KNOWLEDGE PROCESSING IN AUTO-CAMUS

by

Knowledge Processing Team[1]

Technical Report

RTMM, Kyung Hee University

June. 2006

Approved by _____

Porject Supervisor _____

Date _____

[1] Bilal Ahmed, Faraz Rasheed., Guan. Donghai,, Sungyoung Lee, Uzair Ahmad & Young Koo Lee. (Note: All in alphabetical order)

UBIQUITOUS COMPUTING

GROUP, KYUNG HEE

UNIVERSITY

ABSTRACT

KNOWLEDGE PROCESSING IN A CONTEXT-AWARE MIDDLEWARE FOR UBIQUITOUS COMPUTING

by Knowledge Processing Team

Project Supervisor: Professor Sungyoung Lee
Department of Computer Engineering

Context-awareness is one of the fundamental requirements for achieving user-oriented ubiquity. Ubiquitous computing environment consists of diverse range of hardware and software entities, and is about the interactivity of such entities. Context-awareness is one of the fundamental requirements for achieving user-oriented ubiquity. A ubiquitous environment requires extensive data and information processing, for this purpose the middleware should be able to cope up with huge amounts of data, and adequate means of reasoning. The need for reasoning is required for forming complex concepts about the domain from very specific observable data. Similarly maintaining intensive interaction with a user's environment requires the identification of user-activities, user-location and other high-level concepts. Identification of such complex concepts is vital to system performance and can only be done through reasoning in an efficient and timely manner. We define the completer working of the components needed for knowledge proessing in a true pervasive environment and all related issues.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION TO KNOWLEDGE PROCESSING

1.1 Ubiquitous Computing Vision

The term "Ubiquitous Computing" was originally introduced by Mark Weiser [1] in the year 1991. In his fundamental article *"The Computer for the 21st Century"* [2], he elaborated about "the computer that disappears". For Weiser the way into the 21st century was obvious: Computer and Network technologies are getting smaller, cheaper, and more powerful. Therefore, more and more everyday artifacts are going to be equipped with a reasonable amount of computing power and, maybe even more important, are networked together into a virtually unique network of communicating "things that think". In the pure sense of the word, computing gets "ubiquitous", anywhere, any time. Computers in every thing that is calmly doing what we intend it to do, in a way that is non-obtrusive and user-friendly, in a sense that we do not have to focus our attention on the trivia of running an electronic system.

Research on Ubiquitous Computing (Ubicomp) is related to very many other disciplines from Robotics and Embedded Systems, Networking and Distributed Systems, to Artificial Intelligence and Psychology. Thus Ubiquitous computing is a very difficult integration of human factors, computer science, engineering, and social sciences.

1.2 Context-aware Computing

One goal of Context-aware Computing is to acquire and utilize information about the context of a device to provide services that are appropriate to the particular people, place, time, events, etc. For example, a cell phone will always vibrate and never beep in a concert, if the system can know the location of the cell phone and the concert schedule. Often, the term "Context-aware Computing" is used in a sense synonymously to Ubiquitous Computing. This is because almost every ubicomp application makes use of some kind of context. Ubicomp is mainly about building systems which are useful to users, which "...weave themselves into the fabric of everyday life until they are indistinguishable from it" [2].

For ubicomp systems, *Context* is essential. How can a system be helpful for a user? Users tend to move around often, doing new things, visiting new places, changing their mind suddenly, and changing their mood, too. Therefore, a helpful system seems to need some notion of *Context*.
*In the Human point of view***,** we have a quite intuitive understanding of *Context*. Here, *Context* is often referred to as "implicit situational understanding." In social interactions *Context* is of great importance. A gesture, a laugh, or the tone of sentences builds up the implicit "picture" of what is meant or what communication partner is thinking. The same words can have a completely different meaning in different contexts.
In Computer Science, Context is quite a familiar concept, be it within the discipline of Artificial Intelligence ("Thinking machines"), in Robotics ("Adaptive Systems"), in User Interface Design (like adaptive UIs or office assistants like the Microsoft Office assistant called

"Clippy"), or basically any other discipline (to some extent). Especially, every discipline dealing with human users tries to take into account human behavior one way or the other, with the generated output loops back as part of the vector of input values.

From the variety of definitions commonly used by Ubicomp researchers we can imagine how difficult it is to find a common ground. Context definitions are far away from mathematical precision and a particular definition often strongly depends on an authors' subjectiveness:

- Schilit and Theimer [3]: "Context is location, identities of nearby people and objects, and changes to those objects."
- A. Dey and Abowd [4]: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*
- Pascoe [5]: "Context is the subset of physical and conceptual states of interest to a particular entity."

So what is this leading to? Are those definitions helpful or misleading? In the sense of a functional definition they are only helpful as a general description of what to do. As an application designer they are only stating what they are doing anyway: trying to figure out what input is needed to produce the desired output. Hence, it is of topmost importance to have some common ground or a common "vocabulary" when talking about what *Context* is. We need some sort of *formal approach* towards handling and describing Context. Furthermore, in a software engineering sense, we need building-blocks for building context-aware applications in a structured way. The Context Toolkit [6] by A. Dey is a step into this direction and a good example for this principle (fig.1.) The Toolkit includes building blocks called "Widgets", wrapper classes for Sensors which serve as a hardware-abstraction layer, "Aggregators", which concentrate multiple input values to a single output value, and "Interpreters", implementing some application logic and generating application dependant "higher-level" output based on the input given. They interpret the incoming data according to a pre-programmed scheme.

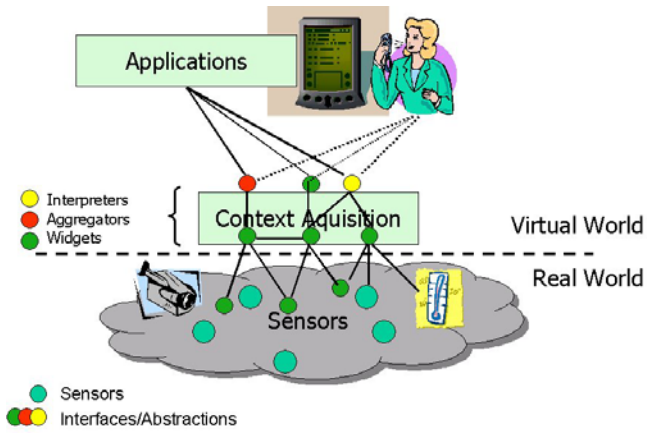


Figure 1: The Context Toolkit Core Components

With the Context Toolkit, the development of Context-aware applications basically consists of several distinguishable steps including
1) The real-world is sensed;

2) Context is detected, aggregated, "interpreted", and
3) Applications are custom-built to match the "context-detection" technology.

However, we believe that there is more tool-support necessary for software engineering and the design of Context-aware applications than provided today. We want to emphasize that the way applications are developed is very dependant on the underlying technology used, which we consider as bad practice in the long run. Research in the direction of decoupling applications from data acquisition seems to be important. This is detailed in the section 2, *Middleware for Context-aware Ubiquitous Computing Environments.*

1.3 Role of Knowledge Processing

Formation of the context information relies heavily on the inter-dependency of the various events, entities and other domain objects. A formal approach for context formation should be able to recognize these inter-dependencies and combine the current information to form the effective context. This sort of an approach would heavily rely on domain-knowledge, the performance of a system made along these lines would also be affected by the knowledge representation scheme used in such a heavy knowledge-based system.

The Knowledge processing layer of Auto-CAMUS is mainly responsible for providing these knowledge-related services especially storing the domain information and definitions in an efficient manner, defining rules for context-processing and synthesis, providing adequate mechanisms for transforming the sensed data into useful complex concepts, providing efficient means of tracking user-location. These tasks can be considered as forming the fundamental aspects of the knowledge processing in a context-aware ubiquitous system. Additional functionalities can be defined to cater for more complex tasks which depend on the degree of interaction to be kept with the user and his environment. As the amount of interaction becomes more intense, so does the need to store more elaborate information, more complex mechanisms for context and complex-concept formation increases. In essence the Knowledge-Processing layer forms the brain of the system.

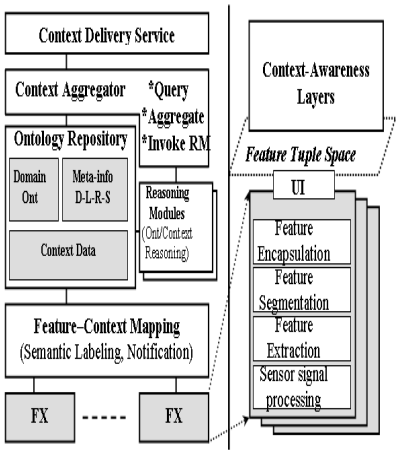


Figure-2: The Auto-CAMUS architecture

The current Auto-CAMUS architecture, as shown in figure-2, incorporates the knowledge processing components at all stages. The extraction of features and their conversion to context information forms the core operation of the

knowledge processing layer. This task is achieved by the collaboration between the ontology repository, the feature-extraction layer, the reasoning modules. The reasoning modules range form general reasoning mechanism to specialized reasoning mechanisms such as the location reasoning module. The knowledge processing layer is responsible for carrying out all the needed tasks and then providing the outcome in the form of general or specific context information to the service delivery layer which in turn is responsible for invoking the required services as mentioned by the application.

The knowledge processing layer requires very intensive computations for carrying out these tasks, it is needed to implement these knowledge-processing in a computationally efficient manner, thus there is a trade-off between the preciseness of the results and the speed with which the computations are performed.

References:

[1] Esprit project 26900: Technology for enabled awareness (tea), 1998.

[2] AKMAN, V., AND SURAV, M. The use of situation theory in context modeling. Computational Intelligence 13, 3 (1997), 427–438.

[3] BACON, J., BATES, J., AND HALLS, D. Location-oriented multimedia. IEEE Personal Communications 4, 5 (1997).

[4] BARWISE, J., AND PERRY, J. Situations and Attitudes. MIT Press, 1983.

[5] BAUER, J. Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic, Mar. 2003. Diplomarbeit.

[6] BOUZY, B., AND CAZENAVE, T. Using the Object Oriented Paradigm to Model Context in Computer Go. In Proceedings of Context'97 (Rio, Brazil, 1997).

