databases, to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone." Here, data fusion is performed with an objective: accuracy improvement. However this definition is restricted to data provided by sensors; it does not foresee the use of data from a single source Multi sensor Integration is another term used in robotics/computer vision and industrial automation .Multi sensor integration "is the synergistic use of information provided by multiple sensory devices to assist in the accomplishment of a task by a system; and multi sensor fusion deals with the combination of different sources of sensory information into one representational format during any stage in the integration process."Multi sensor integration is a broader term than multi sensor fusion. It makes explicit how the fused data is used by the whole system to interact with the environment. Information Fusion stating that "in the context of its usage in the society, it encompasses the theory, techniques and tools created and applied to exploit the synergy in the information acquired from multiple sources (sensor, databases, information gathered by humans, etc.) in such a way that the resulting decision or

action is in some sense better (qualitatively or quantitatively, in terms of accuracy, robustness ,etc.) than would be possible if any of these sources were used individually without such synergy exploitation." Possibly, this is the broadest definition embracing any type of source, knowledge, and resource used to fuse different pieces of information.

The term Data Aggregation has become popular in the wireless sensor network community as a synonym for information fusion . Data aggregation comprises the collection of raw data from pervasive data sources, the flexible, programmable composition of the raw data into less voluminous refined data, and the timely delivery of the refined data to data consumers." By using 'refined data', accuracy improvement is suggested. However, as van Renesse [2003] defines, "aggregation is the ability to summarize," which means that the amount of data is reduced. For instance, by means of summarization functions, such as *maximum* and *average*, the volume of data being manipulated is reduced. However, for applications that require original and accurate measurements, such a summarization may represent an accuracy loss . In fact, although many applications might be interested only in summarized data, we cannot always assert whether or not the summarized data is more accurate than the original data-set. For this reason, the use of data aggregation as an overall term should be avoided because it also refers to one instance of information fusion: summarization. Figure depicts the relationship among the concepts of multisensor/sensor fusion, multisensor integration, data aggregation, data fusion, and information fusion. Here, we understand that both terms, data fusion and information fusion, can be used with the same meaning. Multisensor/sensor fusion is the subset that operates with sensory sources. Data aggregation defines another subset of information fusion that aims to reduce the data volume (typically, summarization), which can manipulate any type of data/information, including sensory data. On the other hand, multisensor integration is a slightly different term in the sense that it applies information fusion to make inferences using sensory devices and associated information (e.g., from database systems)to interact with the environment. Thus, multisensor/sensor fusion is fully contained in the intersection of multisensor integration and information/data fusion.

Here, we chose to use information fusion as the overall term so that sensor and multisensor fusion can be considered as the subset of data fusion that handles data acquired by sensory devices.

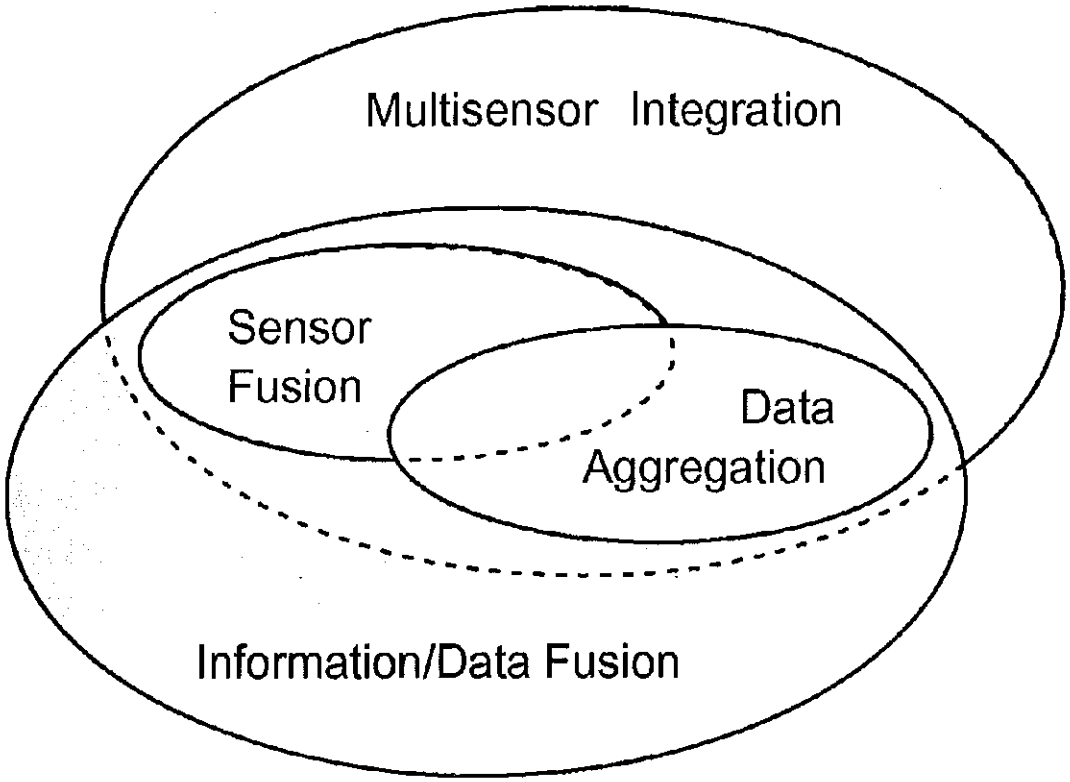


Fig 1: Fusion Terms

2.1.2 Some Limitations

Data fusion should be considered a critical step in designing a wireless sensor network. The reason is that data fusion can be used to extend the network lifetime and is commonly used to fulfil application objectives, such as target tracking, event detection, and decision making. Hence, blundering data fusion may result in waste of resources and misleading assessments. Therefore, we must be aware of possible limitations of information fusion to avoid blundering situations. Because of resource rationalization needs of WSNs, data processing is commonly implemented as in-network algorithms . Hence, whenever possible, data fusion should be performed in a distributed (in-network) fashion to extend the network lifetime. None the less, we must be aware of the limitations of distributed implementations of data fusion. In the early 1980s, some argued that, regarding the communication load, a centralized fusion system may outperform a distributed one. The reason is that centralized fusion has a global knowledge in the sense that all measured data is available, whereas distributed fusion is incremental and localized since it fuses measurements provided by a set of neighbour nodes and the result might be further fused by intermediate nodes until a

sink node is reached. Such a drawback of decentralized fusion might often be present in WSNs wherein, due to resource limitations, distributed and localized algorithms are preferable to centralized ones. In addition, the lossy nature of wireless communication challenges information fusion because losses mean that input data may not be completely available. Another issue regarding information fusion is that, intuitively, one might believe that in fusion processes, the more data the better, since the additional data should add knowledge (e.g., to support decisions or filter embedded noise).

2.2 Classification of Data Fusion

Information fusion can be categorized based on several aspects. Relationships among the input data may be used to segregate information fusion into classes (e.g., cooperative, redundant, and complementary data). Also, the abstraction level of the manipulated data during the fusion process (measurement, signal, feature, decision) can be used to distinguish among fusion processes. Another common classification consists in making explicit the abstraction level of the input and output of a fusion process. These common classifications of information fusion are explored in this section.

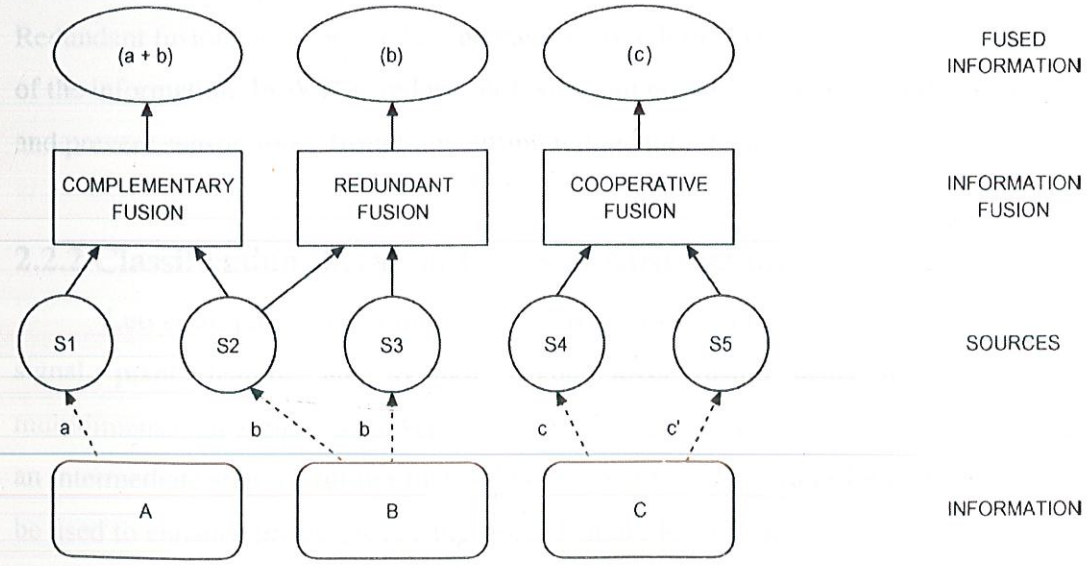


Fig. 2. Types of information fusion based on the relationship among the sources

2.2.1 Classification Based on Relationship Among the Sources

Complementary. When information provided by the sources represents different portions of a broader scene, information fusion can be applied to obtain a piece of information that is more complete (broader). In Figure 2, sources S1 and S2 provide different pieces of information, a and b, respectively, that are fused to achieve a broader information, denoted by (a+b), composed of non redundant pieces a and b that refer to different parts of the environment (e.g., temperature of west and east sides of the monitored area).

Redundant. If two or more independent sources provide the same piece of information, these pieces can be fused to increase the associated confidence. Sources S2 and S3in Figure 2 provide the same information, b, which is fused to obtain more accurate information, (b).

Cooperative. Two independent sources are cooperative when the information provided by them is fused into new information (usually more complex than the original data)that, from the application perspective, better represents the reality. Sources S4 and S5,in Figure 2, provide different information, c and c_, that are fused into (c), which better describes the scene compared to c and c_ individually.

Complementary fusion searches for completeness by compounding new information from different pieces. An example of complementary fusion consists in fusing data from sensor nodes (e.g., a sample from the sensor field) into a feature map that describes the whole sensor field, hence broader information.

Redundant fusion might be used to increase the reliability, accuracy, and confidence of the information. In WSNs, redundant fusion can provide high quality information and prevent sensor nodes from transmitting redundant information.

2.2.2 Classification Based on Levels of Abstraction

Luo et al. [2002] use four levels of abstraction to classify information fusion: signal, pixel, feature, and symbol. Signal level fusion deals with single or multidimensional signals from sensors. It can be used in real-time applications or as an intermediate step for further fusions. Pixel level fusion operates on images and can be used to enhance image-processing tasks. Feature level fusion deals with features or attributes extracted from signals or images, such as shape and speed. In symbol level fusion, information is a symbol that represents a decision, and it is also referred to as decision level. Typically, the feature and symbol fusions are used in object

recognition tasks. Such a classification presents some drawbacks and is not suitable for all information fusion applications. First, both signals and images are considered raw data usually provided by sensors, so they might be included in the same class. Second, raw data may not be only from sensors, since information fusion systems might also fuse data provided by databases or human interaction. Third, it suggests that a fusion process cannot deal with all levels simultaneously. In fact, information fusion deals with three levels of data abstraction: measurement, feature, and decision. Thus, according to the abstraction level of the manipulated data, information fusion can be classified into four categories:

Low-Level Fusion. Also referred to as *signal (measurement) level fusion*. Raw data are provided as inputs, combined into new piece of data that is more accurate (reduced noise) than the individual inputs.

Medium-Level Fusion. Attributes or features of an entity (e.g., shape, texture, position)are fused to obtain a feature map that may be used for other tasks (e.g., segmentation or detection of an object). This type of fusion is also known as *feature/attribute level fusion*. Examples of this type of information fusion include estimation of fields or feature maps [Nowak et al. 2004; Singh et al. 2006] and energy maps (see Section 4.3 for a feature map description).

High-Level Fusion. Also known as *symbol* or *decision level fusion*. It takes decisions or symbolic representations as input and combines them to obtain a more confident and/or a global decision. An example of high-level fusion is the Bayesian approach for binary events.

Multilevel Fusion. When the fusion process encompasses data of different abstraction levels—when both input and output of fusion can be of any level (e.g., a measurement is fused with a feature to provide a decision)—multilevel fusion takes place.

2.2.3 Classification Based on Input and Output

Another well-known classification that considers the abstraction level is provided by Dasarathy [1997], in which information fusion processes are categorized based on the abstraction level of the input and output information. Dasarathy identifies five categories:

Data In–Data Out (DAI-DAO). In this class, information fusion deals with raw data

and the result is also raw data, possibly more accurate or reliable.

Data In–Feature Out (DAI-FEO). Information fusion uses raw data from sources to extract features or attributes that describe an entity. Here, "entity" means any object, situation, or world abstraction.

Feature In–Feature Out (FEI-FEO). FEI-FEO fusion works on a set of features to improve/refine a feature, or extract new ones.

Feature In–Decision Out (FEI-DEO). In this class, information fusion takes a set of features of an entity generating a symbolic representation or a decision.

Decision In–Decision Out (DEI-DEO) Decisions can be fused in order to obtain new decisions or give emphasis on previous ones.

2.3 Method, Techniques and Algorithms

2.3.1 Inference

Inference methods are often applied in decision fusion. In this case, a decision is taken based on the knowledge of the perceived situation. Here, inference refers to the transition from one likely true proposition to another, whose truth is believed to result for the previous one. Classical inference methods are based on Bayesian inference.

Bayesian Inference. Information fusion based on Bayesian Inference offers a formalism to combine evidence according to rules of probability theory. The uncertainty is represented in terms of conditional probabilities describing the belief, and it can assume values in the [0, 1] interval, where 0 is absolute disbelief and 1 is absolute belief. Bayesian inference is based on the rather old Bayes' rule [Bayes 1763], which states that:

$$\Pr(Y \mid X) = \Pr(X \mid Y)\,\Pr(Y)/\Pr(X)$$

where the posterior probability $\Pr(Y \mid X)$ represents the belief of hypothesis Y given the information X. This probability is obtained by multiplying $\Pr(Y)$, the prior probability of the hypothesis Y, by $\Pr(X \mid Y)$, the probability of receiving X, given that Y is true; $\Pr(X)$ can be treated as a normalizing constant. The main issue regarding Bayesian Inference is that the probabilities $\Pr(X)$ and $\Pr(X \mid Y)$ have to be estimated or guessed beforehand since they are unknown.

Pan et al. [1998] propose the use of neural networks to estimate conditional probabilities to feed a Bayesian Inference module for decision-making. Sam et al.

[2001]use Bayesian Inference to decide whether or not a system's voltage is stable by fusing three stability indicators of a small power system. Cou´e et al. [2002] use Bayesian programming, a general approach based on an implementation of Bayesian theory, to fuse data from different sensors (e.g., laser, radar, and video) to achieve better accuracy and robustness of the information required for high-level driving assistance. Typical usage for Bayesian Inference includes robotic map building Within the WSNs domain, Bayesian Inference has been used to solve the localization problem. information from a mobile beacon and determine the most likely geographical location(region) of each node, instead of finding a unique point for each node location.

2.3.2 Feature Maps

For some applications, such as guidance and resource management, it might not be feasible to directly use raw sensory data. In such cases, features representing aspects of the environment can be extracted and used by the application. Commonly, diverse fusion methods of estimation and inference can be used to generate a feature map. Here, we explore two special types of feature maps: occupancy grid and network scans.

Occupancy Grid. Occupancy grids, also called occupancy maps or certainty grids, define a multidimensional (2D or 3D) representation of the environment, describing which areas are occupied by an object and/or which areas are free spaces. According to Elfes [1989], an occupancy grid is "a multidimensional random field that maintains stochastic estimates of the occupancy state of the cells": the observed space is divided into square or cubic cells and each cell contains a value indicating its probability of being occupied. Usually, such probability is computed—based on information provided by several sensors—using different methods, such as Bayesian theory, Dempster-Shafer reasoning, and fuzzy set theory .Occupancy grids were initially used to build an internal model of static environments

based on ultrasonic data , and since then several variations have been proposed. Typical applications of occupancy grids include position estimation , robot perception and navigation . There are also applications in computer graphics, such as simulation of graphical creatures behaviour and collisions detection of volumetric objects.

Network Scans. Network Scans are defined as a sort of resource/activity map for wireless sensor networks. Analogous to a weather map, the network scan depicts the geographical distribution of resources or activity of a WSN. By considering a resource of interest, instead of providing detailed information about each sensor node in the network, these scans offer a summarized view of the resource distribution. The network scan implemented by Zhao et al. [2002b] is called eScan and it retrieves information about the residual energy in the network in a distributed *in-network* fashion. The algorithm is quite simple. First, an aggregation tree is formed to determine how the nodes will communicate. Second, each sensor computes its local eScan and whenever the energy level drops significantly since the last report, the node sends its eScan towards the sink. The eScans are aggregated whenever a node receives two or more topologically

adjacent eScans that have the same or similar energy level. The aggregated eScan is a polygon corresponding to a region, and the summarized residual energy of the nodes within that region. Each energy level is assigned a gray level and the result is a 2D image (map) where white regions have nodes with full charge and black regions have dead nodes. Although this algorithm makes unlikely assumptions for sensor networks, such as a perfect MAC (Medium Access Control) layer with no loss or overhead due to contention or environment changes, the network scan poses an interesting fusion method to present information about the network resources and activity. In the particular case of eScan, it allows the identification of low energy regions, helping designers decide where new sensors should be deployed. In addition, the network may use eScans to reorganize itself, so nodes with low energy levels are spared.

2.3.3 Aggregation

Kulik et al. [2002] define data aggregation as a technique used to overcome two problems: *implosion* and *overlap*. In the former, data sensed by one node is duplicated in the network due to the data routing strategy (e.g., flooding). The *overlap* problem happens when two different nodes disseminate the same data. This might occur when the sensors are redundant—they sense the same property in the same place. In both cases, redundancy, which occurs due to different reasons, might have its negative impact(e.g., waste of energy and bandwidth) reduced by data aggregation and information fusion. Aggregation techniques are the common summarization

functions used by query languages (e.g., SQL) to retrieve summarized data in database systems. The use of data aggregation in WSN sand its impact on energy consumption is the subject for further research. Other aggregation functions can be identified in WSNs, which are *suppression* and *packaging* . The former function simply suppresses redundant data by discarding duplicates. For example, if a node senses the temperature 45◦C and receives the same observation from a neighbour, then only one packet containing a 45◦C observation will be forwarded. The second aggregation function groups several observations in one single packet. The objective of this strategy is to avoid the overhead of the MAC protocol when sending several packets. However, *packaging* may not be classified as a fusion technique because it does not exploit the synergy among the data. *Packaging* is actually a solution to optimize the usage of a communication protocol, which is independent of any fusion method

2.3.4 Data Fusion Models

To ensure that systems are operating within defined conditions, measurements are taken which, when analysed, enable decisions to be made based on condition. These measurements can produce data that are either very similar, often from the same sensor, or completely different from different techniques. Experienced engineers and analysts have traditionally undertaken the analysis of this data. However, with the increased computer power and development of new and novel detection systems, the data produced needs to be handled in a robust and logical manner. As such computer systems have been developed that are capable of extracting meaningful information from the recorded data. The integration of data, recorded from a multiple sensor system, together with knowledge, is known as data fusion.

There are numerous advantages in using multiple sensor systems including:

- Higher signal-to-noise ratio;
- Increased robustness and reliability in the evident of sensor failure;
- Information regarding independent features in the system can be obtained;
- Extended parameter coverage, rendering a more complete picture of the system;

- Increased dimensionality of the measurement;
- Improved resolution;
- Reduced uncertainty;
- Increased confidence;

The actual combination of sensors is dependent upon the requirements of the system. However, a number of things need to be considered when defining the type of fusion algorithm used and level at which fusion will occur. These include:

- How are the sensors distributed ?
- What are the format, type and accuracy of the collected data?
- What is the nature of the sensors used?
- What is the resolution of the sensors used?
- What is the computational capability at the sensors?

Data can be combined either as it arrives into the system or at a defined level within the fusion process. The reliability of the data used within the fusion system will depend on the sensors available and the methodology employed for the fusion of the data. The selection of sensors as well as the number of sensors needed to increase the accuracy of the information transferred depends on the problem at hand. Different types of sensors can be used depending on the application and the output format sought. Table 1 gives a brief overview of sensors typically used in data fusion. Sensors are usually classified according to their physical nature. They are often based on the electromagnetic spectrum, sound waves, touch, odour, or the absolute position of the system.

2.4 Framework for the implementation of data system

A number of data fusion frameworks have been developed both within the research and commercial environments. These frameworks have been used in

numerous projects to aid the development of fusion systems by establishing the most appropriate algorithm for the defined problem.

2.4.1 JDL Process Model

One of the most widely used frameworks and the one being used in this project is the *JDL Data Fusion Framework.* The Joint Directors of Laboratories (JDL) data fusion sub-panel within the US Department of Defence originally defined this system in the early years of data fusion. This framework was developed to aid the developments in military applications.

- **Level 1, object refinement,** attempts to locate and identify objects. For this purpose a global picture of the situation is reported by fusing the attributes of an object from multiple sources. The steps included at this stage are: Data alignment, prediction of
  entity's attributes (i.e. position, speed, type of damage, alert status, etc.), association of data to entities, and refinement of entity's identity.

- **Level 2, situation assessment,** attempts to construct a picture from incomplete information provided by level 1, that is, to relate the reconstructed entity with an observed event (e.g. aircraft flying over hostile territory).

- **Level 3, threat assessment,** interprets the results from level 2 in terms of the possible opportunities for operation. It analyses the advantages and disadvantages of taking one course of action over another.

A process refinement, sometimes referred as Level 4, loops around these three levels to monitor performance, identify potential sources of information enhancement, and optimise allocation of sensors. Other ancillary support systems include a data management system for storage and retrieval of pre-processed data and human-computer interaction. The lay out of this model process is depicted in Figure 3.

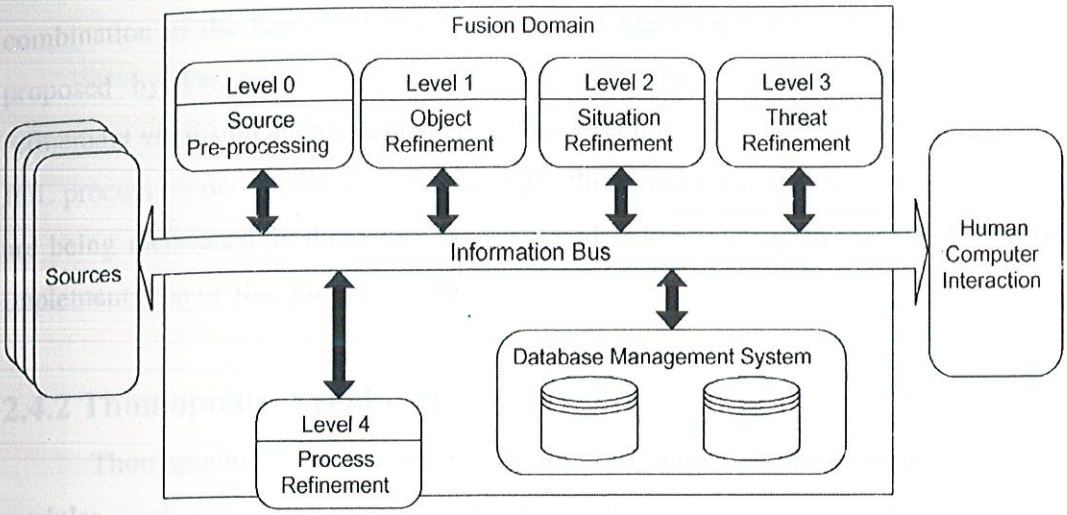


Fig 3 JDL Process Model

The hierarchical distribution of the JDL model allows for the different levels to be broken down into sub-levels. In this manner, level 1 could be further divided into four processes: Data alignment, data association, object estimation, and object identity.

- At the data alignment stage, the data is processed to attain a common spatial and time frame;
- The data association could be further divided as association performed among data units of the same variable and between data units of different variables. At this stage the degree of proximity among the variables is measured;
- Object estimation, on the other hand, could be sub-divided in terms of the processing approach taken (sequential or batch), parameter identification and estimate equations available, best-fit function criteria, and the optimisation of best-fit function approach sought. At this stage the data fusion centre estimates the object's position, velocity, or attributes;
- The object identity stage could be subdivided into feature extraction, identity declaration, and combination of identity declarations. At this stage a prediction of the object's identity or classification is declared.

At each of these lowest sub-levels, the mapping of different types of techniques could be easily allocated, and selected according to the case at hand. Fusion can be performed on raw data in the fusion centre (centralised process) or on pre-processed locally fused data (decentralised process). A hybrid data fusion system, consisting of

the integration of both raw and pre-processed data, could also be considered. The combination of the first three JDL levels into a blackboard data structure has been proposed by Paradis[16]. This framework is further integrated with a process refinement via fusion agents, which act as fusion centres.

JDL process model is being used throughout this project but still some other models are being mentioned as those too, were studied before coming to the conclusion of implementation of JDL process model.

2.4.2 Thomopolus Architecture

Thomopoulos posed an architecture for data fusion consisting of three modules, each integrating data at different levels or modules to integrate the data, namely:

- Signal level fusion, where data correlation takes place through learning due to the lack of a mathematical model describing the phenomenon being measured.

- Evidence level fusion, where data is combined at different levels of inference based on a statistical model and the assessment required by the user (e.g. decision making or hypothesis testing).

- Dynamics level fusion, where the fusion of data is done with the aid of an existing mathematical model

Depending upon the application, these levels of fusion can be implemented in a sequential manner or interchangeably. If continuous health monitoring of a machine is the objective, the combination of data could be done at the signal level, whilst higher order fusion (e.g. evidence fusion) would need to be applied if a wide range of decisions ought to be made from the signals. Figure 2 gives a summary of the architecture. Thomopoulos stressed the point that any data fusion system should consider three essential criteria to achieve the desired performance, these are:

- Robustness with respect to any a-priori uncertainty.

- Môn tonicity with respect to the fused information

- Môn tonicity with respect to the costs involved

In addition, factors such as the delay in the transmission of data, channel errors, and other communication aspects, as well as the spatial/temporal co-alignment of the data should also be taken into account in the data fusion system.

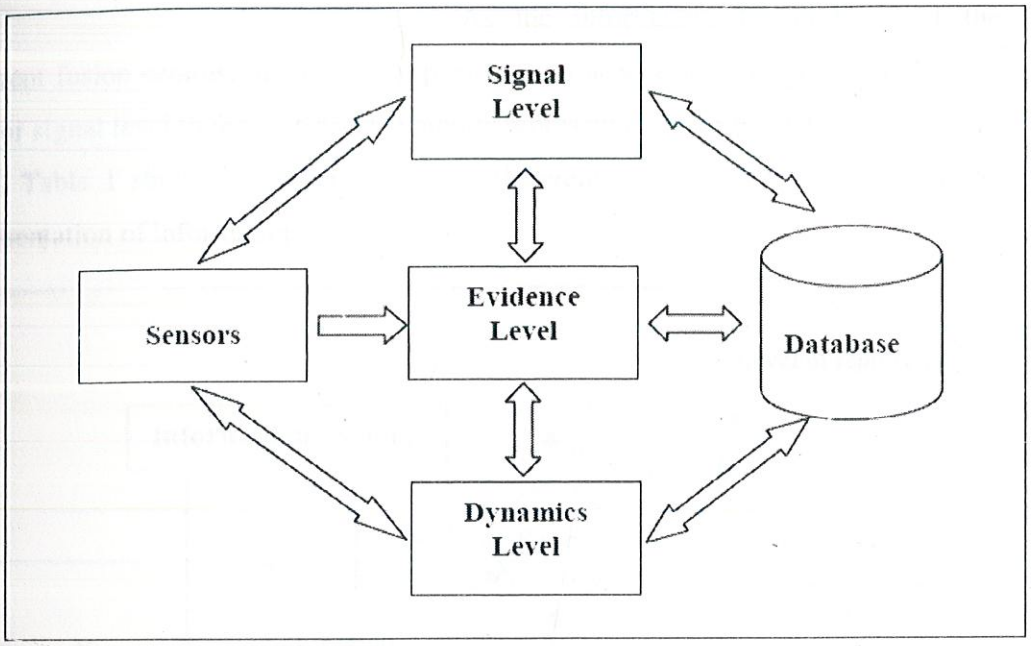


Fig 4 Thomopoulos's Data Fusion architecture

2.4.3 Multi sensor Integration Fusion Model

Luo and Kay[17] introduced a generic data fusion structure based on multi-sensor integration. In this system, data from various sources was combined within embedded fusion centres in a hierarchical manner. They made a clear distinction between multi-sensor integration and multi-sensor fusion. The former refers to the use of multiple sensor information to assist in a particular task, whilst the latter refers to any stage in the integration process where there is an actual combination of the data. Figure 5 shows a diagram of Luo and Kay's framework to represent multi-sensor integration and fusion simultaneously. From this diagram, the data collected at the sensor level is transferred to the fusion centres, where the fusion process takes place, in a

hierarchical and sequential manner. The entire framework shown in Figure 5 is a representation of multi-sensor integration. A description of the measured phenomenon is obtained after the outputs of the n sensors are processed, with the aid of the information system whenever appropriate. The fusion process is facilitated with the incorporation of an information system, containing relevant databases and libraries.

As the information is combined at the different fusion centres, the level of representation needed is increased from the raw data or signal level to more abstract symbolic representations of the data at the symbol level. Table 1 shows a comparison of the different fusion levels classified by the representation of information.