Chapter 2

DATA REPOSITORY IN AUTO-CAMUS

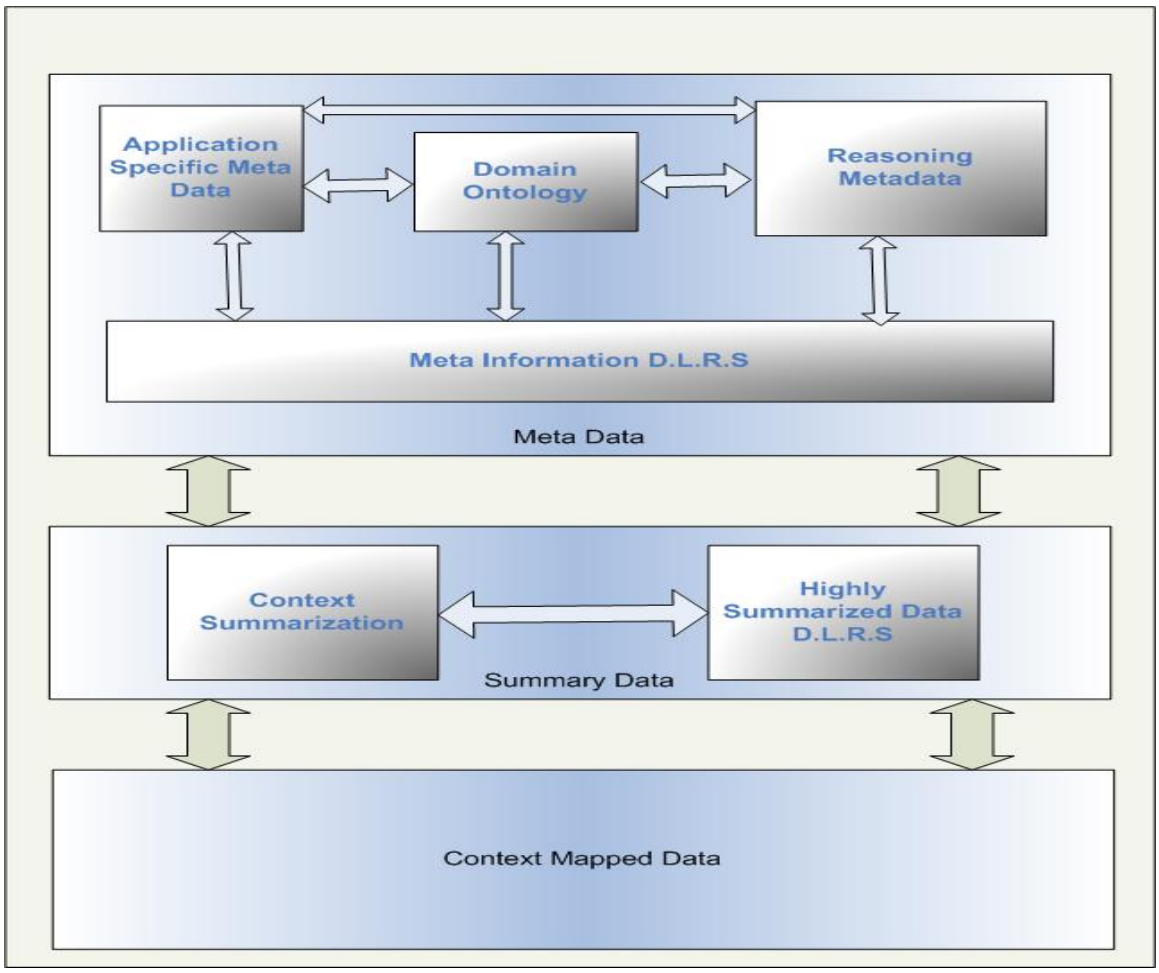
2.1 Motivation

Ubiquitous data management has a lot of challenges, which are not there in conventional data management application. The following list outline a few challenges, this list by no means complete. The ubiquitous application area is growing and a lot of challenges are still on the way.

- Users are moving from place to place (*Location Awareness*, *High Mobility*)
- Data are stored in many places. (*High Distribution*)
- Various devices are with different capabilities and they use different means to store/access data. (*High Heterogeneity*)
- Users cannot consistently control all the smart spaces he ever interacted with. (*High Autonomy*)
- Users generate certain data and may want to access others. (*Sharing and Collaboration*)
- The context of data is also important (*Context Awareness*)
- The source of data must be known (*Provenance*)
- Others: resource-constrained devices, unreliable connectivity and light weight communication and security measures.

Auto-CAMUS has a central repository to maintain data with context. Data is captures by lowest layer through different sensors and given to second layer which build context from that data using OWL Ontology. This context-mapped data is pertinent to query the context later on. The data repository is responsible to maintain this data for future uses. How would this repository attack all the challenges will be discussed later in this report?

2.2 Architecture



2.3 Meta Data Management

The Meta Data is very important in Auto-CAMUS as it gives an open way to accessibility and understanding of the data present in the repository. Metadata can provide richer searching and other services within a service and the glue for integration across several services. There are several key standard for maintaining metadata, we are using RDF based metadata to define and store data related to context and reasoning engines. The metadata is a separate storage in the repository that can be very helpful in migrating this metadata to some other application.

2.4 Data and Knowledge Management

The focus of data and knowledge management in Auto-CAMUS repository is 'doing the thing right' instead of 'doing things right'. It does cater to the critical issues related to adaptation, maintenance and manipulation of both data and knowledge present in the repository. In ubiquitous application the very nature of data required a lot of care and hence the data and knowledge management process and procedures and very important in this regards. One of the main issues in pervasive computing is that how to manage the context data over a large

number of domains. A ubiquitous computing system can consist of many subsystems running on various domains such as home domain, office domain, university domain, etc. Furthermore, many ubiquitous systems can collaborate with each other to build a large pervasive environment. The use of ontology can help sharing the knowledge about data among different domains and systems. However, such a distributed and dynamic environment requires an efficient mechanism to store and retrieve context data over multi-domain repository and at the same time we also required the metadata of this multi-domain to assimilate and understand different domain and context within these domain.

2.5 Knowledge Sharing and Querying

The knowledge sharing and querying process in Auto-CAMUS is very simple and elegant. Each access for querying and sharing requires going through the metadata interface and getting the required information access criteria from there, and then the actual access is being made depending upon the nature of query and information share.

2.6 Provision of Summary Data

Auto-CAMUS offers context summarization and garbage collection for the un-referenced and redundant context data. Similarly a variety of different reasoning engines are available in Auto-CAMUS to apply different type of reasoning required by any application, which is running on top of Auto-CAMUS. Summary data provide a lot of ease in information retrieval. Question arises here what to summarize and how to summarized the required data. The provision to keep summary data in the repository is there and the query processing exploits the fast that this summary data can decrease the query time.

2.7 Conclusion

In this chapter we discussed the role of repository in Auto-CAMUS, we are still evaluating different databases for performance reasons. There is no standard repository available to store and manipulate ontological data. The methodology we suggested here is a framework for knowledge management is ubiquitous environment. The selection of the repository is still on going project and will be incorporated in the next version of this technical report.

Chapter 3

CONTEXT SUMMARIZATION AND GARBAGE COLLECTING CONTEXT

3.1 Introduction

The idea of Ubiquitous Computing [1] is gaining the popularity with every passing day. Several research groups are developing their own ubiquitous computing projects [2] [3] [4] [5]. Ubiquitous (or pervasive) computing provides a computing environment where computing resources are spread through out, present everywhere in the environment and providing services to user seamlessly & invisibly without any explicit user intervention. A ubiquitous computing environment, thus, contains a number of devices, sensors, and software systems.

Context Awareness is among the foremost important features of any ubiquitous computing environment. In order to provide appropriate services to user, an application needs to be aware of the user and environmental context. Similarly at lower levels of abstraction, an application (or middleware) is also required to be aware of the computational context including device and network state. So what is 'context' itself? We take context as the 'implicit situational understanding' and consider all the information that defines a situation as context. So, location, temperature, network bandwidth, device profile, user identity can all be taken as the context information or simply context.

Since a Ubiquitous Computing system needs to deal with such huge and diversified information (context), there should be an appropriate context model to define, represent, and store the context efficiently in some context repository. The management of context information and data in ubiquitous computing imposes lots of issues and challenges. M. J. Franklin [6] has identified a number of such issues in ubiquitous data management such as those posed by adaptivity, ubiquity, mobility and context awareness

We approach the context (or data) management in ubiquitous computing from a different perspective. We are working on to identify the relevance and significance of information that a ubiquitous computing system receives from sensors and its surrounding. We believe that identifying and removing the irrelevant context (we call it 'garbage collecting context') and summarizing the available or incoming context (which we call the 'context summarization') will result in the improved performance of knowledge reasoning, inference making, machine learning and efficient use of computing resource including the storage space required by the Context Repository.

3.2 Problem Definition

Usually a ubiquitous computing environment comprises of a number of different sensors providing context information like

- Environmental context (temperature, pressure, light),
- Audio,
- Video,
- Location context,

- Computational context (network bandwidth, underlying operating system, hardware specification)
- The list goes on and on…

The context information comes in a continuous stream with each sensor emitting the data regularly (at least during some interested activity). We are heading towards flood of context data. Such a huge amount a data requires proper management and should be dealt with great care. At this point, we need to answer what to do with such a huge amount of data? Do we need to store all of this data? More importantly do we really need such a large amount of data?

Several data items sensed from the environment are required for some instant processing and reasoning, e.g., the presence of a person can be used to trigger the activity of turning lights on or caching the data related to the particular user. But most of the time, we also need to store the context for later use; knowledge reasoning, inference making and machine learning. For instance, we may need to keep the context of user presence for some on going (near future) activities, we may also need to store this information to reason about what she is doing and to make the inference what she might be up to. We also need this and other related context information (like time, and other activity details) for machine learning. Using various similar activities, a system can extract the patterns of activities and use such patterns to infer expected user intentions.

But storing all such context information imposes several issues. First, it requires considerable amount of storage space. Since ubiquitous computing systems are essentially distributed, therefore, migrating larger amount of data puts significant burden over network traffic. Secondly, the query processing and data retrieval on large context repository requires significant computing resources decreasing the overall throughput of the system. Thirdly, several contexts needs to be discarded and should not be stored permanently. For example, the data with low precision, because of noise, needs to be filtered out before sensitive operations (e.g., heartbeat rate of a patient). Privacy control also prevents us from storing each and every information, e.g., the information that user is in washroom. Lastly, the performance and efficiency of techniques such as knowledge reasoning, inference making and machine learning depends heavily on the size of supplied data.

So what can be done then? We present our proposed solution to cop up with such issues.

3.3 Proposed Solution

First we need to identify the low precision, irrelevant and redundant context and the one that is no longer useful and remove such context information. We call this process as Garbage Collecting Context (GCC).

Secondly, we need to summarize the actual (raw) context in such a way that it is more meaningful, can be used more efficiently for reasoning, etc and takes up less storage space. We name this process as the Context Summarization (CS).

A simple analogy is the human behavior towards the received news. Every day, we read a lot of news in newspaper, on internet and through television. But do we (need to) remember all the words and information that make up a particular activity or event? What we actually (need to)

remember is some compact information about a particular event that what has actually happened. For example, Bob watches a soccer match for 70 minutes with a lot of attention but after the match is over, he does not remember exactly what had happened in the 14th minute of the game. What he actually remember is some pattern or a summary of the match like who has won the match, few ups and down during the match and how many goals were scored and by whom. This is very close to what we mean by **Context Summarization** that instead of storing each and every raw information, only keep the summarized and meaningful context information. Coming back to the scenario, after the match is over, Bob tends to forget some information, for example, how far did the ball go when Player X kicked it and who received it. Also, as time goes by, he also tends to forget more details like a spectator had broken in to the game field. This act of discarding irrelevant information is analogous to the concept behind **Garbage Collecting Context**

In the following subsections, we will discuss, in more detail, about Garbage Collecting Context and Context Summarization.

3.3.1 Garbage Collecting Context (GCC)

Garbage Collecting Context is analogous to the concept of garbage collection in programming languages [7] [8] where we try to identify the memory areas no longer needed by a program and free it. Similarly in GCC, we try to identify the relevance and significance of context data and filter out (remove) the irrelevant, redundant and useless context.