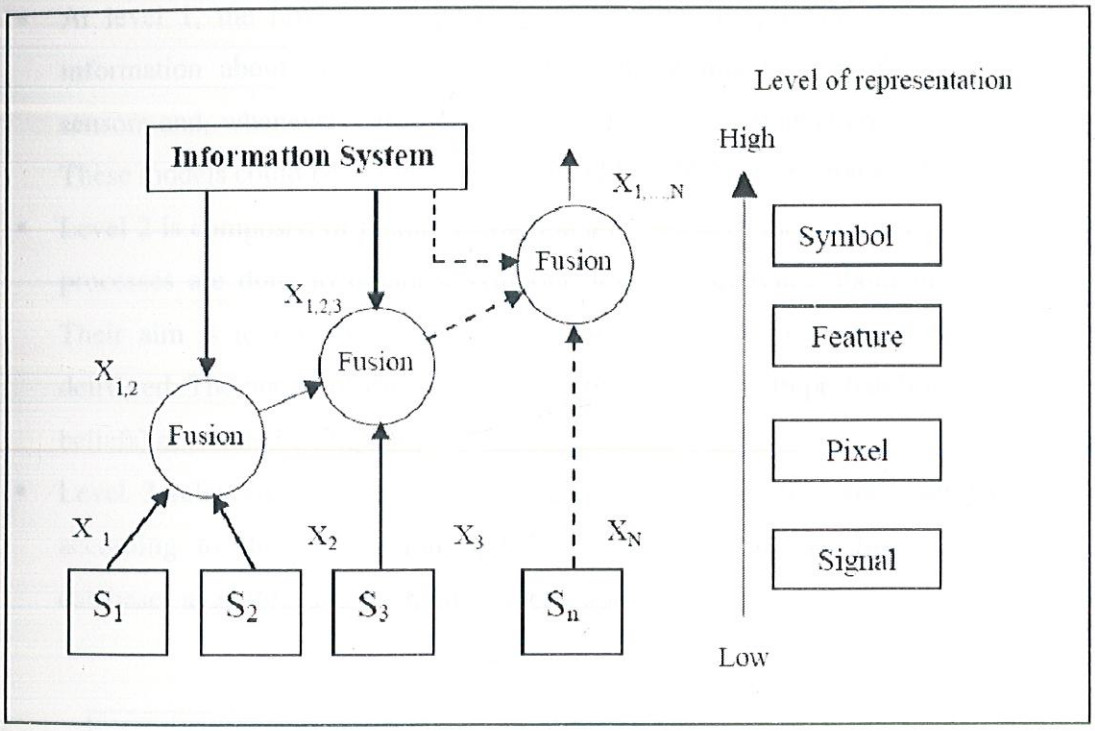


Fig 5 : Luo and Kay's architecture

Characteristics	Signal level	Pixel level	Feature level	Symbol level
Representation level of information	Low	Low	Medium	High
Type of sensory information	Multi-dimensional signal	Multiple images	Features extracted from signals/images	Decision logic from signals/image
Model of sensory information	Random variable with noise	Random process across the pixel	Non-invariant form of features	Symbol with degree of uncertainty

Table 1: Characteristics of data fusion level

2.4.4 Waterfall Model

Another example of hierarchical architecture commonly used by the data fusion community, called the waterfall model. A representation of this model is shown in figure 6. It can be seen from this figure that the flow of data operates from the data level to the decision making level. The sensor system is continuously updated with feedback information arriving from the decision-making module. The feedback element advises the multi-sensor system on re-calibration, re-configuration and data gathering aspects.

There are three levels of representation in the waterfall model, as shown in Figure 6:

- At level 1, the raw data is properly transformed to provide the required information about the environment. To achieve this task, models of the sensors and, whenever possible, of the measured phenomena are necessary. These models could be based on experimental analysis or on physical laws;
- Level 2 is composed of feature extraction and fusion of these features. These processes are done to obtain a symbolic level of inference about the data. Their aim is to minimise the data content whilst maximising information delivered. The output of this level is a list of estimates with probabilities (and beliefs) associated with them;
- Level 3 relates objects to events. Possible routes of action are assembled according to the information that has been gathered, the libraries and databases available, and the human interaction.

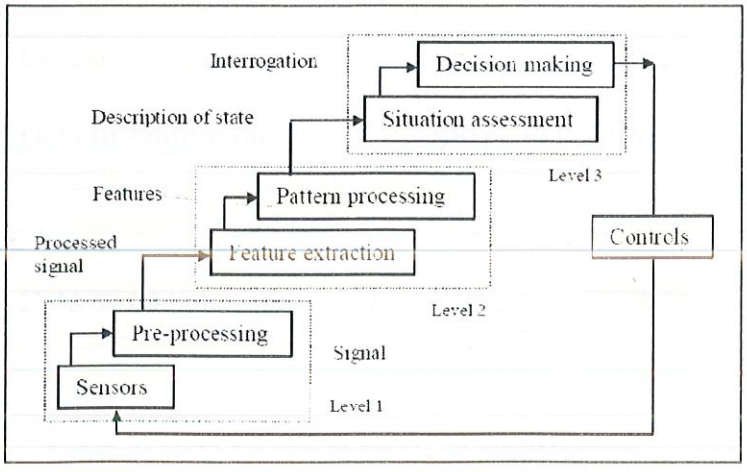


Fig 6: Waterfall Model

2.5 Issues In Data Fusion Models

Before a robust data fusion strategy can be legitimately submitted, there is a need to underline some of the difficulties arising with the application of data fusion, as well as other features that could be incorporated into the proposed model process. Some of the difficulties arising in multi-sensor data fusion could be summarised as follows:

- Diversity of sensors used: nature, synchronisation, location, and sensor outputs.
- Diversity of data representation: image, spatial, statistical, and textual.
- Registration: the information refers to the same entity. There is a need to check the consistency of the sensor measurements. This can be improved by objectively eliminating fallacious data sets.
- Calibration of the sensors when errors in the system operation occur.
- Limitations in the operability of the sensors.

Mode	Example
Data in-data out	Fusion of multi-spectral data
Feature in-feature out	Fusion of image and non-image data
Decision in-decision out	When sensors are not compatible
Data in-feature out	Shape extraction
Feature in-decision out	Object recognition
Data in-decision out	Pattern recognition

Table 2: Data Fusion in terms of input/output provided

Some important architectural issues needed for the implementation of a process model for data fusion are:

- Network configuration of sensors: parallel or serial multi-sensor suite, or a combination of the two. A parallel sensor arrangement is best suited for either identical or dissimilar sensors. Serial sensor configurations are very practical when one sensor delivers complementary information to the next.

- Level of representation of the information: Although a three level system is commonly used, description of the fusion process based on input/output modes, as shown in Table 3, can aid in level selection, and adds flexibility to the JDL model[21].

- Feedback within the data fusion network of fused information with the aid of a sensor management suite. The suite would coordinate the data, handle information flow, and store the data in a database.

- The fusion of data can be done on either raw data (centralised process) or on pre-processed locally fused data (decentralised process). Hybrid data fusion, consisting of fusion of both raw and pre-processed data, can also be considered. The centralised architecture is computationally intensive, but it carries the advantage of developing a global view of the object from the original data. On the other hand a decentralised architecture is less demanding on computational capabilities at the cost of adding complexity to the data fusion process, since each sensor has a processing unit.

- Other issues are related to these difficulties arising in data fusion, and the ability of the system to deal with them (i.e. sensor failures, corrupted data, compatibility of sensors).

2.6 Identification

The identification process is aided, whenever appropriate, by the application of data mining techniques. At this stage, inference about the

system takes place, interrogating the various factors used in the data fusion process:

- What is the information gained by using data fusion? This would be the first question one ought to ask before characterising a problem. It is important to identify performance criteria to identify if the data fusion process is worth doing.

- Understand the physical-chemical phenomenon under study: collect information available by fusing people's knowledge about the problem and propose a model and/or state equations describing the phenomenon. If a model already exists, it should be used and understood.

- Know all your data sources (e.g. sensors, databases, libraries). Especially, identify how the data have been collected, measuring techniques used, availability of processed data, and other issues regarding the fine tuning of the data sources, such as calibration, effects of human interaction, and missing data

- Analyse the data in more depth before mathematical manipulation takes place. Issues to consider include.

- checking and adjusting for the synchronisation of separate data streams;

- Identifying the true dimensionality, and trying ways to reduce it, without reducing the information content;

- Identifying whether the data is concentrated or sparse, and hence choosing appropriate methods for pre-processing;

- checking the repeatability of measurements, and likely error;

- examining the built-in redundancy of the sensor system to ensure a robust data collection process.

- Identify the dominant uncertainty in the system and whether this can be corrected or minimised. Uncertainty could take three forms:

- stochastic noise which cannot be corrected per se, but which could be compensated statistically;

- systematic error which might be corrected by calibration or modelling;

- unknowns, e.g. the transfer function between the real state and the measured state in a non-invasive measurement, or simply a missing parameter.

- Identify the level at which fusion must take place. Usually, data collected from similar sensors can be combined at the lowest level of inference, while data arriving from dissimilar sensors must be fused at higher levels. Fusion at a feature level, or integration of knowledge for decision-making, always occurs at a higher level.

CHAPTER 3: ALGORITHM AND ARCHITECTURE

3.1 Tree Aggregation Model:

The ideas we propose in this project are independent of the fusion model used; however, to simplify our analyses, we assume the data that a sensor generates only represents whether or not an event occurs. A fused report would simply contain a count of the number of reports that either confirm or dispute the occurrence of the event. Following are the sequential steps being followed.

Typically, the sink or the parent node is distant from the area where the sensor nodes reside.

The sensor data has to be ultimately relayed to the sink via multiple sensor node relays or internal nodes. This facilitates the many to-one data transport. This is somewhat similar to multicast tree, the difference is that instead of data propagating from the single source to the multicast group members, the data flows in the opposite direction, i.e., from the members to the sink.

Data may be fused at various level points on the tree. The topology of this aggregation tree determines the efficiency with which data may be fused, to a certain extent. Our only requirement is that each sensor node is aware of its immediate neighbours; specifically it should know its parent, i.e., the sensor node to which it sends data (either fused or raw) and its children, sensors from whom it receives data.

Each non-leaf node or internal node is responsible for relaying (after possibly performing fusion) data received from its children towards the sink node.

Aggregation tree is formed at the network initialization phase, and is dynamically re-organized as sensors sleep, wake up or fail. We note that in the aggregation tree, we refer to nodes as being at particular levels.

The leaves are at the lowest level (Level 0) whereas the sink is at the highest level. Furthermore, note that multiple trees may be formed for gathering information from multiple (possibly geographically separate) sensor sets.

We assume a high density of sensor nodes and that multiple sensors detect each event. The credibility of the final report at the sink is directly reflected by the total number of reports that are fused.

We restrict ourselves to the occurrence of a single event in this work. Multiple events can be treated individually by using the same method. Note that by either including query identifiers or by associating time-stamps and geographical position with events, one could identify a particular event.

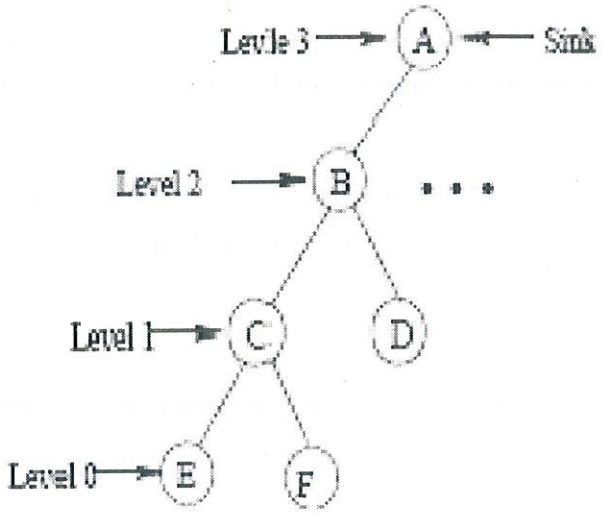


Fig 7 : The model

3.2 The Kalman Filter

In his 1960 famous publication (*"A new approach to linear filtering and prediction problems"*, Trans.ASME J. Basic Engineering., vol 82, March 1960, pp 34-45), Rudolf Kalman based the construction of the state estimation filter on probability theory, and more specifically, on the properties of conditional Gaussian random variables. The criterion he proposed to minimize is the state vector covariance norm, yielding to the classical recursion : the new state estimate is deduced from the previous estimation by addition of a correction term proportional to the prediction error (or the innovation of the measured signal). If one tries to explain this remarkable and elegant construction to students having not a sufficient background in probability theory, there is an inherent difficulty (it is perhaps preferable to speak of a quite paradoxal aspect): its understanding requires a good knowledge of conditional gaussian random variables, while in the simple and efficient final formulation, the corresponding intermediate steps are not visible. In this presentation, I give a first construction based on probability theory in simple cases where Kalman approach is quite easy to follow: assuming the linearity of the predictor and avoiding the

construction in the case of gaussian variables (this formulation based on linearity is mentioned in Kalman's paper, but not developed, probably because considered as obvious by him).

The Kalman filter, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing noise (random variations) and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. (Source Definition: Wikipedia)

Now Kalman filter is just *optimal recursive data processing algorithm.*

It processes all available measurements regardless of precision to estimate current variables of interest.

For example to measure the velocity of any aircraft we can use a radar system, navigation system, pressure methods etc. whereas Kalman filter can be used to combine all this data to have more precise information in hand.

3.2.1 A Simple Case: Estimation of scalar

In order to estimate a constant m, we do several measurements on it;

Let y be one of these measurements. $y$is a random variable with average m and fluctuations or noise v.

$y=m+v$

The zero mean noise v has variance σ_v^2 .

We will perform a recursive estimation of m: we suppose that we

have a first bias less estimation of m in the form of a random variable x_0.

$x_0=m+w$

The zero mean noise v has variance σ_w^2 .

We intend to compute a new estimate $x1$ with the following form :

$x_1=x_0+ k(y-x_0)$

The specific characteristics of this correction are

1. The new estimate is a linear function of the previous estimate and of the measurement ;

2.The previous and the new estimators have no bias, or equivalently : if the measure y is perfectly predicted by the previous estimate x_0 , this implies that it not necessary to correct this estimate

We note that the average of the new estimate is m: in replacing x_0 and y by their values :

$x_1 = m + w + k(m + v - m - w)$

$x1 = m + w + k(v - w)$

$E(x1) = m$

We compute the variance of the new estimator assuming that the noise v is independent of x_0 ; so the variance of x_1 is:

$\sigma_1^2 = E(x_1 - m)^2 = E[w + k(v - w)]^2$

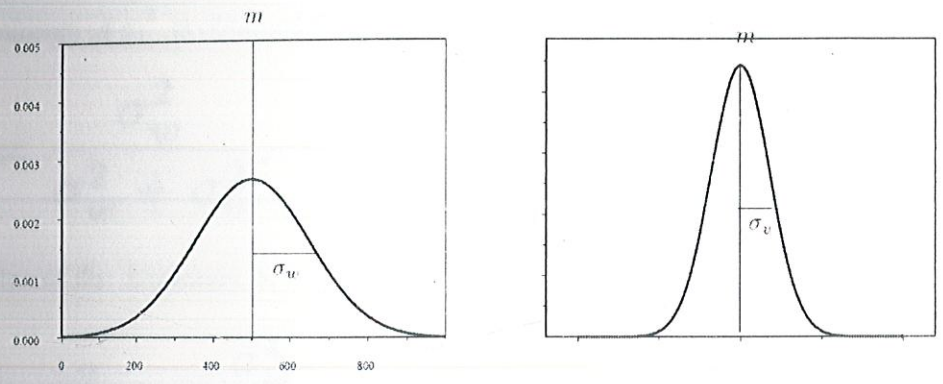


Fig 8 Two biasless estimators with variances σ_w^2 and σ_v^2; we look for a linear combination of these estimators giving a new estimator with lowest variance.

Assuming the independence of the two noises v and w

$$\sigma_1^2 = (1-k)^2 E[w]^2 + k^2 E[v]^2,$$

$$\sigma_1^2 = (1-k)^2 \sigma_w^2 + k^2 \sigma_v^2.$$

It is reasonable to look for the estimate $x1$ with lower variance and to compute the corresponding k. For this purpose we write

$$\sigma_1^2 = (1 - 2k + k^2)\sigma_w^2 + k^2 \sigma_v^2,$$

$$\sigma_1^2 = \sigma_w^2 - 2k\sigma_w^2 + k^2(\sigma_w^2 + \sigma_v^2),$$

where we exhibit constant term and a quadratic function of k

$$\sigma_1^2 = \sigma_w^2 - \beta^2 + \left(\beta - k\sqrt{\sigma_w^2 + \sigma_v^2}\right)^2,$$

$$\sigma_1^2 = \sigma_w^2 - 2\beta k\sqrt{\sigma_w^2 + \sigma_v^2} + k^2(\sigma_w^2 + \sigma_v^2).$$

Consequently,

$$\beta = \frac{\sigma_w^2}{\sqrt{\sigma_w^2 + \sigma_v^2}},$$

$$\sigma_1^2 = \sigma_w^2 - \frac{\sigma_w^4}{\sigma_w^2 + \sigma_v^2} + \left(\frac{\sigma_w^2}{\sqrt{\sigma_w^2 + \sigma_v^2}} - k\sqrt{\sigma_w^2 + \sigma_v^2}\right)^2.$$

The minimum of σ_1^2 is obtained for

$$k = \frac{\sigma_w^2}{\sigma_w^2 + \sigma_v^2}.$$

The Value of this minimum is:

$$\sigma_1^2 = \sigma_w^2 - \frac{\sigma_w^4}{\sigma_w^2 + \sigma_v^2},$$

$$\sigma_1^2 = \frac{\sigma_w^2\sigma_v^2}{\sigma_w^2 + \sigma_v^2},$$

$$\frac{1}{\sigma_1^2} = \frac{1}{\sigma_w^2} + \frac{1}{\sigma_v^2}.$$

We note that when the variance σ_w^2 of the previous estimate is very large, the new estimate reduces to the new measure, and its variance is σ_v^2 : so, when initializing the Kalman filter, it is reasonable to start from an uncertain initial state with large covariance if there is no *prior* knowledge on the initial variance of th estimator of *m*.

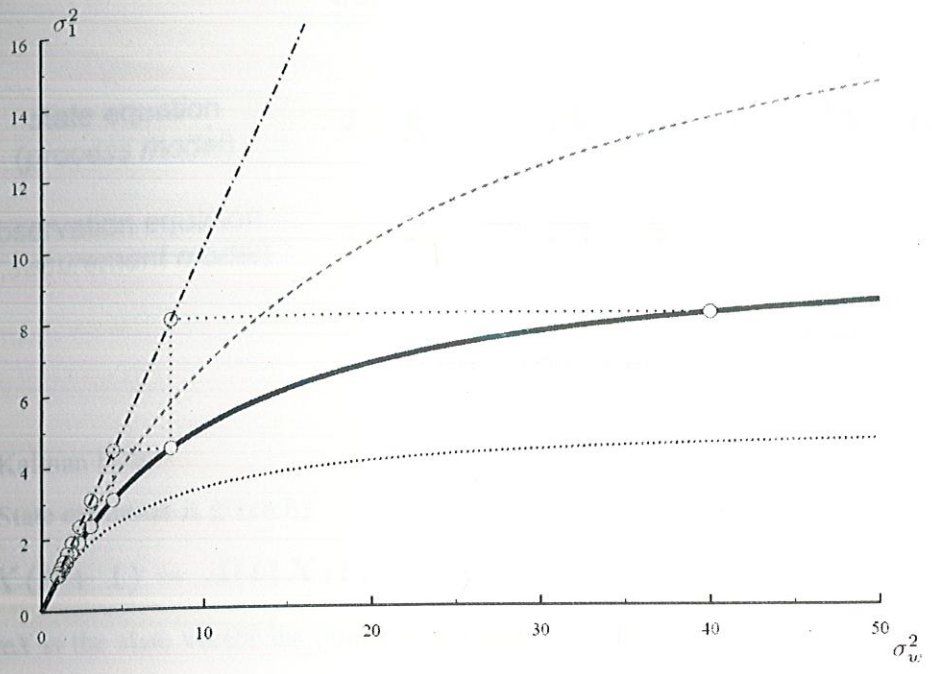


Fig 9:

Evolution de σ_1^2 en fonction de σ_w^2 for different values of σ_v^2 in the scalar case.

3.2.2 Discrete Kalman Filter Algorithm

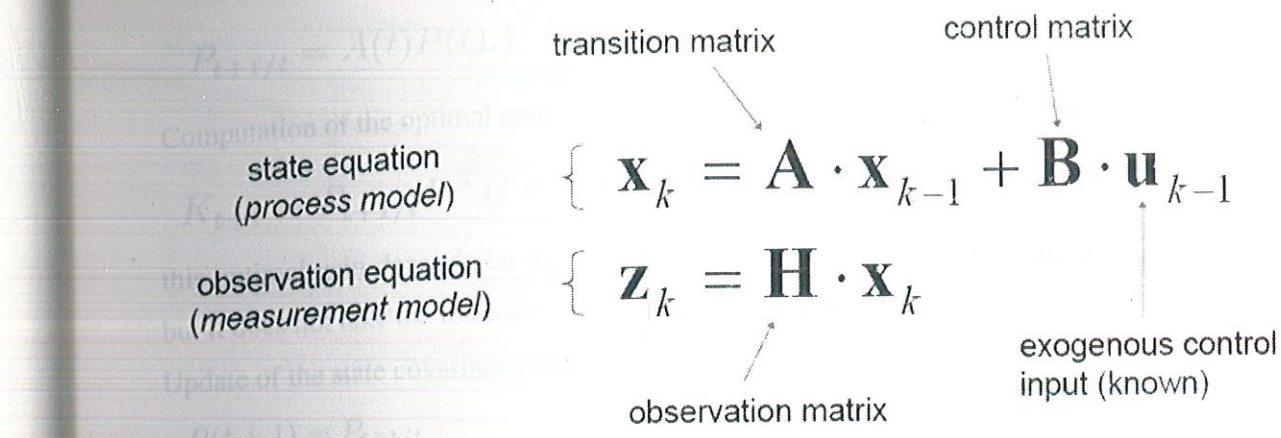
This section describes the filter in its original formulation (Kalman 1960) where the measurements are measured and state estimated at discrete points in time. The mathematical derivation will not be given here.

Dynamic systems (this is, systems which vary with time) are often represented in a state-space model.

State-space is a mathematical representation of a physical system as a set of inputs, outputs and state variables related linearly.

The Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by the models shown in the next slide

Discrete Time Controlled Process Model



The Kalman Filter

The State evolution is given by:

$$X(t+1) = A(t)X(t) + b(t) + w(t),$$

where$X$ is the state vector we intend to estimate, $A(t)$ is the known square transition matrix of the process. The control $b(t)$ is given and there is a zero mean process noise $w(t)$ with known covariance $r^{w}(t)$. This noise $w(t)$ is independent of $X(t)$.

The measured vector $y(t)$ is given by the measurement equation :

$$y(t) = H(t)X(t) + v(t).$$

$H(t)$ is the rectangular measurement matrix, $v(t)$ is the zero mean measurement noise, of known covariance $r^{v}(t)$. The noise $v(t)$ is independent of $X(t)$. The dimension of $w(t)$ is the dimension of $x(t)$; the dimension of $v(t)$ is the dimension of $y(t)$.

The covariance of the state vector $X(t)$ is:

$$P(t) = E\big[\,(X(t) - E[X(t)])\,(X^T(t) - E[X^T(t)])\,\big]$$

where$X^T$is the transpose (possibly conjugate) of X.

The purpose of the Kalman filter is to deduce from $y(t)$ the vector $X(t)$ whose covariance matrix has the lowest norm (its trace). The steps of the estimation are the following:

Prediction of the state $X(t)$:

$$X_{t+1/t} = A(t)X(t) + b(t);$$

Intermediate update of the state covariance matrix that takes into account the evolution given by the process transition :

$$P_{t+1/t} = A(t)P(t)A^T(t) + r^w(t);$$

Computation of the optimal gain :

$$K_{t+1} = P_{t+1/t}H^T(t+1)\big(H(t+1)P_{t+1/t}H^T(t+1) + r^v(t+1)\big)^{-1}$$

this optimal gain depends on the statistical characteristics of the measurement noise, but it does not take the measures into account : it may be computed *a priori*.

Update of the state covariance matrix :

$$P(t+1) = P_{t+1/t} - P_{t+1/t}H^T(t+1)\big(H(t+1)P_{t+1/t}H^T(t+1) + r^v(t+1)\big)^{-1}H(t+1)P_{t+1/t},$$

or, expressed as a function of K_{t+1}

$$P(t+1) = [I - K_{t+1}H(t+1)]P_{t+1/t};$$

Computation of the new estimate of the state :

$$X(t+1) = X_{t+1/t} + K_{t+1}[y(t+1) - H(t+1)X_{t+1/t}].$$

Dimensions

$\mathbf{x}_k$	$n \times 1$	– State vector
$\mathbf{u}_k$	$l \times 1$	– Input/control vector
$\mathbf{w}_k$	$n \times 1$	– Process noise vector
$\mathbf{z}_k$	$m \times 1$	– Observation vector
$\mathbf{v}_k$	$m \times 1$	– Measurement noise vector
$\mathbf{A}_k$	$n \times n$	– State transition matrix
$\mathbf{B}_k$	$n \times l$	– Input/control matrix
$\mathbf{H}_k$	$m \times n$	– Observation matrix
$\mathbf{Q}_k$	$n \times n$	– Process noise covariance matrix
$\mathbf{R}_k$	$m \times m$	– Measurement noise covariance matrix

Model and Algorithm

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \\ \mathbf{z}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \end{aligned}$$

Start:

Initialization:

$$\mathbf{x}_0^a = \mu_0 \text{ with error covariance } \mathbf{P}_0$$

While new data exists, do

1. Model Forecast Step/Predictor:

$$\begin{aligned} \mathbf{x}_k^f &= \mathbf{A}_{k-1}\mathbf{x}_{k-1}^a + \mathbf{B}_{k-1}\mathbf{u}_{k-1} \\ \mathbf{P}_k^f &= \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1} \end{aligned}$$

Data Assimilation Step/Corrector:

$$\begin{aligned} \mathbf{x}_k^a &= \mathbf{x}_k^f + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k^f) \\ \mathbf{K}_k &= \mathbf{P}_k^f\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^f\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f \end{aligned}$$

End while

End

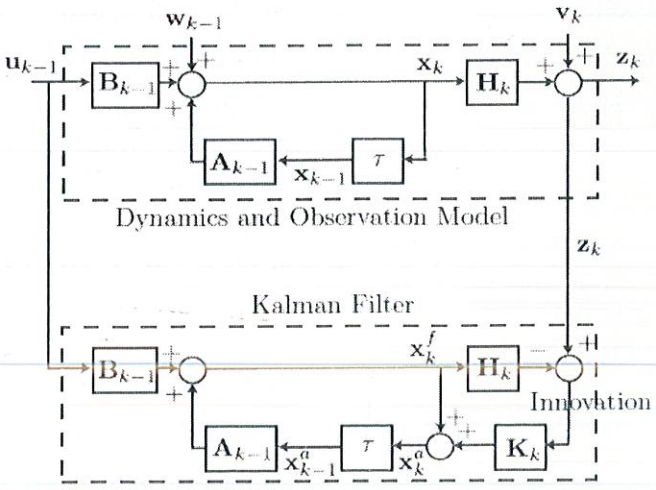


Fig 10: The Model

3.2.3 Properties & Remarks

The filter produce the error covariance matrix Pk which is an important estimate for the accuracy of the estimate.

The filter is optimal for Gaussian sequences only.

While the measurement noise covariance Rk is possible to be determined, the process noise covariance matrix Qk has to be computed to adjust to different dynamics. We are not able to directly observe the process we are estimating. Therefore a tuning on Qk has to be performed for superior filter performances.

3.3 Bayesian Algorithm

3.3.1 Introduction

In probability theory and statistics bayes' theorem is a result that is of importance in the mathematical manipulation of conditional probabilities. In particular, with the bayesian interpretation of probability the theorem expresses how a subjective degree of belief should rationally change to account for evidence: this is bayesian inference which is fundamental to bayesian statistics .However, Bayes' theorem has applications in a wide range of calculations involving probabilities.

Suppose someone told you they had a nice conversation with someone on the train. Not knowing anything else about this conversation, the probability that they were speaking to a woman is 50%. Now suppose they also told you that this person had long hair. It is now more likely they were speaking to a woman, since women are more likely to have long hair than men. Bayes' theorem can be used to calculate the probability that the person is a woman.

To see how this is done, let W represent the event that the conversation was held with a woman, and L denote the event that the conversation was held with a long-haired person. It can be assumed that women constitute half the population for this example. So, not knowing anything else, the probability that W occurs is $P(W) = 0.5$.

Suppose it is also known that 75% of women have long hair, which we denote as $P(L|W) = 0.75$ (read: the probability of event L given event W is 0.75). Likewise, suppose it is known that 25% of men have long hair, or $P(L|M) = 0.25$, where M is the

complementary event of W, i.e., the event that the conversation was held with a man (assuming that every human is either a man or a woman).

Our goal is to calculate the probability that the conversation was held with a woman, given the fact that the person had long hair, or, in our notation, P(W|L). Using the formula for Bayes' theorem, we have:

$$P(W|L) = \frac{P(L|W)P(W)}{P(L)} = \frac{P(L|W)P(W)}{P(L|W)P(W) + P(L|M)P(M)}$$

where we have used the law of total probability. The numeric answer can be obtained by substituting the above values into this formula. This yields

$$P(W|L) = \frac{0.75 \cdot 0.50}{0.75 \cdot 0.50 + 0.25 \cdot 0.50} = 0.75,$$

i.e., the probability that the conversation was held with a woman, given that the person had long hair, is 75%.