Garbage Collecting Context can be used at various places. It can be used to filter out the noise in the data, i.e., the data with low precision so that it does not affect the efficiency of system actions taken on the basis of incoming context. For example, in some hospital system if a particular context value of the heartbeat rate of a patient is not sensed accurately because of some noise and interference, we should discard it before entering the system. Several systems [5] provide the precision value or the probability of the correctness of sensed value which can be employed.

Garbage Collecting Context (GCC) can also be used to identify and remove the context that is no longer needed by an application. For example, if in an application, we are getting and storing temperature values after every 5 minutes then an application may not require such context information for a large period of time. It might be useful to discard this history after certain period of time say 3 days. But generally, discarding information is not considered as a good idea; therefore, here we can employ the idea of context summarization and replace the raw history with this summarized history.

Privacy control can also be dealt using the Garbage Collecting Context. In this case, certain privacy policies determine which context should not be stored and included in the system processing and should be discarded. For example, the location of user in private places (like washrooms) and activities during the lunch break should not be processed and stored permanently in the system. Similarly, the list of private telephone calls received and made by the employees (e.g., to or from family members) should not be preserved by the system.

Garbage Collecting Context (GCC) Manager is a strong candidate to deal with such issues and solve them efficiently.

3.3.2 Context Summarization (CS)

Where Garbage Collecting Context (GCC) identifies and removes the irrelevant and insignificant context, the Context Summarizer (CS) operates on the incoming and existing context data to
extract the useful information from the original data, and
convert existing context information to more useful form

so that the output context consumes less storage space and improve the performance and efficiency of query processing, data retrieval, reasoning mechanism and machine learning. The summarized context thus produced, replaces the existing raw context.

3.4 Context Summarization (CS)

Lets consider an example of context summarization. Consider a temperature sensor emitting the temperature value after every 5 minutes. We can simply store this as it is coming. Table 1 demonstrates this case.

Table 1. Temperature values stored after every 5 minute

Time	Temp.
12:05	23 °C
12:10	21 °C
…	
15:35	15 °C
…	

Using Context Summarization, for example, we can summarize this information and group on the daily basis. Table 2 demonstrates one such implementation

Table 2. Temperature values stored daily

Date	Avg. Temp	Max. Temp	Max. Temp At	Min. Temp	Min. Temp At
12/01	8 °C	14 °C	15:15	2 °C	04:35
12/02	7 °C	15 °C	14:55	0 °C	06:05
12/03	9 °C	13 °C	12:40	-1 °C	03:50
…					

Another possible implementation could be achieved when a day is divided into several periods like morning, afternoon, evening, early night and late night and context information is kept for each such period. Table 3 demonstrates such approach

Using the similar concept, location and computational environment context (like network bandwidth and processor load at a particular time) can also be summarized.

Table 3. Temperature values stored for different periods of day

Date	Period	Avg. Temp	Max. Temp	Min. Temp
12/01	Morning	5 °C	8 °C	3 °C
12/01	Afternoon	10 °C	14 °C	8 °C
12/01	Evening	9 °C	11 °C	7 °C
12/01	Early Night	7 °C	8 °C	5 °C
12/01	Late Night	4 °C	5 °C	2 °C
…				
12/04	Morning	4 °C	8 °C	1 °C
12/04	Afternoon	8 °C	11 °C	5 °C
…				

The above examples demonstrate the summarization on historical data, i.e., the data that has been recorded earlier. The context summarization can also be used as the data is received from the sensor. For example, when receiving the data from some audio and video sensors we can summarize it by extracting the useful information from it. With audio, we can extract information like Intensity, Spectral centroid, Transient detection, Low Energy ratio and Audio type (Music, Talk, Telephone Ring, etc). From video sensor, we can extract information like Pixel percent change, Pixel change variance, Motion pattern, Luminous intensity, etc. As a result, instead of storing the complete audio & video context, we can summarize it and only store the relevant information that what the particular audio and video data represent.

One of the benefits of performing the context summarization is reduced storage space. Such a compaction of data repository will save significant amount of storage space which will result in the faster query execution and data retrieval. It will also make the data migration in distributed environment more efficient as the larger the size of context data needs to be migrated the more will be the burden on network traffic. But this is not the only motivation for context summarization. After all, storing the data in Giga Bytes (GBs) is not much a problem in terms of storage space. The primary motivation behind the idea of Context Summarization (CS) is to store only the relevant context information in such a way that it is more useful for context consumers.

Reasoning about the context and drawing inferences based on the context is the primary reason why we are keeping the context in context repository in first place. Reasoning and inference making are the primary tools for providing context aware services to the user. For example, if a ubiquitous computing system knows that when Bob comes to his office in the morning, he likes to check his emails, then a system can start downloading his emails when Bob enters the room in the morning. What makes a reasoning engine perform more efficiently is the amount of source data and the quality of supplied data. We believe that if the context summarization is done properly according to the nature of target context then it will result in less data; optimized for reasoning and inference making. Likewise, machine learning also

depends heavily upon the historical data and if it is supplied with relevant, compact and optimized context data (by context summarization) then it is also expected to perform better and will provide efficient results

Context Summarization is a goal driven task, i.e., if we know why we are summarizing the data and for what purpose it will be used, then the method opted is likely to produce more efficient results. In short context summarization is the process of transformation of raw context data into summarized more useful, relevant context data. Figure 1 depicts this

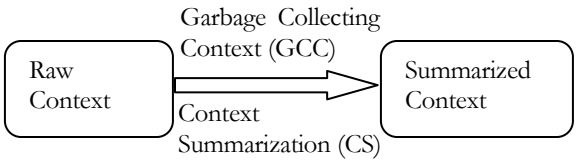


Figure 1. Context Summarization (CS) and Garbage Collecting Context (GCC) Process

3.4.1 Instantaneous & Delayed Summarization

Context Summarization can be classified into '**Instantaneous**' and '**Delayed**' in terms of when the summarization is applied. In **Instantaneous Context Summarization**, the context is summarized instantly as it is received from the context sources; sometimes, even before it being stored in the Context Repository. For example, the summarization of audio and video context (as discussed earlier in this section) comes in this category. Instantaneous Context Summarization is usually irregular and event-based and is performed more frequently.

Delayed Context Summarization is usually performed on the context already stored in the context repository. The summarization of temperature (as discussed earlier in this section), location, humidity and available network bandwidth comes in this category. Delayed Context Summarization, usually, is regular and periodic, i.e., performed in the background after a certain regular interval or at some pre-specified time. Delayed CS is usually performed less frequently and may consume considerable amount of computing resources

Figure 2 demonstrates the flow of context in Instantaneous & Delayed context summarization

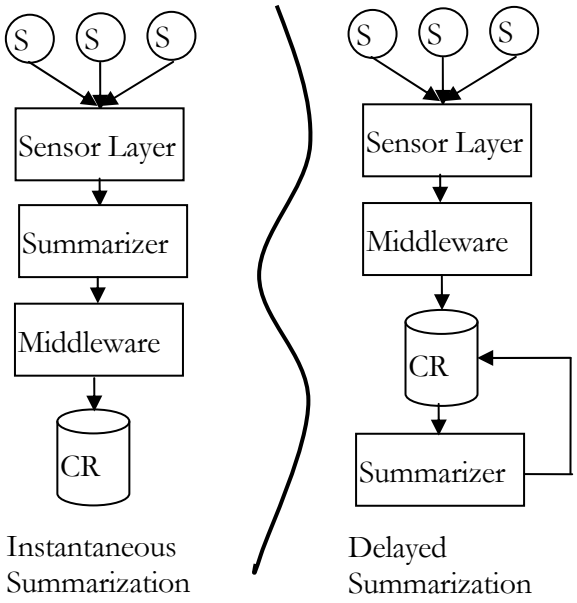


Figure 2. Instantaneous & Delayed Summarization

3.4.2 Three Levels of Summarization

Context Summarization can be performed at three levels. First of all, summarization can be applied at the **sensor layer** by summarizing the context information just as it is received from sensors. The summarization of audio/video context through feature extraction can be carried out efficiently at this level. Secondly, the summarization can be carried out at the **middleware level** when the context has been retrieved from the sensors and system environment and stored in the context repository. The summarization of numerical valued contexts (like temperature, pressure, available network bandwidth, etc) through aggregation and generalization techniques and that of location and activity information through pattern identification can be performed at this middleware level. Finally the summarization can also be carried out at the **application level**. Here the application specific logic specifies how to carry out the summarization and on which context information. The summarization of user and device profile through categorization technique falls under this level of summarization. Figure 3 depicts the three levels of context summarization

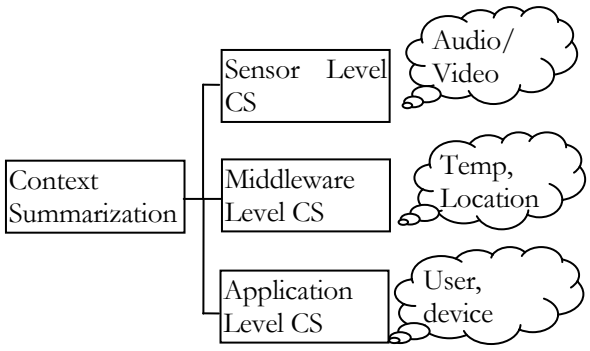


Figure 3. Three Levels of Context Summarization

3.4.3 Autonomic Context Summarization

Context Summarization is purely done for the ubiquitous systems internal use; hence it must be managed by the system with the least possible involvement of the human administrators. Our context summarization model [21] is policy driven where context summarization techniques exists as self managed components with each serving each different category of summarization. The administrator only specifies the policies through meta-data for each context type and the system configures itself accordingly. The system do not remove the context history just as it summarizes it but keep it in the secondary or backup storage and performs the lazy deletion as the history gets matured, thus providing system the ability of self healing which proves handy when something goes wrong. Thus our model justifies the self configurability, self management and self healing properties of IBM Autonomic Computing System model [22]

3.5 Techniques of Context Summarization

Now we will present various techniques that can be employed for Context Summarization in ubiquitous computing systems. We have identified several categories of context information based on the similarities in the context information and nature of context. Each technique is designed for a particular category of context and depends upon the nature of the context.

3.5.1 Aggregation

In aggregation, the history of context information is aggregated to generate compact and consolidated context. Numerical context types like temperature, light intensity, pressure, humidity, available network bandwidth and state of current system resources can be summarized using this technique. In previous section, we have demonstrated how this technique can be used to summarize numerical context information history.

Aggregation is a delayed, regular and periodic type of context summarization, which usually works in the background periodically after certain time interval or at some specific time and is usually less frequent. The Aggregation Context Summarization removes the original (raw) context after the context summarization has been performed.

Note that aggregation is performed when the context information has become the history and is not directly useful for application. Such historical information is useful mostly in identifying user preferences, machine learning and adaptation, reasoning and inference making.

3.5.2 Categorization

This technique of context summarization categorizes different context entities and summarizes the context values of these entities. For example, context information like user profile and device profile can be categorized to form user or device groups having some similar properties. In this way, we can identify the activities and features of a particular group or category like we can track the network bandwidth utilization by some particular user group (say doctors) or by some particular device group (say PDAs) during office hours.

Categorization is delayed and usually static type of context summarization, i.e., it is not performed and changed frequently. Categorization can be performed at system startup by some human or the system can learn itself and define categories as it is executed for elongated

period of time. In any case, the categorization supports Machine Learning and higher level reasoning. Unlike other techniques, Categorization does (should) **not** remove the original context information such as existing user or device profiles.

3.5.3 Context Extraction

In Context Extraction, useful and interested context is extracted from continuous context streams such as audio and video streams. For example, Context Extraction can be applied to video stream received from video sensors like Camera, Webcam to extract features like pixel percent change, pixel change variance, picture motion pattern (such as stable, regular, irregular), luminous intensity, etc. In the similar way, audio context can also be summarized.

Context Extraction is an instantaneous, irregular and event based context summarization. It can start at any time whenever an interested activity starts. Unlike other techniques of context summarization, it can be triggered even before the context is stored in the context repository. In fact, it may discard the original (or raw) context even before it being stored in the repository; hence resulting in only storing the extracted features and not the original data. It results in saving a lot of storage space but may take considerable time in doing so. Some ubiquitous computing projects [16], including our project Auto-CAMUS [5], have been using this technique for some time.

3.5.4 Pattern Identification

Context information can be summarized by identifying existing patterns in the context repository or history of activities. For example, the location context can be summarized using this technique. Consider the location context history stored in the context repository as depicted in Table 4

Table 4. Location Context History of Users and Rooms

Time	User	Room
09:05	1	1
09:02	2	1
09:02	3	1
10:08	1	2
10:37	5	2
10:59	6	3
11:26	3	3
11:44	3	3
…		

Using pattern identification, a system may deduce the pattern of user's location during week days and come out with something as presented in Table 5.

Table 5. Pattern Identification for User Location

Time Period		User	Room	Probability
From	To			
09:00	12:00	1	1	0.76
13:00	17:00	1	1	0.83
09:00	12:00	2	2	0.67
13:00	17:00	2	1	0.89
14:00	19:00	4	3	0.36
…				

In the similar way, system can find the pattern of room occupants during various time periods. Using categorization along with pattern identification, system may also infer which user group (doctors, programmers, operators) occupies which room at different time periods.

Pattern Identification is again delayed, regular and periodic class of context summarization, i.e., it is invoked periodically after certain time interval or at some pre-specified time and works in background. It is resource intensive and thus, performed less frequently. On the positive side, it results in reducing considerable amount of storage space and also supports higher level inference making, machine learning and in predicting future intentions of a user or expected behavior of a device in the current situation. As mentioned earlier, pattern identification works on the existing history of context and replaces the larger history with patterns of activities.

3.5.5 Generalization

In generalization, we map various ranges of context values to a general higher level context. For example, we can map the raw temperature, network bandwidth and user movement speed to general concepts as presented in Table 6, 7 and 8 respectively.

Table 6. Generalization of Temp. Range

Temp. Range (°C)	Generalized Weather
20 ~ 30	Hot
10 ~ 19	Moderate
0 ~ 9	Cold

Table 7. Generalization of Network Bandwidth

Network Bandwidth Range Available	Generalized Network Traffic
1 mbps or more	Mostly free
500 kbps ~ 1 mbps	Moderately used
Less than 500 kbps	Busy

Table 8. Generalization of User Movement Speed

User Movement Speed (km/hr)	Motion Pattern
1 ~ 5	Walking
6 ~ 15	Running
16 or more	Rushing

The examples presented, until now, only contain the one to one mapping of the context values to higher level context (or concept) so where is the summarization in this technique? Actually generalization is an instantaneous kind of summarization performed between the sensor and middleware. This real time summarization only supplies the values to middleware when there is the difference of context general state. For example, suppose the generalization CS module informs the middleware that the current room temperature is moderate then it will only re-inform the middleware (and thus the system above) when the temperature general state is changed from moderate state (to cold, hot or other).

This type of summarization will definitely reduce the processing burden from the middleware in addition to saving the storage space used otherwise for keeping each and every sensed temperature value in the context repository. Generalization is mostly instantaneous kind of summarization working between the sensor layer and middleware. We can combine generalization technique with other techniques like Aggregation and Pattern Identification to further optimize the system performance.

3.5.6 Drift Calculation

This technique calculates the drift or the change of behavior of contextual activities from some fixed known points. For example, consider the network bandwidth of a particular server monitored regularly in a ubiquitous system. The server is supposed to have 1 Mbps network speed. The system hourly monitors the available speed. With drift calculation, we can calculate the drift rate of the network speed during peak hours, normal hours or the average drift from the assigned speed. The system, for instance, may find that during peak hours 5 pm to 10 pm the drift rate is 20% else the drift rate is less than 10%. Now we can keep only this drift rate into the context repository as inferred result from the network speed monitoring.

Similarly, drift calculation is also useful in identifying the user preference. The system can calculate how often the user behavior is different (drifts) from the system assumed behavior. For example, a system may calculate how often it makes appropriate decision for user's favorite TV show or other user intentions and keeps on recording the drift in the context repository.

Mostly drift calculation is delayed summarization and is performed at middleware level. It is especially useful for application adaptation for user preference and machine learning.

3.6 Related Work

Unfortunately, Garbage Collecting Context (GCC) and Context Summarization (CS) have not yet got the attention of researchers. One primary reason is that most of the ubiquitous

computing systems are academic projects and are still in the phase of development. Not many systems have been deployed in real environment and actually used for elongated periods. The issues identified in this work come in front only to one's attention when the actual system is deployed and run for considerable time in real environments. Also, the focus of research community in ubiquitous computing is not towards the context data management, its techniques and issues in this field. Most of the research is still going on in finding ways to make ubiquitous computing operational in first place.

Several existing ubiquitous computing systems support features like noise filtering, privacy control, feature extraction [9] [10] [5] but we believe that using separate components for GCC and CS with clearly defining the responsibility of each component will produce better results; mainly because of the separation of concerns. Also the perception provided by GCC and CS attacks the issues in different and clearer way.

In Database Management Systems (DBMS), there are techniques that deal with similar problems. Data mining [11] and data ware housing [12] use the concept of histogram [13] and multidimensional views of database and work on the aggregate, consolidated data instead of raw data to support the higher level decision making and to identify the hidden patterns in the data. This can be considered as related to the idea of context summarization. Hence, when we extract underlying meaning from the context data, it can be considered as something like 'Context Mining' where we extract higher level context from the lower level context. Online Analytical Processing (OLAP) and data mining is not done on the actual data but on the historical, consolidated and aggregate data while we are performing the context summarization on the actual context. The goal of data mining and OLAP is somewhat similar but we want to transform the raw context to summarized form taking less storage space and provide improved and efficient reasoning and machine learning. Anyhow, the concepts explored in the field of data mining and OLAP are highly useful for the Context Summarization.

Researchers in DBMS have also analyzed the time series data streams for very large databases [14] [15]. Here, they analyze the data coming in continuous streams with time. They have proposed solutions on how to manage, represent and store the time series data streams. This is also highly related to the context summarization.
In traditional DBMS, the data is seldom deleted. But in our context summarizer, we do remove the raw context once it has been summarized and higher context have been extracted. We believe that we can afford to remove certain context data in ubiquitous computing environment and replace it with summarized information. Why? The answer lies in why, in first place, we are storing the context? We are storing the context and maintaining context history so as to reason on context, draw inferences from the context and make the machine learn. As we mentioned above that if the context is summarized properly, keeping the target usage in mind, the application can reason, infer and learn about the activities more efficiently as what they need is the history and consolidated data which we are providing as a result of context summarization.

3.7 Proposed Model for GCC and CS

In this section, we will present our proposed model for designing and developing Garbage Collecting Context (GCC) and Context Summarizer (CS) and provide few implementation guidelines.

The first question, while designing and developing the GCC and CS, is should these components be part of middleware or not? We believe that making these components part of a middleware will yield us the re-usability of design and code and the specific applications will not be required to re-write all the logic and code again.

We prefer designing these components (GCC and CS) as frameworks [17] [18] so that applications only need to provide the **hotspots** (areas of specification) for their specific needs. Hence, **Garbage Collecting Context** (**GCC**) can be developed in such a way that application specific techniques for Noise Filtering and Privacy Policies can be induced even while the application is operational. XML provides a good solution to specify which kind of data can be considered as garbage. For example, an application can specify, through XML, that from 1 pm to 2 pm, there is a lunch time at room X, so the location and other activities of users over there should not be monitored. The GCC contains some pre-specified noise filtering techniques and privacy policies while newer or updated policies can also be inducted in it exploiting the framework based design of GCC module. Figure 4 shows the proposed architecture of Garbage Collecting Context (GCC) module.

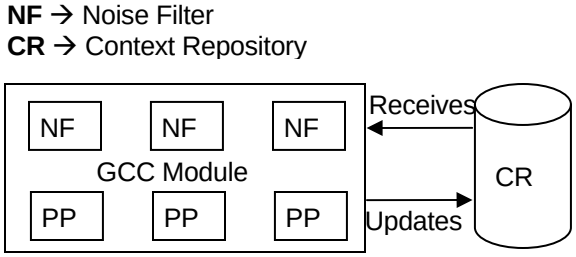


Figure 4. Garbage Collecting Context Module

The GCC retrieves context data from Context Repository (CR), identifies noise (corrupted) context using its Noise Filters (NF), applies Privacy Policies (PP) to remove privacy sensitive context and updates the context repository. Depending on the implementation, GCC may not actually delete the context as it identifies the context as garbage but only mark that particular context information and later remove the context or move such context to some other repository for some human or system analysis.

Context Summarizer (**CS**) can also be developed with the framework technique. There are various context summarizer sub-modules for each different category of context. We call these sub-modules as Context Category Summarizer (CCS). Each CCS is responsible for dealing with each different category of context. Thus temperature, humidity, network bandwidth, luminous intensity can all be summarized using a single Context Category Summarizer (CCS). Context Summarizer (CS) is supplied context information along with Context Meta-Data (CMD). This context meta-data, usually represented through XML, specifies the type (or category) of supplied data, so that the CS may decide which Context Category Summarize (CCS) should be used to summarize this context information. All Context Category Summarizers (CCS) implement a particular interface so that the CS can access each of the CCS uniformly. Because of the framework based design of the CS, new CCS can be added and the existing CCS can be updated while the application is operational. Figure 5 shows the architecture of Context Summarizer (CS).

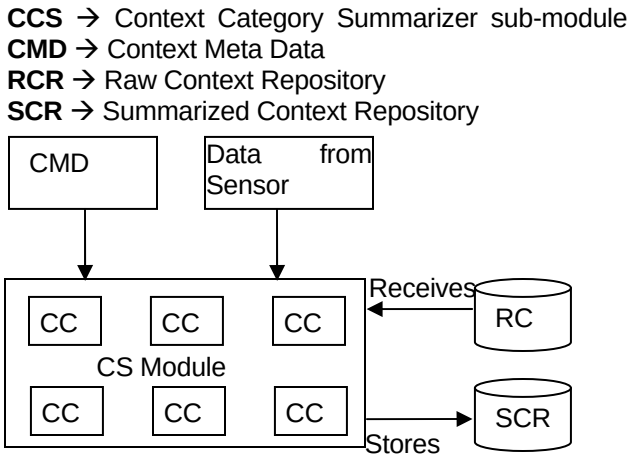


Figure 5. Context Summarization Module

Context is received from various context sources like sensors, computational and departmental infrastructure. The received context is then stored in some Raw Context Repository (RCR). Context Summarizer (CS) usually receives the source context data from this repository (RCR), summarizes and stores it in a separate repository called Summarized Context Repository (SCR). In practice, the RCR and SCR are not physically and logically different databases. Instead, they are managed in separate database tables, in case of related database management systems. Figure 6 presents the flow of Context in the presence of Context Summarizer.