Another way to do this calculation is as follows. Initially, it is equally likely that the conversation is held with a woman as to a man. The prior odds on a woman versus a man are 1:1. The respective chances that a man and a woman have long hair are 75% and 25%. It is three times more likely that a woman has long hair than that a man has long hair. We say that the likelihood ratio or bayesfactor is 3:1. Bayes' theorem in odds form, also known as Bayes' rule, tells us that the posterior odds that the person was a woman is also 3:1 (the prior odds, 1:1, times the likelihood ratio, 3:1). In a formula:

$$\frac{P(W|L)}{P(M|L)} = \frac{P(W)}{P(M)} \cdot \frac{P(L|W)}{P(L|M)}.$$

Mathematically, Bayes' theorem gives the relationship between the probabilities of A and B, P(A) and P(B), and the conditional probabilities of A given B and B given A, P(A|B) and P(B|A). In its most common form, it is:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

The meaning of this statement depends on the interpretation of probability

3.3.2 Bayesian interpretation

In the bayesian interpretation probability measures a degree of belief. Bayes' theorem then links the degree of belief in a proposition before and after accounting for evidence. For example, suppose somebody proposes that a biased coin is twice as likely to land heads than tails. Degree of belief in this might initially be 50%. The coin is then flipped a number of times to collect evidence. Belief may rise to 70% if the evidence supports the proposition.

For proposition A and evidence B, P(A), the prior, is the initial degree of belief in A. P(A|B), the posterior, is the degree of belief having accounted for B. the quotient P(B|A)/P(B) represents the support B provides for A.

3.3.3 Frequents interpretation

In the frequents interpretation probability measures proportion of outcomes. For example, suppose an experiment is performed many times. P(A) is the proportion of outcomes with property A, and P(B) that with property B. P(B|A) is the proportion of outcomes with property Bout of outcomes with property A, and P(A|B) the proportion of those with A out of those with B.

The role of Bayes' theorem is best visualized with tree diagrams, as shown to the right. The two diagrams partition the same outcomes by A and B in opposite orders, to obtain the inverse probabilities. Bayes' theorem serves as the link between these different partitionings.

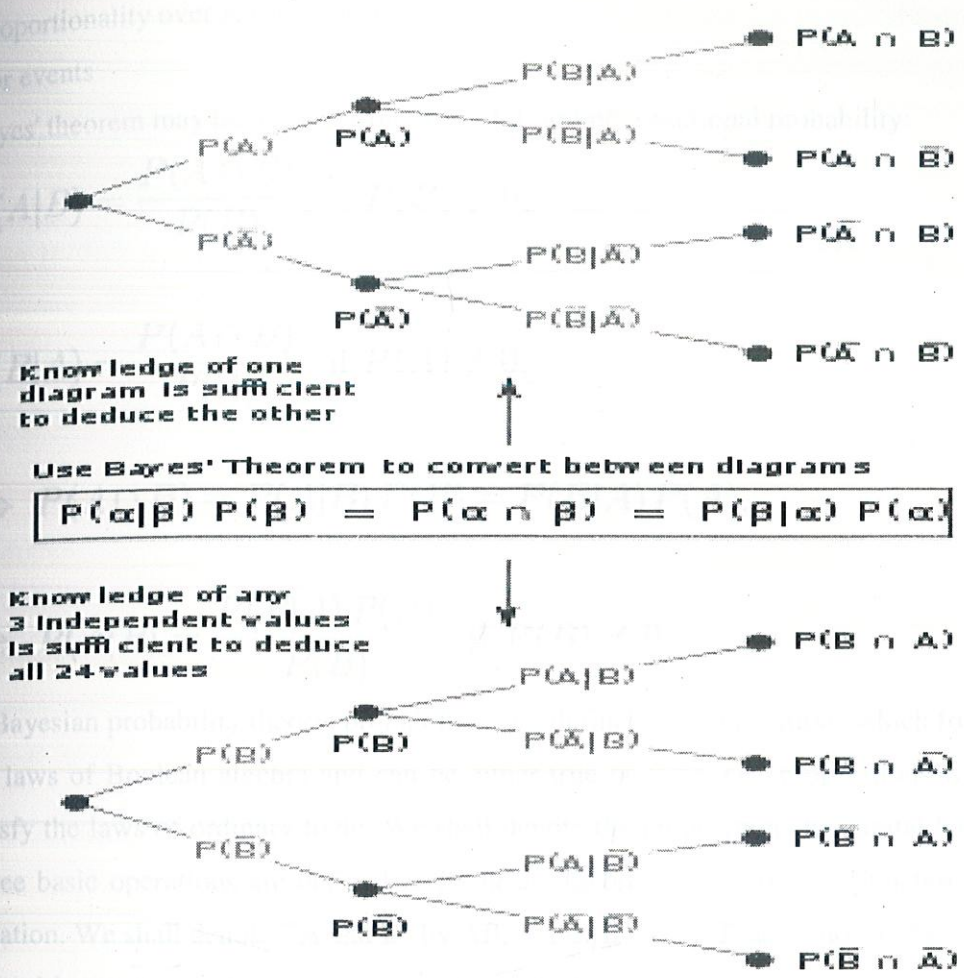


FIG 11:Bayesian Techniques

Events

For events A and B, provided that P(B) ≠ 0,

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

In many applications, for instance in Bayesian inference, the event B is fixed in the discussion, and we wish to consider the impact of its having been observed on our belief in various possible events A. In such a situation the denominator of the last expression, the probability of the given evidence B, is fixed; what we want to vary is A. Bayes theorem then shows that the posterior probabilities are propotional to the numerator:

$$P(A|B) \propto P(A) \cdot P(B|A)$$

(proportionality over A forgiven B).

For events

Bayes' theorem may be derived from the definition of conditional probability:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \text{ if } P(B) \neq 0,$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)}, \text{ if } P(A) \neq 0,$$

$$\Longrightarrow P(A \cap B) = P(A|B)\,P(B) = P(B|A)\,P(A),$$

$$\Longrightarrow P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}, \text{ if } P(B) \neq 0.$$

In Bayesian probability theory, probabilities are defined for propositions which follow the laws of Boolean algebra and can be either true or false, i.e., propositions which satisfy the laws of ordinary logic. We shall denote the propositions by capital letters. Three basic operations are defined in Boolean algebra: conjunction, disjunction and negation. We shall denote ``A and B" by AB, ``A or B" by A+B and ``not A" by $\neg A$ Natural language involves propositions whose truth value is ambiguous, ``sky is blue" for example. The definitions of words sky and blue are more or less ambiguous and therefore it is possible to think that the truth value of ``sky is blue" is neither completely true nor completely false but something in between. Fuzzy logic tries to capture the ambiguity of propositions in natural language, but we shall consider only unambiguously defined propositions.

3.4 Elementary rules of Bayesian probability theory

Sum rule:

$$P(A|B) + P(\neg A|B) = 1$$

Product rule:

P(AB | C) = P(A | C) P(B | AC)

Here P(A | B) denotes the probability of A on the condition that B is true. These rules correspond to the negation and conjunction operations of Boolean algebra. The

disjunction does not need a separate rule because it can be derived from negation and conjunction:

$$A + B = \neg(\neg A \neg B)$$

In fact, only one operation would suffice since other operations can be derived from either NAND or NOR operation alone. The NAND operation, for instance, yields the following rule, starting from which every other rule of Bayesian probability theory can be derived:

$$P(\neg A + \neg B|C) + P(A|C)P(B|AC) = 1$$

Here P(A | B) denotes the probability of A on the condition that B is true. These rules correspond to the negation and conjunction operations of Boolean algebra. The disjunction does not need a separate rule because it can be derived from negation and conjunction:

$$A + B = \neg(\neg A \neg B)$$

In fact, only one operation would suffice since other operations can be derived from either NAND or NOR operation alone. The NAND operation, for instance, yields the following rule, starting from which every other rule of Bayesian probability theory can be derived:

$$P(\neg A + \neg B|C) + P(A|C)P(B|AC) = 1$$

These rules fix the scale on which the degrees of belief are measured. Cox showed that under very general requirements of consistency and compatibility with common sense, the rules of calculus with beliefs have to be homomorphic with the sum and product rule. This means that one can measure the degrees of beliefs on any scale, but it is possible to transform the degrees of beliefs on the canonical scale of probabilities such that the rules for negation and conjunction take the form of the sum and product rule.

3.5 Marginalization principle

While Bayes' rule specifies how the learning system should update its beliefs as new data arrives, the marginalization principle provides for the derivation of probabilities of new propositions given existing probabilities. This is useful for

prediction and inference. Suppose the situation is the same as in the example with Bayes' rule, but now the learning system tries to compute the probability of making observation B before it has actually made the observation, that is, the learning system tries to predict the new observation. Suppose A1 and A2 are exhaustive and mutually exclusive propositions, in other words, exactly one of Ai is true while the rest are false. As before, assume that Ai are possible explanations for B and the prior assumptions and experience C are such that both P(B | AiC) and P(Ai | C) are determined. The marginalization principle then states the following:

$$P(B|C) = \sum_i P(A_i|C)P(B|A_iC) .$$

The probability of B thus depends on the prior probabilities P(Ai | C) of the different explanations and the probability P(B | AiC)which each explanation gives to B.

Notice also that P(B | C) appears in Bayes' rule, but the marginalization principle shows that it can be computed from P(Ai | C) and P(B | AiC) alone. Therefore P(Ai | C) and P(B | AiC) suffice for computing the posterior probability P(Ai | BC):

$$P(A_i|BC) = \frac{P(A_i|C)P(B|A_iC)}{\sum_j P(A_j|C)P(B|A_jC)} .$$

Beliefs alone are not sufficient for making decisions. Preferences are also needed. Decision theory points out how the beliefs and preferences should be combined when making decisions. We shall denote by U(A) the utility of proposition A. By definition, A is preferred over B if U(A) > U(B).

Decision theory can be summarized in a single rule:

$$U(A) = P(B|A)U(AB) + P(\neg B|A)U(A\neg B).$$

We shall call it the rule of expected utility. In case of mutually exclusive and exhaustive propositions , it generalizes into

$$U(A) = \sum_i P(B_i|A)U(AB_i) .$$

The significance of the rule becomes apparent if one considers A to be an action and Bi to be the possible consequences. Basically the rule indicates that the utility of A

depends on the utilities of the possible consequences of A, weighted by the probabilities of the consequences.

The sum and product rules of probability theory fix the scale by which degrees of beliefs are measured to be the canonical scale of probabilities. The rule of expected utility does the same for utilities, up to linear scaling and an additive constant. This is because the beliefs have clear limits in absolute belief and disbelief while there are no absolutely worst or best possible states of the world, or if there are, the difference of their utility is probably infinitely greater than the difference between any other states of the world.

We can now summarize what Bayesian probability theory and decision theory say about learning, reasoning and action by giving a simple example. Suppose there are prior assumptions and experience I and possible explanations expressed as states of the world Si. An observation D is made and an action Aj is chosen based on the belief about what is the consequence D' of the action. We assume D' is one of several possible observations D'k made after the action is chosen.

The prior assumptions and experience I are assumed to be such that it is possible to determine the prior probability P(Si | I) of each state of the world; the probability P(D | SiI) of observation D given the state of the world Si; the probabilities P(D'k | SiAjD I) of different consequences of actions given the state of the world and prior experience; and the utility of the consequences U(AjD'kD I). The action Aj is assumed to have no effect on the state Si of the world and thus P(Si | AjD I) = P(Si | D I).

First the states of the world have prior probabilities P(Si | I). After making the observation D, the probabilities change according to Bayes' rule:

$$P(S_i|DI) = \frac{P(S_i|I)P(D|S_iI)}{\sum_{i'} P(S_{i'}|I)P(D|S_{i'}I)}.$$

The belief in those states of the world which were able to predict the observation better than average increases, and vice versa.

The next stage is to infer which consequences different actions have. According to the marginalisation principle,

$$P(D'_k|A_jDI) = \sum_i P(S_i|A_jDI)P(D'_k|S_iA_jDI).$$

Notice that Aj was assumed to have no effect on Si and thus P(Si | AjD I) is equal to the posterior probability P(Si | DI) which was computed in the first stage.

The third stage of the example is choosing an action which has the greatest utility. The utilities can be computed by the rule of expected utility:

$$U(A_j DI) = \sum_k P(D'_k | A_j DI) U(A_j D'_k DI).$$

The utilities of actions are based on the utilities of consequences and the probabilities of consequences in light of the experience, which were computed in the previous stage.

So far we have explicitly denoted that the probabilities are conditional to the prior assumptions and experience I. In most cases the context will make it clear which are the prior assumptions and usually I is left out. This means that probability statements like P(Si) should be understood to mean P(Si | I) where Idenotes the assumptions appropriate for the context.

3.6 Probability density for real valued variables

In symbolic representations, the propositions are discrete and similar to simple statements of natural language. When trying to learn models of the environment, the problem with discrete propositions is that an unimaginable number of them is needed for covering all the possible states of the world. The alternative is to build models which have real valued variables. This allows one to manipulate a vast number of elementary propositions by manipulating real valued functions, probability densities. Following the usual Bayesian convention, probability density is denoted by a lower case p and the ordinary probability by a capital P throughout this thesis. We also use the convenient short hand notation where p(x | y) means the distribution of the belief in the value of x given y. Alternative notation would be fXIY(x | y), which makes explicit the fact that p(x | y) is not the same function as, for instance, p(u | v). In cases where the ordinary probability needs to be distinguished from probability density, it is called probability mass in analogy to physical mass and density.

Bayes' rule looks exactly the same for probability densities as it does for probability mass. If a and b are real valued variables, Bayes' rule takes the following form:

$$p(a|bC) = \frac{p(a|C)p(b|aC)}{p(b|C)}.$$

observations. For the real valued latent variable models considered in this thesis, the MAP estimators cannot be used as such. The models include products of unknown quantities, weights and factors in this case, which means that by increasing the value of one variable, the value of another variable can be decreased. This scaling does not change the model but the density of the first variable decreases and the second value increases. For each observation, a new set of values is estimated for factors which means that typically the number of unknown values for factors is far greater than the number of unknown values for weights. If the MAP estimates were used for factor analysis models, the result would be that the weights of the model would grow and the values of the factors shrink. The resulting low density of the weights would be overwhelmed by the high density of the factors. In other words, MAP estimation would find the values of the weights which give the highest density for the factors, but would not say much about the posterior probability mass of the model because the high density would be obtained at the cost of narrow posterior peaks of the factors.

In any case, the use of a point estimate will cause a phenomenon called over fitting. Most people are familiar with the concept at least in the context of fitting polynomials to observations. Using only the best model means being excessively confident that the best fit is the correct one. In the case of probability densities, for instance, all probability mass is in models which have a poorer fit than the best model. This means that the optimal prediction based on the full posterior density will necessarily be less confident about the fit than a prediction based on only the ``best'' model.

3.7 Single sensor tracking

As a first example of data fusion, we apply Bayes' rule to tracking. Single sensor tracking,also known as filtering, involves a combining of successive measurements of the state of a system, and as such it can be thought of as a fusing of data from a single sensor over time as opposed to sensor set, which we leave for the next section. Suppose then that a sensor is tracking a target, and makes observations of the target at various intervals. Define the following terms:

xk = target state at "time" k (iteration number k);

yk = observation made of target at time k;

Yk = set of all observations made of target up to time k = {y1 , y2 , . . . , yk }.