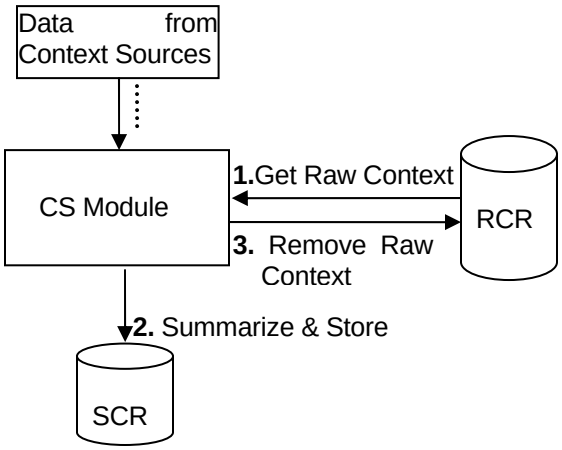


Figure 6. Context Flow in case of Context Summarizer

We are using our middleware Auto-CAMUS [5] to apply the context summarization. The interaction of summarization module with other components of middleware is presented in Figure 7.

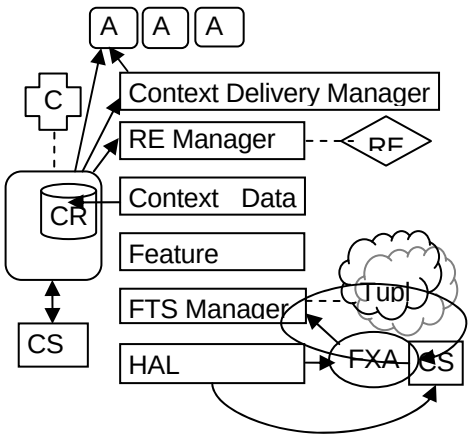


Figure 7. Interaction with other middleware modules

First we extract features (unified representation of sensory data) through our Feature Extraction Agents (FXA) and store all these features in Feature Tuple Space (FTS) which is an in memory repository of current context or the latest information received from sensors. As a new instance of information is inserted in FTS, the older one is transferred to the Context Repository (CR) represented using ontology in OWL through Feature-Context Mapping Layer. From then, all the middleware modules (reasoning engines, middleware services) and application access this information from the context repository. As data is stored in the repository, we summarize this information timely and store back to repository. One approach (used in case of temperature, humidity, etc) the raw information is removed from the repository and only the summaries or aggregates are used to answer queries. Another approach is to keep multiple summaries of different strength are kept and used to reply the query with appropriate confident values. A hybrid approach can also be used in which both summaries and raw information are kept; specific or precise queries are answered from raw data while the general queries are answered through summarized information

3.7.1 Query Translation

Context Summarization modules change the context repository and form data units with different schema than the original one. How can context consumers cater with this? How do they know whether particular information is in summarized state or it is still in raw form? As in Figure 2, there is a special module called Query Translation (QT) which encapsulates context repository (CR). All other modules (CS, Reasoning Engine, Applications, etc) interact with repository through QT. Query Translator makes all the access to CR transparent, i.e., even the modules and applications are not required to be aware of summarization process. It keeps track of partition of summarized and raw data and directs the access to these accordingly by intercepting each and every access to CR. If the required data has been used in the summarization, it directs the queries to the summarized data repository. The results produced due to QT are not 100% accurate; hence it also returns a confidence value with each query result. Further, a query may also specify the minimum degree of confidence for the required results.

3.7.2 Context Category Summarizer (CCS)

Each category of context is summarized by a particular Context Category Summarizer (CCS); hence there is a different CCS for aggregation, pattern identification, etc based information. For example, temperature, available network bandwidth and noise level can be summarized using aggregation based CCS. Each CCS instance contains
(a) summarization algorithm,
(b) general parameters (key field, required fields, etc),
(c) specific parameters (source & target data source, summarization strength, time interval for repeated invocation of summarization),
(d) query translator for summarized information
Context Summarization Manager also maintains a list of context information used by different CCS for summarization, an example for such a table is present in Table 9.

Table 9. List of context information summarized by different CCS

CCS_ ID	CCS_ Instance_ ID	Context_Type _ID	Last_Updated
017	1	1 (temperature)	09/19/05 05:42
017	2	4(light intensity)	09/19/05 11:37
019	1	21 (location_A)	09/19/05 17:16

Using this list, a Query Translation Manager can identify whether a particular context information type is summarized and also if the required data has been in summarized or it is still in raw format. Moreover, if the required information is in summarized state, then which CCS's QT should be invoked to get the query result? The general process flow of query processing is presented in Figure 8

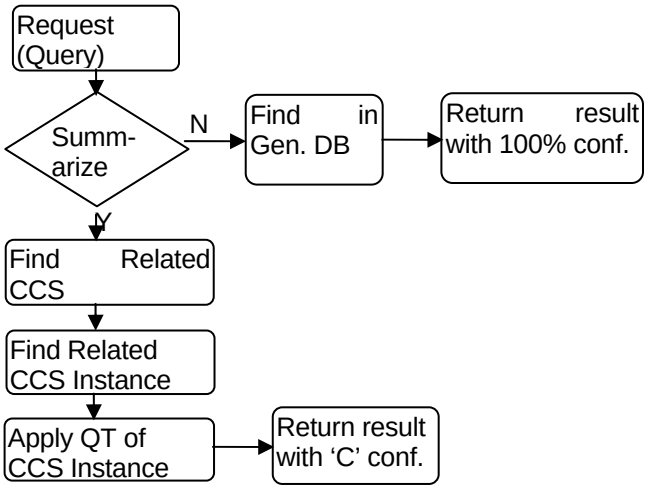


Figure 8. General Process Flow of Query Processing

3.8 Issues & Challenges

Context Summarization (CS) has its own unique research issues and challenges both at conceptual and implementation level. The issues range from questions like what/when/how context can be summarized to the performance and security of the ubiquitous computing system. In the following subsections, we will identify several such issues and wherever possible identify few applicable solutions.

3.8.1 Performance Overhead

Perhaps the foremost concern to apply Context Summarization (CS) techniques is the performance cost. What will be the performance requirement and what will it provide in return? Do the benefits achieved by these methods justify the computing resource consumption? We believe that a proper application of CS (like those discussed in section 4) will yield the performance improvement and will not eat up many resources. In any case, the overall system should not be ceased or hung-up during the execution of CS modules, the resources (like context repository) should not be locked for noticeable period of time and the regular execution of the system should continue without any disturbance or interruption by these modules. But the problem is how to achieve this? We need Context Summarization only when there is considerable amount of context information; a considerable amount of context means a considerable amount of processing and resource consumption to produce useful output. Designing algorithms and techniques to minimize this resource consumption is probably the biggest issue in Context Summarization.

3.8.2 Security & Risks

Security is the most questionable part of today's computing systems. What security threats would the techniques of context summarization will pose? The CS modules operate totally inside the middleware and ubiquitous systems and directly access and modify the context information which is the most valued asset of any ubiquitous system. Hence, the components and modules must be administered and validated carefully.

About the risks involved, firstly Context Summarization (CS) results in some data and precision loss. Failing to compensate this precision lost may result in decreasing the performance and overall throughput of the system. Secondly, improper Context Summarization may make the reasoning and machine learning even more difficult, complicated, inefficient, incorrect and misleading instead of improving it. Finally, CS makes changes to the existing Context Repository (CR). Several modules of middleware and application might be accessing the CR at the same time. Such a sudden modification might be unexpected for these modules and may make them produce unexpected results and must be avoided.

3.8.3 Incorporation of Summarized Context

Incorporation of summarized context into existing context repository is another research issue. The important point to note here is that the summarization process is performed when the ubiquitous system is completely operational. The context information might be in use when summarization modules access it to summarize and attempt to remove some information

replacing them with the summarized context. But how the application processing can be diverted to the summarized context in between the processing? Our idea is to direct all the access to context repository first to the summarized part of repository. If the system requirement can not be fulfilled by it only then the access should be forwarded to the un-summarized part of the repository. The access to un-summarized repository should block all the CS modules on this part of un-summarized repository until the request has been fulfilled. To avoid such collision, we also recommend making the summarization on context history as periodic and scheduled according to the system load.

3.8.4 Other Issues

Some other research issues and challenges are;

- Using ontology for context representation is gaining acceptance in ubiquitous computing community [5] [20]. We need to define & implement techniques of CS that can efficiently operate on the ontology based context repository [19]
- How are we going to deal with the distributed and ubiquitous nature of middleware, data repository and applications?
- What are the security, trust and service level guarantees required for systems using CS techniques?
- What could be the possible impacts of these techniques, especially when something goes wrong?

In order to make Context Summarization (CS) feasible, we need to solve these issues and answer the challenges posed by these concepts.

3.8.5 Risks Involved

Garbage Collecting Context (GCC) and Context Summarization (CS) are sensitive in nature as they directly access context information and modify it. Information is always one of the most important assets of any system and organization. Hence, techniques like these must be applied with great care. In this section, we will briefly mention about some risk factors that should be considered while developing and implementing GCC and CS techniques

Garbage Collecting Context (GCC) and Context Summarization (CS) both will result in some data and precision loss. Failing to compensate this precision lost may result in decreasing the performance and overall throughput of the system.

Improper Context Summarization may make the reasoning and machine learning even more difficult, complicated, inefficient, incorrect and misleading instead of improving it

GCC and CS will make changes to the existing Context Repository (CR). Several modules of middleware and application might be accessing the CR at the same time. Such a sudden modification may be unexpected for these modules and may make them produce unexpected results and it must be avoided.

3.9 Future Work & Conclusion

The foremost important issue is the performance cost and the selection of time interval for the invocation of summarization. Synchronization of the different CCS modules is also an

important consideration. If too many CCS modules start performing summarization then the overall system performance might degrade. Also there might be some queries for the data that is currently being summarized; we are also working on implementing the appropriate locking mechanism. Another interesting future work is to implement hierarchical summarization with different summarization strength and which allows inter-module negotiation [23] for required summarization strength and confidence values for queries. For our future work, we also want to use the concept present in [23] for summarization strength negotiation.

In the conclusion, we will say that Garbage Collecting Context (GCC) and Context Summarization (CS) are new, interesting and useful research areas and include a number of interesting research issues. We have presented both the benefits that can be achieved and risk factors that are involved in using these techniques and have also identified four different techniques for implementing Context Summarization (CS). We have also presented our proposed model for implementing these concepts and identified certain research issues and challenges we expect to face. We have concluded that these are sensitive operations and must be handled with great care and applied after rigorous testing. Finally, 'to summarize and how to summarize?' that is the question!