The fundamental problem to be solved is to find the new estimate of the target state (xk |Yk) given the old estimate (xk−1 |Yk−1). That is, we require the probability that the target is some- thing specific given the latest measurement and all previous measurements, given that we know the corresponding probability one time step back. To apply Bayes' rule for the set Yk , we separate the latest measurement yk from the rest of the set Yk−1 (since Yk−1 has already been used in the previous iteration) to write (xk |Yk) as (xk |yk , Yk−1). We shall swap the two terms xk ,yk using a minor generalization of Bayes' rule. This generalization is easily shown by equating the probabilities for the three events (A, B, C) and (B, A, C) expressed using conditionals as in :

(A, B, C) = (A|B, C) (B|C) (C);

(B, A, C) = (B|A, C) (A|C) (C);

so that Bayes' rule becomes

(A|B, C) = (B|A, C) (A|C) /(B|C)

Before proceeding, we note that since only the latest time k and the next latest k − 1 appear

in the following expressions, we can simplify them by replacing k with 1 and k − 1 with 0.

So we write

"likelihood" "predicted density"

"conditional density"

(x1 |Y1)= (x1 |y1 , Y0) = (y1 |x1 , Y0) (x1 |Y0) (y1 |Y0)

normalization There are three terms in this equation, and we consider each in turn. There are three terms in this equation, and we consider each in turn.The likelihood deals with the probability of a measurement y1 . We will assume the noise is "white", meaning uncorrelated in time,1 so that the latest measurement does not depend on previous measurements. In that case the likelihood (and hence normalization) can be simplified: likelihood = (y1 |x1 , Y0) = (y1 |x1). Fusing data from several sensors Centralizing the fusion combines all of the raw data from the sensors in one main processor. In principle this is the best way to fuse data in the sense that nothing has been lost in preprocessing; but in practice centralized fusion leads to a huge amount of data traversing the network, which is not necessarily practical or desirable. Preprocessing the data at each sensor reduces the amount of data flow needed, while

in practice the best setup might well be a hybrid of these two types. Bayes' rule serves to give a compact calculation for the fusion of data from several sensors.

Extend the notation from the previous section, with time as a subscript, by adding a superscript to denote sensor number: Single sensor output at indicated time step = Y sensor number time step all data up to and including time step =X sensor number time step.

CHAPTER 4 ANDROID GPS LOCATOR

4.1 Introduction

Android is linux-based operating system. Android consists of a kernel based on linux kernel, with middleware, libraries and API's written in C and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Android uses the dalvil virtual machine with just in time compilation to run Dalvik 'dex-code' (Dalvik Executable), which is usually translated from java bytecode. The main hardware platform for Android is the ARM architecture. There is support for x86 from the android-x86 project, and Google TV uses a special x86 version of Android.

In general, the user's pinpointing on the map is feasible in one of the following ways:

Using the GPS device that comes with the mobile

Using the ID of the Cell that the user is currently served by these days a big number of phones do have GPS devices onboard, we used the first way.

The Android SDK's emulator can emulate changes in the sensor location and provide dummy data for its coordinates.

4.2 Starting the project

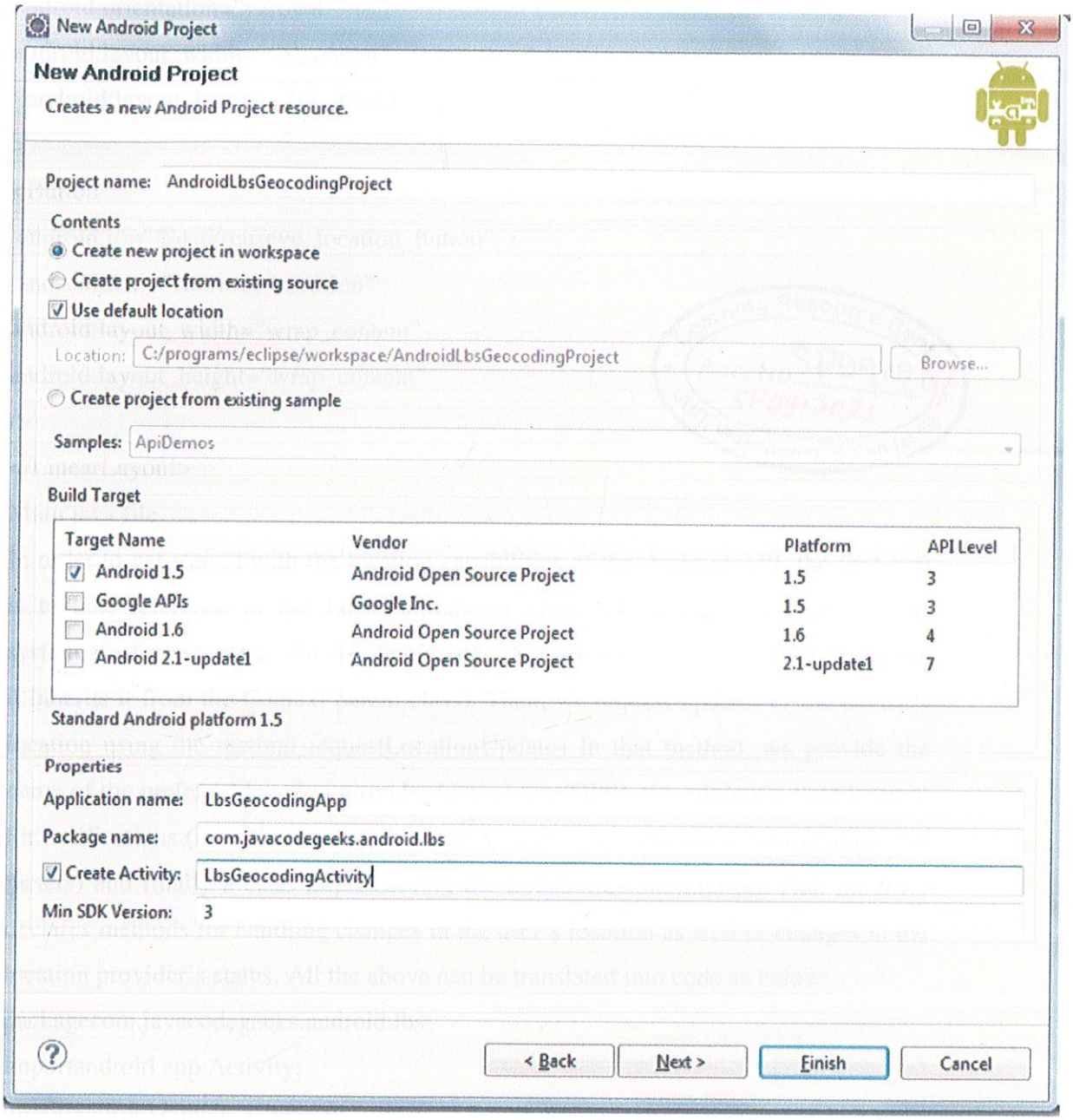


Fig 12: The Start

Xml file

To start with, we used only add a button which will trigger the retrieval of the current location's coordinates from a location provider. Thus, the "main.xml" file for the application's interface will be as simple as this:

```
importandroid.widget.Toast;
public class LbsGeocodingActivity extends Activity {
privatestaticfinallongMINIMUM_DISTANCE_CHANGE_FOR_UPDATES = 1;
private static final long MINIMUM_TIME_BETWEEN_UPDATES = 1000;
protectedLocationManagerlocationManager;
protected Button retrieveLocationButton;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
retrieveLocationButton = (Button) findViewById(R.id.retrieve_location_button);
locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
locationManager.requestLocationUpdates(
LocationManager.GPS_PROVIDER,
MINIMUM_TIME_BETWEEN_UPDATES,
MINIMUM_DISTANCE_CHANGE_FOR_UPDATES,
newMyLocationListener()
);
retrieveLocationButton.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
showCurrentLocation();
showCurrentLocation();
});
}
protectedvoidshowCurrentLocation() {
Location location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
if (location != null) {
String message = String.format(
"Current Location \n Longitude: %1$s \n Latitude: %2$s",
location.getLongitude(), location.getLatitude()
);
```

```
Toast.makeText(LbsGeocodingActivity.this, message,
Toast.LENGTH_LONG).show();
}
}
private class MyLocationListener implements LocationListener {
public void onLocationChanged(Location location)
String message = String.format(
"New Location \n Longitude: %1$s \n Latitude: %2$s",
location.getLongitude(), location.getLatitude()
);
Toast.makeText(LbsGeocodingActivity.this,message,
Toast.LENGTH_LONG).show();
}
public void onStatusChanged(String s, inti, Bundle b) {
Toast.makeText(LbsGeocodingActivity.this,
"Provider disabled by the user. GPS turned off",
Toast.LENGTH_LONG).show();
}
public void onProviderEnabled(String s) {
Toast.makeText(LbsGeocodingActivity.this,
"Provider enabled by the user. GPS turned on",
Toast.LENGTH_LONG).show();
}
}
}
```

For the LocationListner interface, we implemented the MyLocationListener inner class. The methods in that class just use Toats to provide info about the GPS status or any location changes. The only interface element is a Button which gets hooked up with aOmClickListner and when it is clicked, the showCurrentLocation method is invoked. Then, the getLstKnownLocation of the LocationManager instance is executed returning the last known Location. From a Location object we can get information regarding the user's altitude, latitude, longitude, speed etc. In order to be able to run the above code, the necessary permissions have to be granted.

AndroidManifest.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package="com.javacodegeeks.android.lbs"
android:versionCode="1"
 android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".LbsGeocodingActivity"
 android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<categoryandroid:name="android.intent.category.LAUNCHER" />
 </intent-filter>
  </activity>
</application>
<uses-permission  android:name="android.permission.ACCESS_FINE_LOCATION"
/>
 <uses-permission
android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-sdkandroid:minSdkVersion="3" />
</manifest>
```

use the AVD manager in order to create a new Android device and make sure that GPS support is included in the features:

Using AVD manager

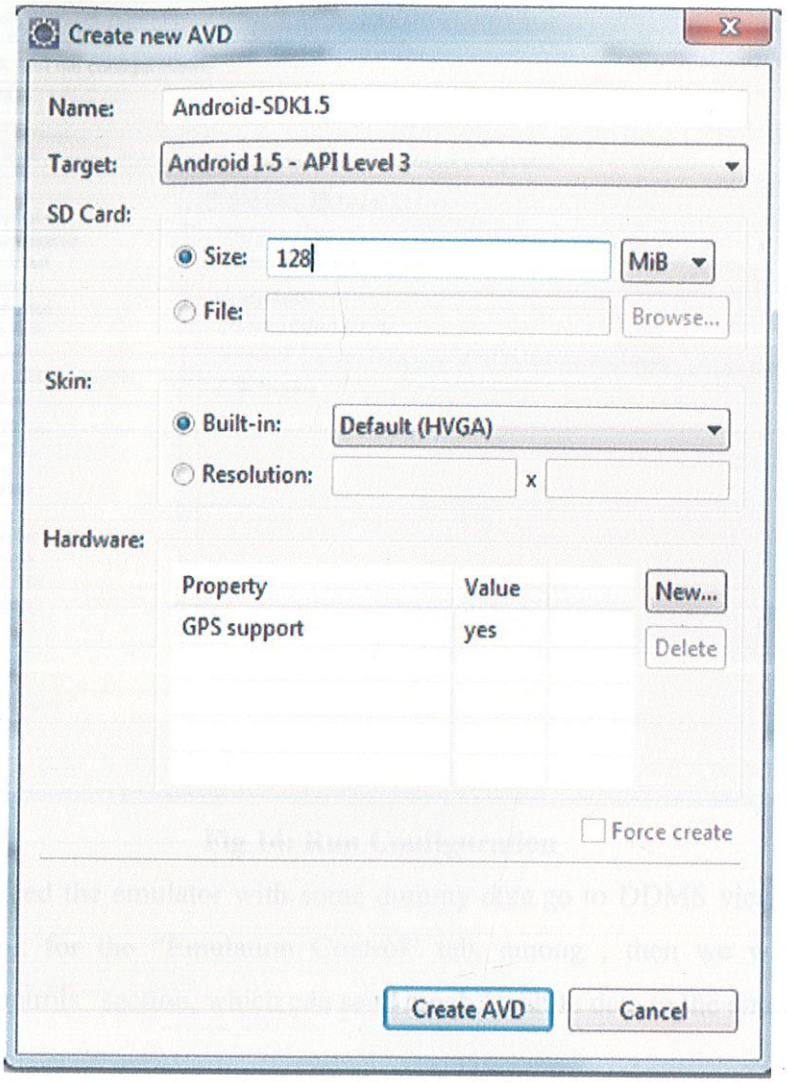


Fig 13: AVD Manager

Next, use Run–> Run Configurations

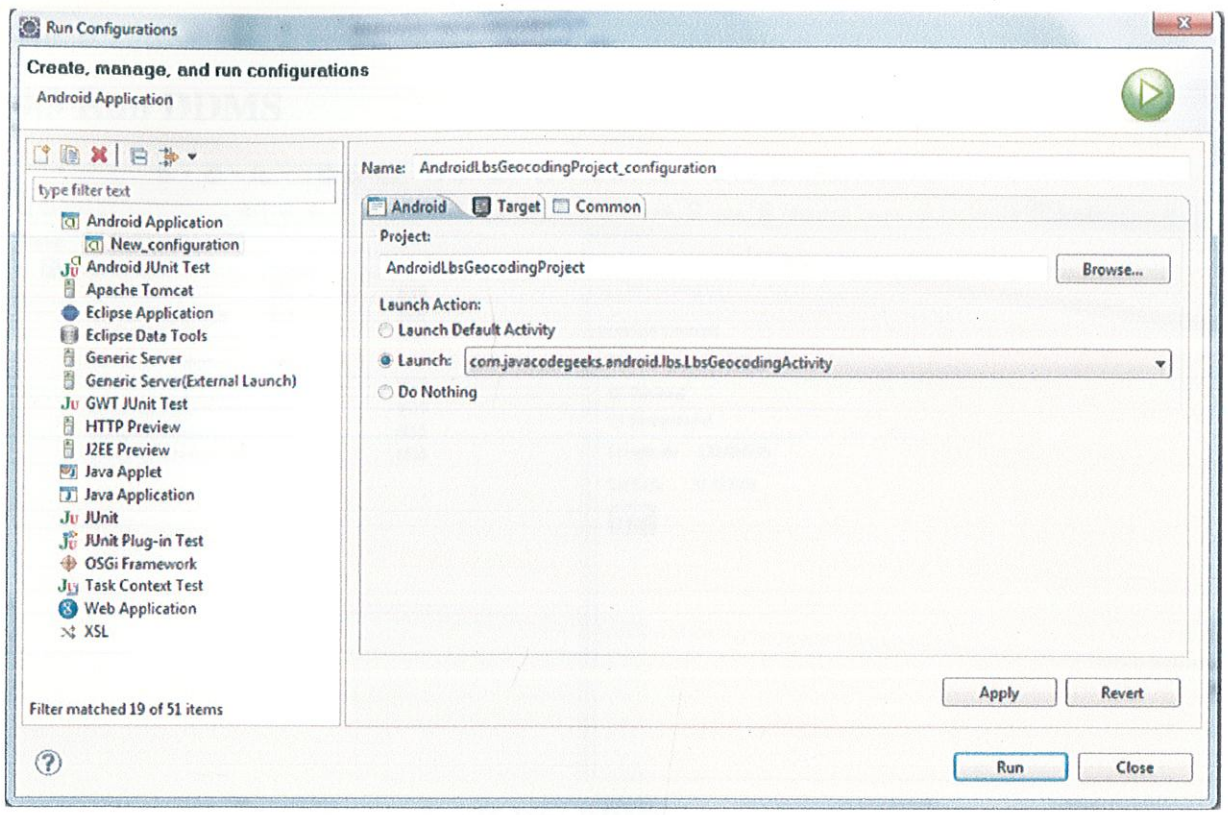


Fig 14: Run Configuration

We have to feed the emulator with some dummy data.go to DDMS view of Eclipse and then look for the "Emulation Control" tab. among , then we will find the "Location Controls" section, which can send mock location data to the emulator.

4.3 Run DDMS

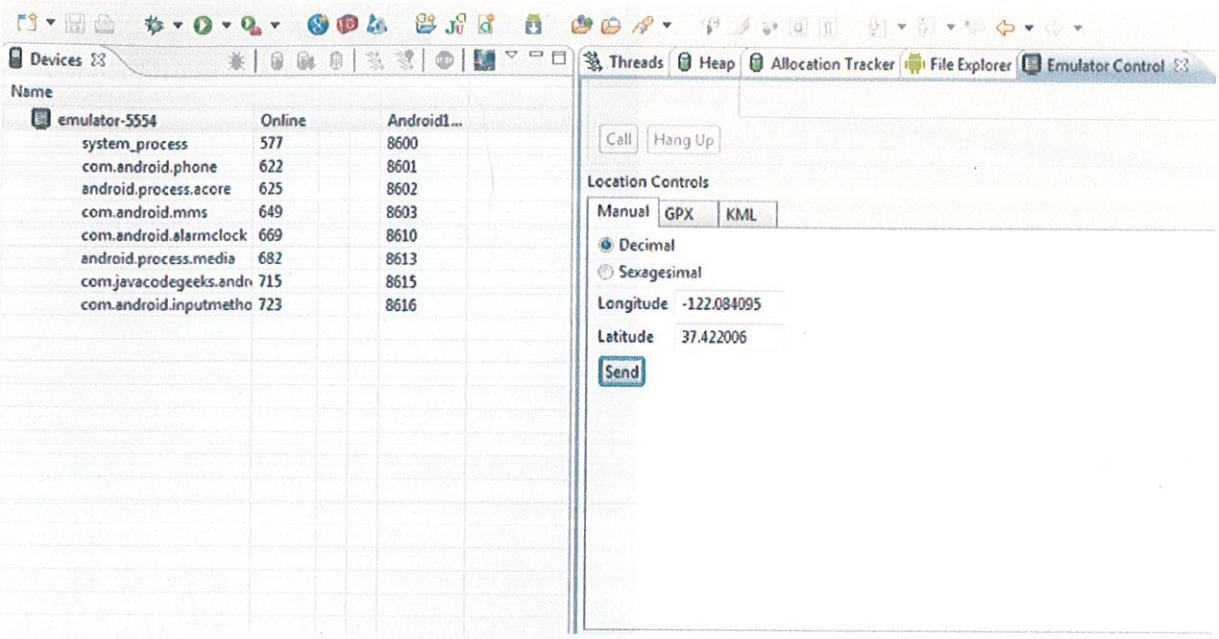


Fig 15 Run DDMS

When the data are sent to the emulator's GPS device, our listener will be triggered and the current location will be printed in the screen

4.4 Emulator screen

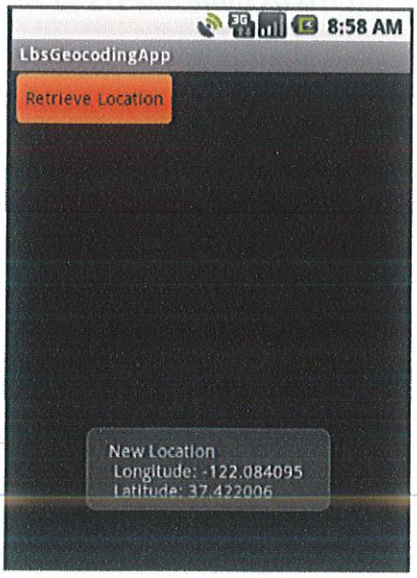


Fig 16 Emulator Screen

4.4 Retrieve Location

Used to retrieve last location

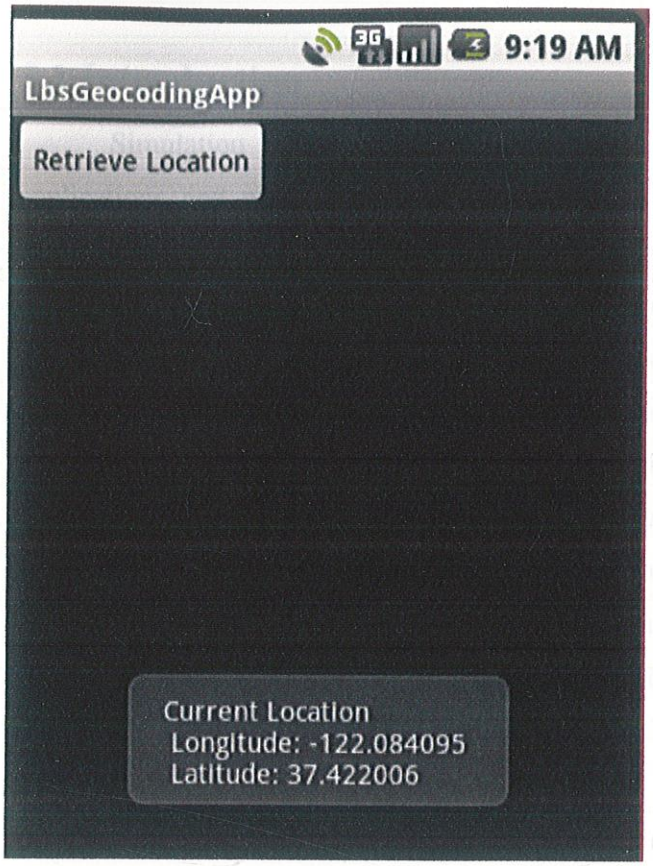


Fig 17: Location Retrieval

CHAPTER 5: SIMULATION AND RESULTS

5.1 Introduction

Simulation is essential to study WSN, being the common way to test new applications and protocols in the field. This fact has brought a recent boom of simulation tools available to model WSN. However, obtaining reliable conclusions from research based on simulation is not a trivial task. There are two key aspects that should be evaluated before conducting experiments: (1) The correctness of the model and (2) the suitability of a particular tool to implement the model. A tool that helps to build a model is needed, and the user faces the task of selecting the appropriate one. Simulation software commonly provides a framework to model and reproduce the behaviour of real systems. However, actual implementation and "secondary goals" of each tool differ considerably, that is, some may be designed to achieve good performance and others to provide a simple and friendly graphical interface or emulation capabilities.

Sensor networks have been developed and deployed in various civilian applications. Research fields in this area include increasing the potential of hardware components in terms of smaller size and less energy consumption, operating system, protocol and application. Traditionally, each work has to be tested and evaluated to ensure achievement of the predefined objectives. There are several ways to determine results such as small-scale experiment in a lab, wide-area testbed and custom simulation. Unlike traditional networks, sensor networks are application-specific and may compose thousands of resource-constrained sensors. Therefore, conducting a simulation seems to be the most efficient way to obtain a preliminary result. Several simulators are described in this section.

5.2 Network Simulating Motivation

Networking plays a major role in today's communication worldwide. Several underlying technologies become involved and are continually evolving. The Internet is one of the obvious phenomena reflecting the profound networking technology that has changed the way of how people communicate.

The number of the Internet users is exponentially increasing and the trends in application use are becoming hard to predict. Newly developed application software and protocols may be widely-used even in couple months. Further, heterogeneity in topology, link properties, and protocols [22] are obstacles to fully understanding how the Internet works.

The Virtual InterNetwork Testbed (VINT) project was initiated and performed by USC/ISI, XeroxPARC, LBNL and UCB. The objectives of this project are to develop methods and tools to study protocol operations in both high level including interaction and scaling, and low levels such as congestion control, reliable multicast, multicast routing and dynamic topologies. Finally, an increase in the quality of analysis and rate of progress in protocol development can be achieved by providing a common simulation infrastructure.

5.2.1 Requirements for Simulators

Simulating wireless sensor networks requires more specific properties to reflect the real operational environment. This section provides both general and specific requirements to effectively simulate wireless sensor networks. Non-functional and functional requirements are the general requirements which each simulator should address. Specific requirements are such characteristics needed to test and evaluate the new generic, lightweight and reliable transport protocol for wireless sensor networks.

Non-functional Requirements

The non-functional requirements are:

- Open Source – Publicly available simulators allow various users to freely develop their own contributed modules. This supports rapid enhancement, but bug reports and contributing report forms are required to keep the developing information updated.

- Platform Independence – A simulator should support all sensor platforms. An ability to simulate different platforms would support a wide range of sensors developed by various communities.

- Visualisation Module – A friendly visualised and/or animated environment displaying results should be provided. For example, node mobility, data packet and energy level should be displayed at different timeline. This could promote better understanding and interpretation of the result.

Functional Requirements

Several functional requirements are provided:

- Hardware Simulation Coverage – A network simulator should be capable of simulating all the hardware of a sensor such as CPU, transceiver and sensing unit. This could reflect performance of each component and also interaction amongst them.

- Battery and Power Models – Resource constraint is still a major drawback of sensor networks. All of the energy comes from a tiny battery and each operation needs different energy levels. With both models, a picture of energy consumption and energy depletion can be seen and forecasted.

Propagation Modeling – All reviewed simulators support only radio frequency (RF) modeling. This may be because this medium is currently the most widely used by deployed sensor networks. However, other models such as optical communication (laser) and infrared should be also determined to support various requirements.

- Protocols Modeling – A large number of protocols at different network layers have been developed. It is impossible to include all protocols but it should include all such common Internet protocols such as TCP and UDP. Current routing protocols for ad hoc mobile wireless networks such as Destination Sequenced Distance Vector (DSDV), Ad-hoc On Demand Vector (AODV), Dynamic Source Routing (DSR) and Temporally Ordered Routing Algorithm (TORA) should also be included. However, providing a convenient API for defining new protocol in a simulator would be another effective approach.

- Physical Environment Modeling – Sensors will be scattered in/on various placing areas such as soil, water, cement surface or human body. Different physical areas have different signal propagation characteristics. Radio signal changes when travelling through different physical media.

- Emulation – There are two approaches to address such deficiencies of simulation through real-world interaction including network and environment emulations [28]. In the network emulation approach, simulated entities are facilitated to communicate with the real-world entities such as protocol implementation. Another approach, the environment emulation, an implementation of the real-world entities is built in order to be directly executed within the simulator. This requirement will then promote better understanding of the network behavior. More accurate results will also be obtained.

5.3 Network Model

The following components are considered:

1) Nodes: Each node is a physical device monitoring a set of physical variables. Nodes communicate with each other via a common radio channel. Internally, a protocol stack controls communications. Unlike classical network models, sensor modes include a second group of components: The physical node tier, which is connected to the environment. Nodes are usually positioned in a two or three

dimensional world. An additional "topology" component, may control node coordinates. Depending on the application and deployment scenario, a WSN can contain from a few to several thousands of nodes.

2) Environment: The main difference between classical and WSN model is the additional "environment" component. This component models the generation and propagation of events that are sensed by the nodes, and trigger sensor actions, i.e. communication among nodes in the network. The events of interest are generally a physical magnitude as sound or seismic waves or temperature.

3) Radio channel: It characterizes the propagation of radio signals among the nodes in the network. Very detailed models use a "terrain" component, connected to the environment and radio channel components. The terrain component is taken into consideration to compute the propagation as part of the radio channel, and also influences the physical magnitude.

4) Sink nodes: These are special nodes that, if present, receive data from the net, and process it. They may interrogate sensors about an event of interest. The use of sinks depends on the application and the tests performed by the simulator.

5) Agents: A generator of events of interest for the nodes. The agent may cause a variation in a physical magnitude, which propagates through the environment and stimulates the sensor. This component is useful when its behaviour can be implemented independently from the environment, e.g., a mobile vehicle. Otherwise, the environment itself can generate events.

5.4 The Network Simulator – ns-2

Amongst the existing network simulators, ns-2 is an open-source, discrete-event simulator and is one of the most widely used tools in the networking research community. It was developed within the VINT project in 1995 which attempted to provide an efficient simulation tool to facilitate new protocol design. Current ns-2 users come from several universities and research communities. It also provides much useful information on its website such as downloading and installation guides, examples, tutorials and on-line manuals, development help, and mailing lists.

The ns-2 includes several common protocols for the Internet such as TCP and UDP. Moreover, network emulation is included in ns-2 to simulate real-world interaction.

5.5 Implementation and Code

The **NS-2** simulation environment offers great flexibility in investigating the characteristics of sensor networks because it already contains flexible models for energy-constrained wireless ad hoc networks. In the NS-2 environment, a sensor network can be built with many of the same sets of protocols and characteristics as those available in the real world. The mobile networking environment in NS-2 includes support for each of the paradigms and protocols shown in Fig. 2. The wireless model also includes support for node movements and energy constraints. By leveraging the existing mobile networking infrastructure, we added the capability to simulate sensor networks.

OTcL- NS is basically an OTcl interpreter with network simulation object libraries. It is very useful to know how to program in OTcl to use NS. In Tcl, the keyword **proc** is used to define a procedure, followed by an procedure name and arguments in curly brackets. The keyword **set** is used to assign a value to a variable. [**expr** ...] is to make the interpreter calculate the value of expression within the bracket after the keyword. One thing to note is that to get the value assigned to a variable, $ is used with the variable name. The keyword **puts** prints out the following string within double quotation marks. Below is an example of simple OTcl script along with the output.

NAM- Nam is a Tcl/Tk based animation tool for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation, and various data inspection tools.

The network animator ``nam'' began in 1990 as a simple tool for animating packet trace data. This trace data is typically derived as output from a network simulator like ns or from real network measurements, e.g., using tcpdump

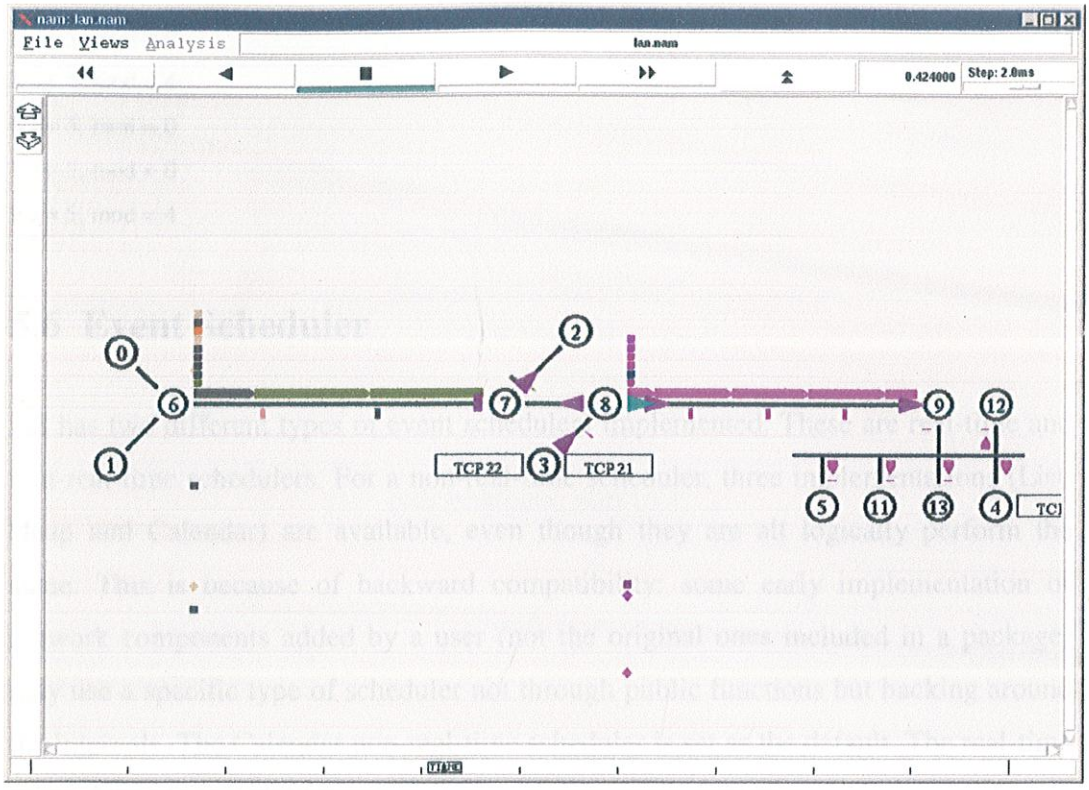


Fig 19: Snapshot 1