References

[1] M. Weiser, The computer for the 21st century. *ACM SIGMOBILE 1999 Review*

[2] Dey, A.K., et al.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Anchor article of *a special issue on Context-Aware Computing, Human-Computer Interaction (HCI) Journal*, Vol. 16. (2001)

[3] Chen Harry, Tim Finin, and Anupam Joshi: An Intelligent Broker for Context-Aware Systems. In: *Ubicomp 2003*, Seattle, Washington

[4] Gaia: A Middleware Infrastructure to Enable Active Spaces. Manuel Román et al., In *IEEE Pervasive Computing*, Oct-Dec 2002

[5] Hung Q. Ngo, Anjum Shehzad, Saad Liaquat, Maria Riaz, Sungyoung Lee: Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. *EUC 2004*: 672-681

[6] Michael J. Franklin, Challenges in Ubiquitous Data Management. . Informatics: 10 Years Back, 10 Years Ahead, LNCS #2000, R. Wilhiem (ed)., Springer-Verlag 2001

[7] Henry Lieberman , Carl Hewitt, A real-time garbage collector based on the lifetimes of objects, *Communications of the ACM*, v.26 n.6, p.419-429, June 1983

[8] Richard Jones, The Garbage Collection page, http://www.cs.ukc.ac.uk/people/staff/rej/gc.html

[9] Mike Spreitzer, Marvin Theimer, Providing location information in a ubiquitous computing environment, *ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles* Dec 1993, Volume 27 Issue 5

[10]Jason I. Hong, James A. Landay, Support for location: An architecture for privacy-sensitive ubiquitous computing, *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, June 2004

[11]Alex Berson , Stephen J. Smith, Data Warehousing, Data Mining, and OLAP, McGraw-Hill, Inc., New York, NY, 1997

[12]Inmon, W.H., Building the Data Warehouse. John Wiley, 1992

[13]D. Barbara et al., The New Jersey Data Reduction Report, Bulletin of the *IEEE Technical Committee on Data Engineering* December 1997 Vol. 20
[14]Lin Qiao et al, Data streams and time-series: RHist: adaptive summarization over continuous data streams, *Proceedings of the eleventh international conference on Information and knowledge management*, Nov 2002
[15]Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation, *Proceedings of the 18th International Conference on Data Engineering* (ICDE'02), Feb 2002
[16]Moore, D., I. Essa, and M. Hayes, Exploiting Human Actions and Object Context for Recognition Tasks, In Proceedings of IEEE International Conference on Computer Vision 1999 (ICCV'99), Corfu, Greece, March 1999
[17]Mohamed Fayad, Douglas C. Schmidt, Object-Oriented Application Frameworks, *Communications of the ACM*, Volume 40 Issue 10, Oct 1997
[18]Erich Gamma , Richard Helm , Ralph Johnson , John Vlissides, Design patterns: elements of reusable object-oriented software, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1995
[19]Jena – A Semantic Web Framework for Java http://jena.sourceforge.net/
[20]Anand Ranganathan, Roy H. Campbell, An infrastructure for context-awareness based on first order logic, *ACM Personal and Ubiquitous Computing*, Volume 7 Issue 6, Dec 2003
[21]Faraz Rasheed, Y.K. Lee, S.Y. Lee. Context Summarization & Garbage Collecting Context: In Proceedings (II) of the International Conference on Computational Science & its Application 2005 (ICCSA 2005) (Singapore).
[22]The Vision of Autonomic Computing. IEEE Computer, January 2003
[23]Khedr, M. Karmouch, A: Negotiating context information in context aware systems. *IEEE Intelligent Systems* Dec 2004

Chapter 4

BAYESIAN REASONING IN AUTO-CAMUS

4.1 Introduction to Bayesian Reasoning

Bayesian reasoning is based on the celebrated Bayesian rule of conditional probabilities. Application of the Bayes' rule in very complex cause-effect maps requires intense computational and memory resources [1]. Bayesian Networks were developed keeping this limitation in mind. Bayesian networks efficiently represent the cause and effect relationships which exist between the various domain features and at the same time they also provide adequate mechanisms for belief updation based on evidences extracted from the domain [1,2,3]. Traditional reasoning mechanism like rule-based reasoning do not incorporate provision for uncertainty, Bayesian Networks have been designed so that reasoning can be performed under uncertain conditions.

A Bayesian Network consists of variable-set V (discrete/continuous random variables), a directed acyclic graph G and a joint probability distribution P defined over all the variables in the set V [1,2]. A sample Bayesian Network is shown in figure-1.

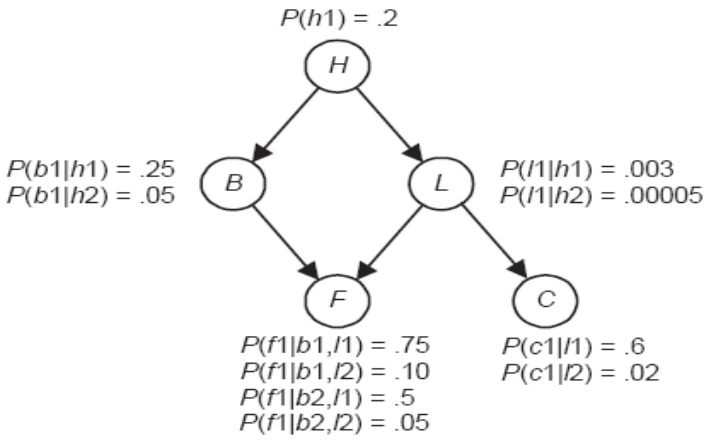


Figure-1: A sample Bayesian Network along with the probability distributions.

As the number of variables in a network grow the size of the joint probability distribution also grows exponentially. Thus it is better to store the conditional probability of each variable conditioned on its parent-set locally, as shown in fig-1. These locally stored probabilities can be combined using the chain rule [1,2] to construct the overall joint probability distribution P.

Bayesian Networks allow a structured representation of all the related concepts in a domain along with adequate representation of the strength of this relationship. The main language used for representing this strength is the probability calculus and the main motivation for using probability calculus comes from the well-formed axiomatic foundations of probability theory and the ease with which related beliefs can be combined [2].

The inference mechanisms for Bayesian Networks are also very well established and a number of inference algorithms exist which can be used to perform inference on sparse graphs.

Algorithms for performing both exact and approximate reasoning exist and can be used depending on the domain being modeled and the requirements. Along with these algorithms for inference generation Bayesian Networks can also be learned from data, and similarly they also provide adequate mechanisms of adaptation and tuning [1,2]. Adaptation can be viewed as online learning, and tuning is parameter adjustment (probability measure) in a supervised manner.

Bayesian Networks have been used in expert systems, decision support systems, fault and anomaly detection for industrial processes, automated planning etc.

4.2 Motivation

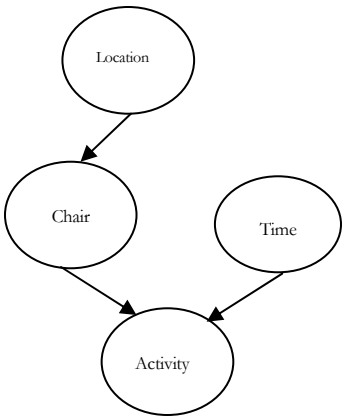
In a ubiquitous environment the system is supposed to infer high-level concepts from sensor and device readings. These sensor and device readings provide only very specific data about certain environment entities such as temperature, pressure, etc. As the system is unable to determine the exact nature of the concept (with complete confidence) the environment can be labeled as being partially-observable. For such partially-observable environments employing a technique which does not cater for uncertainty would be inappropriate.

As an example of the above mentioned description of a ubiquitous environment, consider the case in which the system needs to infer whether the user is having lunch or not. For inferring such an activity it is needed that we have some data about the location of the user, time of the day, and some data about his actions. Let's assume that the system has sensors for estimating the location of the user, similarly there are some pressure sensors on the chair in the dining room. If now a snapshot of the environment is taken into consideration and we find that the chair is indeed occupied, and also that the location of the user is the dining room. These pieces of information are not enough to correctly state whether the user is having lunch or not. It could be that he is reading some thing, and that he is just sitting there talking over the phone. Thus the system is now faced with multiple hypothesis (candidate conclusions) and it is needed that one of them should be adjudged as the most likely one. One way to break this tie among candidate conclusions is to give a score to each of these conclusions based on experience. This would require some data about the habits of the user, where and when he takes his lunch, how often he talks on the phone while in the dining room, does he usually reads in the dining room while not eating anything, thus the system requires some sort of probabilistic knowledge to rate these candidate conclusions. Once these conclusions have been rated the conclusion having the highest score can be considered as the most plausible conclusion which can be reached given the data at hand. Thus all that a system can do is behave in a normative manner, and predict these events in an uncertain manner. A Bayesian network for the example is shown in figure-2.

4.3 Where to use Bayesian Networks in a Ubiquitous Environment

Bayesian Networks are used for reasoning under uncertainty and should only be used for carrying out those tasks which involve a very complex structure of reasoning. They should not be used for a task which can be solved without reasoning for example turning on the lights when motion has been detected. These kind of trivial tasks do not require much knowledge and it would be very costly as far as performance costs are concerned to use a Bayesian network for such a trivial task.

In Ubiquitous environments Bayesian Networks are best suited for accomplishing more complex tasks such as activity recognition, conflict resolution in a multi-user environment, assessing and modeling situations and advising the best action to take.
The design of a Bayesian Network is not an easy job, it requires that variables be identified and adequately modeled and then probability distributions for the root nodes be specified *a priori*. This requirement of prior measures can be fulfilled either through experience (a subjective guess by the designer) or they must be extracted from a large database of cases (objective measure based on recorded cases). Thus the most difficult task in the design of a Bayesian network is the specification of these probability measures. As this task requires a lot of skill and time it is not possible to model all the functionalities of a ubiquitous system through Bayesian networks hence it is needed that they be used only where complex decision making is required.



Every node represents a discrete random variable and each link contains the conditional probability, P(Child | Parent).

Figure 2: A candidate Bayesian Network for the example.

4.4 Bayesian Reasoning in Auto-CAMUS

Context reasoning provided by the knowledge processing layer of Auto-CAMUS, has been designed to accommodate multiple reasoning paradigms, for reasoning about uncertain contexts [4]. The reasoning modules can be developed separately and then plugged into the context-reasoning layer.

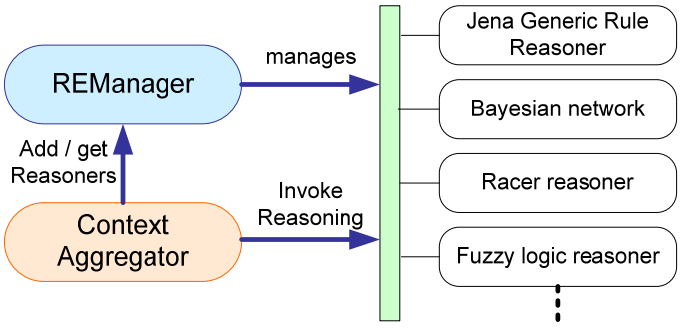


Figure 3: The design of Auto-CAMUS reasoning layer.

As can be seen from figure-3 that Auto-CAMUS provides a Reasoning Manager for managing multiple reasoning modules and a context-aggregator for invoking the reasoning processes and also managing the storage and retrieval of the high-level context so formed.
Another aspect to be considered is the knowledge-representation used in the context-repository of Auto-CAMUS, currently Auto-CAMUS uses OWL for representing ontologies about domain entities. Every reasoning paradigm has its own knowledge-representation so there is a need for translating the native knowledge-representation of the system into a representation which can be understood by the individual module. Similar translation is also needed for storing the high-level context formed through reasoning into the native knowledge-representation scheme of the system.

4.4.1 Design of the core Bayesian reasoning module

The main Bayesian reasoning module consists provides the basic operations for loading a Bayesian Network, absorbing evidences from the domain and providing the inferences on the basis of these evidences.