```
# Writing a procedure called "test"
proc test {} {
   set a 43
   set b 27
   set c [expr $a + $b]
   set d [expr [expr $a - $b] * $c]
   for {set k 0} {$k < 10} {incr k} {
         if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
         } else {
            puts "k >= 5, mod = [expr $d % $k]"
         }
   }}
# Calling the "test" procedure created above
Test
```

OUTPUT :

k < 5, pow = 1.0

k < 5, pow = 1120.0

k < 5, pow = 1254400.0

k < 5, pow = 1404928000.0

k < 5, pow = 1573519360000.0

k >= 5, mod = 0

k >= 5, mod = 4

k >= 5, mod = 0

k >= 5, mod = 0

k >= 5, mod = 4

5.6 Event Scheduler

NS has two different types of event schedulers implemented. These are real-time and non-real-time schedulers. For a non-real-time scheduler, three implementations (List, Heap and Calendar) are available, even though they are all logically perform the same. This is because of backward compatibility: some early implementation of network components added by a user (not the original ones included in a package) may use a specific type of scheduler not through public functions but hacking around the internals. The Calendar non-real-time scheduler is set as the default. The real-time scheduler is for emulation, which allow the simulator to interact with a real network. Currently, emulation is under development although an experimental version is available. The following is an example of selecting a specific event scheduler:

```
.       .       .
set ns [new Simulator]
$ns use-scheduler Heap
...
```

Another use of an event scheduler is to schedule simulation events, such as when to start an FTP application, when to finish a simulation, or for simulation scenario generation prior to a simulation run. An event scheduler object itself has simulation scheduling member functions such as at *time "string"* that issue a special event called AtEvent at a specified simulation *time*. An "AtEvent" is actually a child class of "Event", which has an additional variable to hold the given *string*. However, it is treated the same as a normal (packet related) event within the event scheduler. When a simulation is started, and as the scheduled time for an AtEvent in the event queue comes, the AtEvent is passed to an "AtEvent handler" that is created once and handles all AtEvents, and the OTcl command specified by the *string* field of the AtEvent is

executed. The following is a simulation event scheduling line added version of the above example.

```
...
set ns [new Simulator]
$ns use-scheduler Heap
$ns at 300.5 "complete_sim"
...

proc complete_sim { } {
...
}
```

Followings are a partial list and brief description of Simulator object member functions that interface with scheduler member functions:

```
Simulator instproc now          # return scheduler's notion of current time
Simulator instproc at args      # schedule execution of code at specified time
Simulator instproc at-now args  # schedule execution of code at now
Simulator instproc after n args # schedule execution of code after n secs
Simulator instproc run args     # start scheduler
Simulator instproc halt         # stop (pause) scheduler
```

5.7 NS Simulation:

This section shows a simple NS simulation script and explains what it does.

5.7.1 Source Code

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
```

```
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}


#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
```

```
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
```

```
$cbr set rate_ 1mb
$cbr set random_ false


#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

This network consists of 4 nodes (n0, n1, n2, n3) as shown in the output. The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "tcp" agent is attached to n0, and a connection is established to a tcp "sink" agent attached to n3. As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "udp" agent that is attached to n1 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "ftp" and a "cbr" traffic generator are attached to "tcp" and "udp" agents

respectively, and the "cbr" is configured to generate 1 KByte packets at the rate of 1 Mbps. The "cbr" is set to start at 0.1 sec and stop at 4.5 sec, and "ftp" is set to start at 1.0 sec and stop at 4.0 sec.

The following is the explanation of the script above. In general, an NS script starts with making a Simulator object instance.

- set *ns* [new Simulator]: generates an NS simulator object instance, and assigns it to variable *ns* (italics is used for variables and values in this section). What this line does is the following:

    - Initialize the packet format (ignore this for now)
    - Create a scheduler (default is calendar scheduler)
    - Select the default address format (ignore this for now)

    The "Simulator" object has member functions that do the following:

    - Create compound objects such as nodes and links (described later)
    - Connect network component objects created (ex. attach-agent)
    - Set network component parameters (mostly for compound objects)
    - Create connections between agents (ex. make connection between a "tcp" and "sink")
    - Specify NAM display options

Most of member functions are for simulation setup (referred to as plumbing functions in the Overview section) and scheduling, however some of them are for the NAM display. The "Simulator" object member function implementations are located in the "ns-2/tcl/lib/ns-lib.tcl" file.

- *$ns* color *fid color*: is to set color of the packets for a flow specified by the flow id (fid). This member function of "Simulator" object is for the NAM display, and has no effect on the actual simulation.

- *$ns* namtrace-all *file-descriptor*: This member function tells the simulator to record simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command *$ns* flush-trace. Similarly, the member function trace-all is for recording the simulation trace in a general format.
- proc *finish* { }: is called after this simulation is over by the command *$ns* at 5.0 "*finish*". In this function, post-simulation processes are specified.
- set *n0* [*$ns* node]: The member function node creates a node. A node in NS is compound object made of address and port classifiers (described in a later section). Users can create a node by separately creating an address and a port classifier objects and connecting them together. However, this member function of Simulator object makes the job easier. To see how a node is created, look at the files: "ns-2/tcl/libs/ns-lib.tcl" and "ns-2/tcl/libs/ns-node.tcl".
- *$ns* duplex-link *node1 node2 bandwidth delay queue-type*: creates two simplex links of specified bandwidth and delay, and connects the two specified nodes. In NS, the output queue of a node is implemented as a part of a link, therefore users should specify the queue-type when creating links. In the above simulation script, DropTail queue is used. If the reader wants to use a RED queue, simply replace the word DropTail with RED. The NS implementation of a link is shown in a later section. Like a node, a link is a compound object, and users can create its sub-objects and connect them and the nodes. Link source codes can be found in "ns-2/tcl/libs/ns-lib.tcl" and "ns-2/tcl/libs/ns-link.tcl" files. One thing to note is that you can insert error modules in a link component to simulate a lossy link (actually users can make and insert any network objects). Refer to the NS documentation to find out how to do this.
- *$ns* queue-limit *node1 node2 number*: This line sets the queue limit of the two simplex links that connect node1 and node2 to the number specified. At this point, the authors do not know how many of these kinds of member functions of Simulator objects are available and what they are. Please take a look at "ns-2/tcl/libs/ns-lib.tcl" and "ns-2/tcl/libs/ns-link.tcl", or NS documentation for more information.

- *$ns* duplex-link-op *node1 node2 ...*: The next couple of lines are used for the NAM display. To see the effects of these lines, users can comment these lines out and try the simulation.

Now that the basic network setup is done, the next thing to do is to setup traffic agents such as TCP and UDP, traffic sources such as FTP and CBR, and attach them to nodes and agents respectively.

- set *tcp* [new *Agent/TCP*]: This line shows how to create a TCP agent. But in general, users can create any agent or traffic sources in this way. Agents and traffic sources are in fact basic objects (not compound objects), mostly implemented in C++ and linked to OTcl. Therefore, there are no specific Simulator object member functions that create these object instances. To create agents or traffic sources, a user should know the class names these objects (Agent/TCP, Agnet/TCPSink, Application/FTP and so on). This information can be found in the NS documentation or partly in this documentation. But one shortcut is to look at the "ns-2/tcl/libs/ns-default.tcl" file. This file contains the default configurable parameter value settings for available network objects. Therefore, it works as a good indicator of what kind of network objects is available in NS and what are the configurable parameters.
- *$ns* attach-agent *node agent*: The attach-agent member function attaches an agent object created to a node object. Actually, what this function does is call the attach member function of specified node, which attaches the given agent to itself. Therefore, a user can do the same thing by, for example, $n0 attach $tcp. Similarly, each agent object has a member function attach-agent that attaches a traffic source object to itself.
- *$ns* connect *agent1 agent2*: After two agents that will communicate with each other are created, the next thing is to establish a logical network connection between them. This line establishes a network connection by setting the destination address to each others' network and port address pair.

Assuming that all the network configuration is done, the next thing to do is write a simulation scenario (i.e. simulation scheduling). The Simulator object has many scheduling member functions. However, the one that is mostly used is the following:

- *$ns* at *time "string"*: This member function of a Simulator object makes the scheduler (scheduler_ is the variable that points the scheduler object created by [new Scheduler] command at the beginning of the script) to schedule the execution of the specified string at given simulation time. For example, *$ns* at *0.1 "$cbr start"* will make the scheduler call a start member function of the CBR traffic source object, which starts the CBR to transmit data. In NS, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, and the agent, just knowing how much of the data to transfer, creates packets and sends them.

After all network configurations, scheduling and post-simulation procedure specifications are done, the only thing left is to run the simulation. This is done by *$ns* run.

5.7.2 Terms

Node: A node is a compound object composed of a node entry object and classifiers. There are two types of nodes in NS. A unicast node has an address classifier that does unicast routing and a port classifier. A multicast node, in addition, has a classifier that classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

Link: A link is another major compound object in NS. When a user creates a link using a duplex-link member function of a Simulator object, two simplex links in both directions are created.

Agents: Agents are the objects that actively drive the simulation. Agents can be thought of as the processes and/or transport entities that run on nodes that may be end hosts or routers. Traffic sources and sinks, dynamic routing modules and the various protocol modules are all examples of agents. Agents are created by instantiating objects in the subclass of class Agent, i.e., Agent/type where type specifies the nature of the agent. For example, a TCP agent is created using the command: set tcp [new Agent/TCP]

CONCLUSION & FUTURE SCOPE

Sensor Networks hold a lot of promise in applications where gathering sensing information in remote locations is required. It is an evolving field, which offers scope for a lot of research. Their energy-constrained nature necessitates us to look at more energy efficient design and operation. Our main focus till now has been on data fusion and sensor network implementation on Network Simulator-2. There is often a need to combine diverse data sets into a unified (fused) data set which includes all of the data points and time steps from the input data sets. The fused data set is different from a simple combined superset in that the points in the fused data set contain attributes and metadata which might not have been included for these points in the original data set. Developing a fusion application is challenging in general, for the fusion operation typically requires time-correlation and synchronization of data streams coming from several distributed sources. Since such applications are inherently distributed, they are typically implemented via distributed threads that perform fusion hierarchically. With the help of data fusion process we can enhance the robustness and accuracy of information which is obtained by entire network, certain redundancy exists in the data collected from sensor nodes thus data fusion processing is needed to reduce the redundant information.

REFERENCES

Eduardo F Nakamura, Antonio A.F. Lourerio and Alejandro C. Frery, "Information Fusion for Wireless Sensor Networks: Methods, Models, and Classifications" , ACM Computing Surveys, Vol. 39, No. 3, Article 9, Publication date: August 2007.

AI.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", Computer Networks 38 (2002) 393–422

Kiran Maraiya, Kamal Kant, Nitin Gupta, "Study of Data fusion in Wireless Sensor Network", Proc. of the International Conference on Advanced Computing and Communication Technologies (ACCT 2011).

Ruixin Niu, Michael Moore, Dale Klamer "Decision Fusion in a Wireless Sensor Network with a Large Number of Sensors", Niu, Ruixin; Moore, Michael; and Klamer, Dale, "Decision Fusion in a Wireless Sensor Network with a Large Number of Sensors" (2004). Electrical Engineering and Computer Science. Paper 82.

José M. Bernardo ,"Bayesian Statistics", Probability and Statistics (R. Viertl, ed) of the Encyclopedia of Life Support Systems (EOLSS). Oxford, UK: UNESCO, 2003.

Jaime Esteban, Andrew Starr, Robert Willetts, Paul Hannah, Peter Bryanston-Cross " A Review of Data Fusion Models and Architectures: Towards Engineering Guidelines"

R. Olfati-Saber , Distributed Kalman Filtering for Sensor Networks

Joel Le Roux, An introduction to Kalman Filtering :Probabilistic and Deterministic Approaches, University of Nice leroux@essi.fr, 20 nov 2003

John Spletzer, "The Discrete Kalman Filter". Lecture notes CSC398/498. Lehigh university. Bethlehem, PA, USA. March 2005.

R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE, 95, Jan. 2007.

Guoliang Xing1; Rui Tan2; Benyuan Liu3; Jianping Wang2; Xiaohua Jia2; Chih-Wei Yi4, "Data Fusion Improves the Coverage of Wireless SensorNetworks", MobiCom'09, September 20–25, 2009, Beijing, China.

R. G. Ingalls, "Introduction to simulation: Introduction to simulation," in WSC'02: Proceedings of the 34th conference on Winter simulation. Winter Simulation Conference, 2002, pp. 7–16.

Yashwant Singh, Ankur Nadda and Sahil Gupta, " Data Fusion in Wireless Sensor Networks: A Perspective", International Journal on Information and Communication technologies (IJICT), serial Publication Delhi, Volume 6 No.1-2, pp.54-77, march 2013. ISSN: 0973-5836.