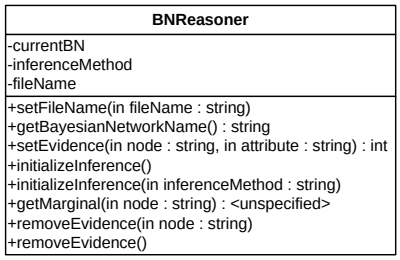


Figure 4: The core Bayesian Reasoning Module

The core module is able to handle a single Bayesian at any time instance. The network is designed and stored in an XML format. Other functionalities provided by the reasoning module is the entry and removal of evidences at any node in the network.
Various Inference mechanisms such as Pearl's Message Passing Algorithm [1], Junction Tree Algorithm for belief propagation [2], and the Bucket Elimination algorithm [2,3]. The output of the inference algorithm is the marginal probability distribution of each variable. This marginal probability distribution represents the posterior odds (the belief in the variables' state) corresponding to the evidence entered and the prior distributions.

4.4.2 The Bayesian Reasoning Manager Module

As the core reasoning module is able to manage only a single Bayesian network, the task of the manager module is to spawn new instances of the core module, so that multiple networks can be loaded into the memory and concurrent reasoning can be managed.
The Bayesian reasoning manager module receives requests from applications, and spawns new instances of the core module in the form of JINI™ services. These services are then provided the necessary input such as the XML files for the network structure and the probability distributions. A sequence diagram which shows the complete working of the Bayesian Reasoning process in Auto-CAMUS is given in figure 5.

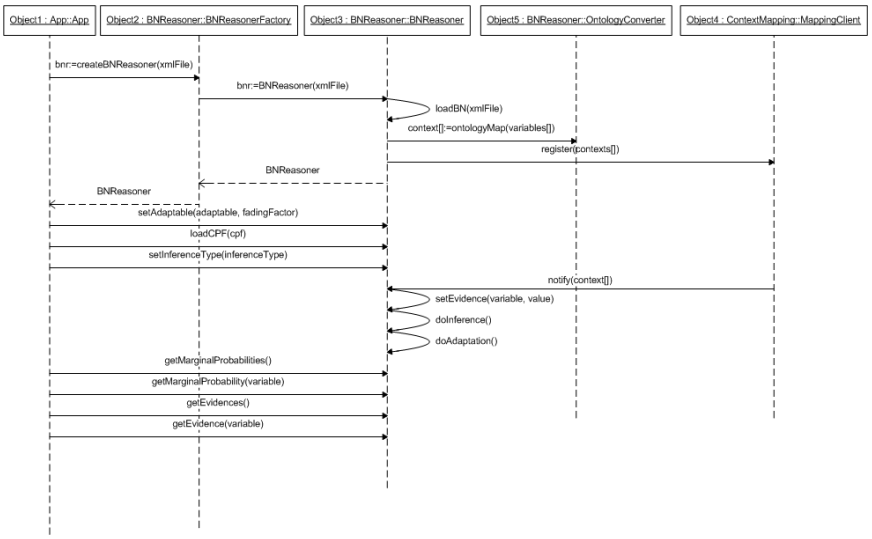


Figure 5: Sequence diagram for the Bayesian Reasoning Process.

The application can interact with the spawned reasoning service directly, and ask for any high-level inference which it wants.

4.4.3 Translation Modules

As mentioned previously there is a need for translating the knowledge maintained in Auto-CAMUS in the form of ontologies to a form that is required for creating a Bayesian Network. As an example consider the following piece of OWL:

```
<owl:Class rdf:ID="InDoor_loc">
        <rdfs:subClassOf>
                <owl:Class rdf:ID="Location"/>
        </rdfs:subClassOf>
</owl:Class>
```

Figure 6: Description of indoor location

The above piece of description logic defines the 'indoors' as a sub-class of location. One way of modeling this piece of knowledge in a Bayesian Network would be to define a discrete random variable named location, and represent the "indoor" as a state of this variable.
Similarly the context information is also represented as an ontology (in OWL) hence it is also needed that the higher-level inference generated by the reasoning module should also be converted back into the OWL format. Hence translation is needed twice in the reasoning process, once when the reasoning is to be initialized and again when the reasoning results are to be provided back to the application.
This translation also defines a mapping for the incoming data from the sensors and devices registered with the middleware to the variables in the network. Thus a mapping client is needed which can map this incoming data directly to the network nodes and instantaneous evidence absorption and propagation can be performed in the Bayesian Network.

References:

[1] Pearl, J. "Probabilistic Reasoning in Intelligent System: Networks of plausible inference", 1988, Morgan Kaufmann Publishers. ISBN 1-55860-479-0

[2] Jensen, F.V. "Introduction to Bayesian Networks", 1998, University College London Press.
[3] Jensen, F.V. "Bayesian Networks and Decision Graphs", 2002, Springer-Verlag. ISBN 0-387-95259-4

Chapter 5

USER PREFERENCE LEARNING

5.1 Introduction

The most famous application of Context-aware computing are smart environments, such as smart home, smart office, smart campus, etc. Absorbing and developing the key idea of ubiquitous computing as well as context-awareness, a smart environment contains a large number of invisible sensors and actuators which enable the system to "think and work" base on its perception of user's context.

The logic to go from input to output of a smart system or from the sensor data to context and then to control commands depends on each system, each user and each kind of service. However, because the most important purpose of a smart system is to satisfy a user, every decision will be based on the user's preference. Therefore learning user preference becomes the most important task.

Here we need a formal concept for user preference. User preference can be understood as what the user wants the system to do in certain situation. For example, when Alice enters the house in a hot sunny day, she wants the air conditioner to be started 10 minutes before that, maintain the temperature of 20oC, and the curtain should be closed, while in rainy days she likes to open the curtain to enjoy the rain. But as for Bob, he wants the curtain to be closed on rainy days, and opened in sunny days.

The task of learning user preference has become very difficult due to many issues.
First, the preference of user does not static. It changes quickly by time, which makes online learning (or adaptation) a crucial requirement.

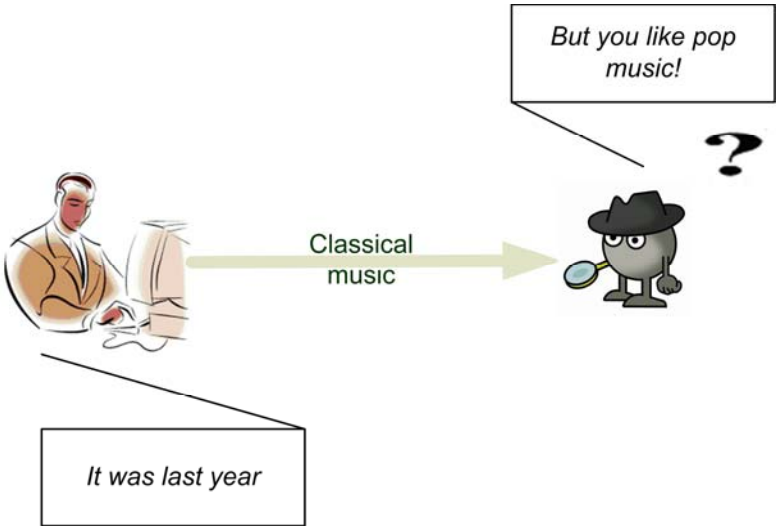


Figure 1: Problem when lack of adaptation.

Second is the uncertainty, which comes not only from the inaccuracy of sensor data but also from the ambiguity of users about their own preference.
Third, when there are many users in the smart environment, the desire of one user can be affected by others. It raises the challenges of distinguishing the preference of each user as well as resolving the conflicts among different user preferences.

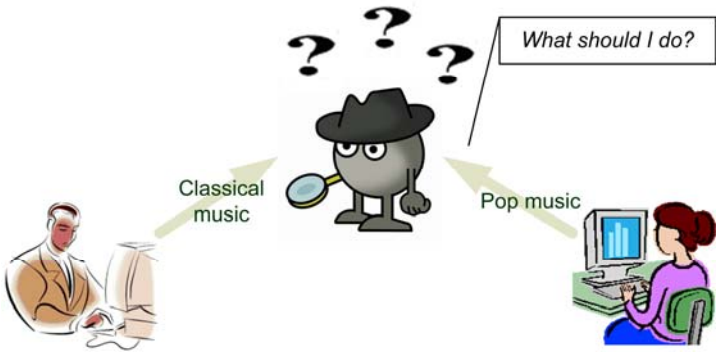


Figure 2: Conflict among user preferences.

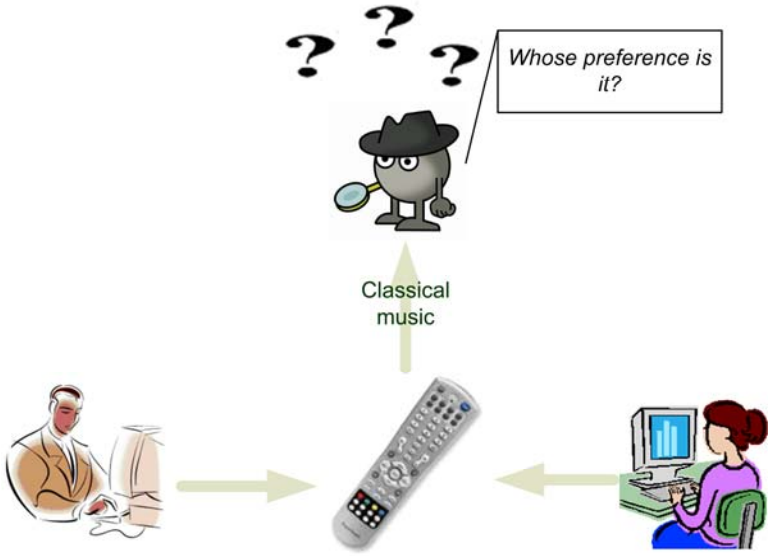


Figure 3: Distinguish among user preferences.

To deal with uncertainty in ubiquitous environment, Bayesian network [7] and Bayes theorem are widely used. Martin Muhlenbrock et.al. [8] uses a Bayesian approach to infer the notions of activity and availability from labeled sensor data in an office environment. The approach is rather simple without considering the multi-user activities. In another ubiquitous system project, to resolve conflicts for Context-aware Media services in smart home environments, GIST [9] proposes Conflict Manager which applies Bayes theorem. In order to resolve conflicts among users, the Conflict Manager sums preferences of users who are collided with each other and recommends the specific contents ordered by the summed preference. It also resolves conflicts among Media services by selecting a Media service with the highest priority. Furthermore, Conflict Manager resolves conflicts among Media services occupied to users by

recommending the Media services. However, this approach lacks of a comprehensive model for user preference.

The Bayesian Metanetwork is first proposed by Vagan Terziyan et.al. [10], to select the appropriate substructure from the basic network level based on contextual features from user's profile (e.g. user's location). Two models of the Metanetwork are considered: C-Metanetwork for managing conditional dependencies and R-Metanetwork for modeling feature selection. An example of this approach is shown in Fig. 4.

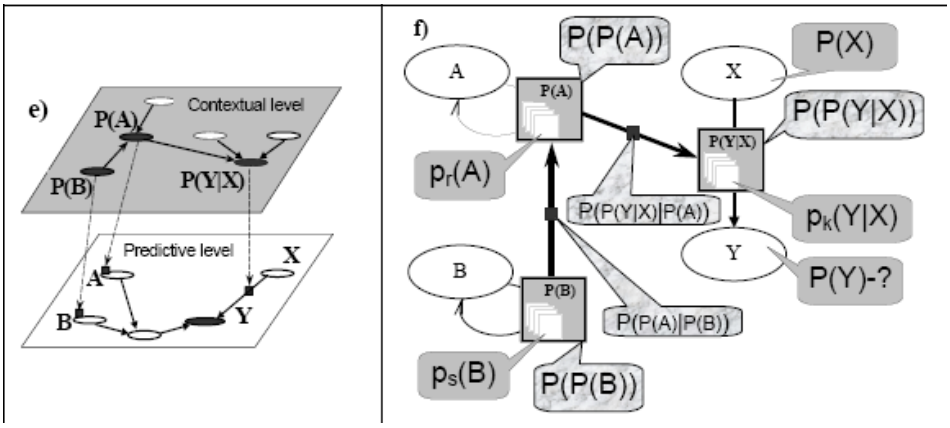


Figure 4: An combination example of R-Metanetwork and C-Metanetwork

However, neither adaptation algorithms for Bayesian Metanetwork nor detailed application for these models are mentioned. One meta-levels of C-Metanetwork is merely described as second order conditional dependencies among probabilistic distributions in previous level (i.e. all the nodes are conditional dependencies), and the R-Metanetwork only model the relevant features selection in a "Nodes-exist-or-not" manner. The big drawback of this approach is that the selection of relevance features and the distribution of conditional dependencies are modeled separately in two different Metanetworks, while in real world the change in a Bayesian network structure often leads to changes in conditional dependencies.
The limitations of current approaches raise the need of a method to learn user preference in ubiquitous environment which can satisfy following requirements:

- Dealing with uncertainty;
- Learning correctly and efficiently the preference of each user in a Multi-user environment;
- Reusable;
- Utilization of the domain knowledge and user-defined rules;
- Online adaptation to the newest user preference.

Besides, we see that when combining the preferences of many users, we often base on user priority. Priority can be fixed, but normally it changes by time, or by situation; for example a sleeping user will be considered more important than others. Hence the context-aware system should be able to learn and adapt to the changes of user's situation-based priority.

To address these issues, we propose a Bayesian RN-Metanetwork (RN stands for Relevant Network). A Bayesian RN-Metanetwork consists of many levels. In each level, there are many sets of Bayesian networks; the distribution of each set depends on the local probability

distributions associated with the nodes of the next level network. we use the Bayesian RN-Metanetwork to learn both user preference and priority at the same time, as well as to resolve the conflict among many user preferences.

My approach differs from the previous ones in at least four aspects.

- First of all, by handling the relevant Bayesian network selection, the Bayesian RN-Metanetwork models can cover both the relevant feature selection and conditional dependency distribution at the same time. Its computations are simpler than those of C-Metanetwork and R-Metanetwork, but it can be used very efficiently for learning in multi-user or multi-agent systems. It also inherits the propagation algorithms of traditional Bayesian network, why in case of C-Metanetwork and R-Metanetwork proposal, the old propagation algorithms can not be applied, and they also have not given any algorithm for it.

- Second, the Bayesian RN-Metanetwork model is very suitable for multi-agent systems. Especially in ubiquitous environments, due to the complex, diverse and open-ended characteristic of the system, the multi-agent paradigm brings much more advantages than the single-agent paradigm.

- Third, the adaptation algorithm for Bayesian RN-Metanetwork is fully described to adapt the model to the continuously changing preference of users.

- Finally, no matter how many users are there in the environment, the priority and preference of each user is calculated separately. This is very useful for widening the scale of systems, as well as knowledge reuse.

5.2 Traditional Bayesian network

Probabilistic graphical models are graphs in which nodes represent random variables, and the (lack of) arcs represent conditional independence assumptions. Hence they provide a compact representation of joint probability distributions. Directed graphical models also called Bayesian Networks or Belief Networks [7].

In addition to the graph structure, it is necessary to specify the parameters of the model. For a directed model, we must specify the Conditional Probability Distribution at each node. If the variables are discrete, this can be represented as a table (CPT), which lists the probability that the child node takes on each of its different values for each combination of values of its parents. Consider the following example, in which all nodes are binary, i.e., have two possible values, which we will denote by T (true) and F (false).

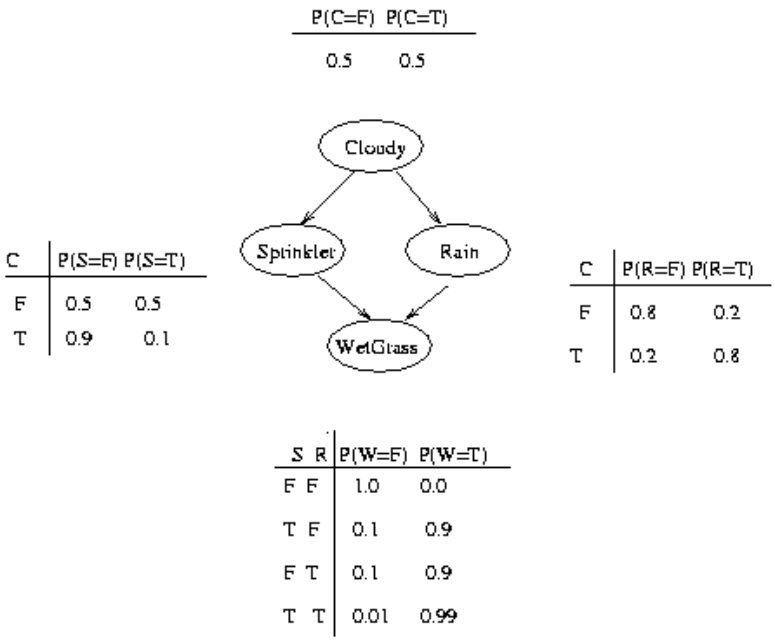


Figure 5: Example of Bayesian network.

Bayesian network is widely used in probabilistic modeling and dealing with uncertainty in machine learning. Some ubiquitous systems are using Bayesian network to learn the user activity, availability, etc. However, this "traditional" approach is restricted to simple use cases such as single user activity or single user preference. The next session explains why we need to replace the traditional Bayesian network by a Metanetwork, by discussing an example of multimedia service preference learning in a Smart Home system. It also introduces the concept of Bayesian RN-Metanetwork.

5.3 The Drawback of Traditional Bayesian network in Preference Learning

Let consider an example of using traditional Bayesian network to learn user preference about multimedia service. we assume that the multimedia service preference of user depends on user location, current activity, time and is different from user to user. If there is only one user in a system, this user's preference can be modeled by a basic Bayesian network as depicted in figure 6.

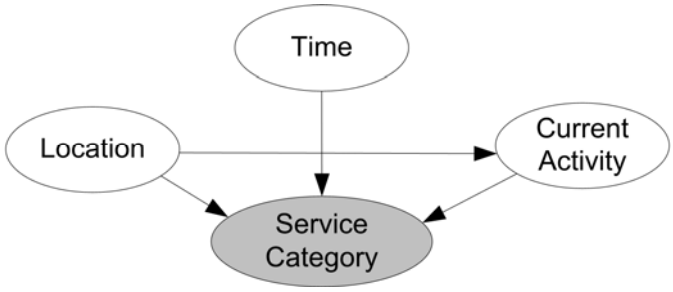


Figure 6: A simple Bayesian network to learn multimedia service preference of one single user. User's current activity is also related to user location.

In this case, it will be very easy for the system to learn the preference of user based on user's commands to select services. Location is acquired from location sensors such as RFID or wireless LAN (PDA), and current activity can be inferred using another Bayesian network [8]. Then the conditional probabilities in this model will be calculated using Bayes theorem.

However, when many users are present at the same time, there will be conflicts among the preferences. When a service is selected, the system can not know whose preference that service is. If there are three users in a smart home, one can think about another model to learn the user preference which is depicted in Fig. 7.

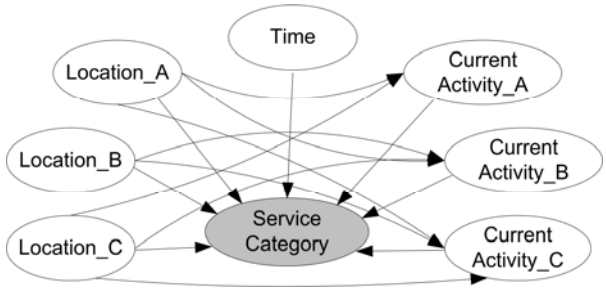


Figure 7: A Bayesian network to learn multimedia service preference of 3 users A, B, C. Each user's activity can be affected by others' location (in case of group activities)

Although a model likes the one in Fig. 7 is easy to create, the size and complexity of the network increases exponentially with the number of users and number of features which affect the preference. Moreover, using this model the system can not learn the preference of each user. In next sub-sections we introduce our approach using Bayesian RN-Metanetwork.

5.4 Bayesian RN-Metanetwork

The Bayesian network model can be divided into many sub-models, each of which models the preference of one single user with no affection from others. Then a Bayesian RN-Metanetwork will be used to manage the distribution of those sub-models and combine them in calculating the final preference.

Definition of Bayesian RN-Metanetwork: The Bayesian RN-Metanetwork is a set of Bayesian networks, which are put on two levels in such a way that the distribution of probabilistic networks on first level depends on the local probability distributions associated with the nodes of the second level network.

The Bayesian RN-Metanetwork is a triplet:

$$RMBN = (BN0, BNS, R)$$

where BNS = {BNS1, BNS2, ... BNSn} is a set of sets of Bayesian networks in first layer and BN0 is the second level Bayesian networks; R = {R1 ... Rn} is a set of interlevel links. The probability distribution of each Bayesian network is included inside it. Each Ri is a link "vertex – network set" meaning that stochastic values of vertex vi in the network BN0 correspond to the distribution of one set of Bayesian networks in the first level.

Bayesian RN-Metanetwork supports multi-agent systems. As depicted in Fig. 8, each set of Bayesian networks in the first level is hold by an agent. Each agent uses the distribution of its Bayesian networks to calculate some needed values, and they communicate with other agents through some interfaces. The interfaces consist of common nodes between agents' networks. The mechanism for belief updating in a multi-agent Bayesian network system is described in [29].

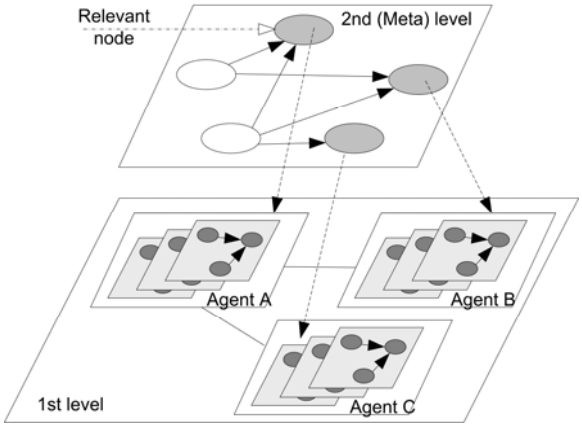


Figure 8: A Bayesian RN-Metanetwork structure.

The RN-Metanetwork can be freely expanded because any Bayesian network included can be itself another Metanetwork.

5.4.1 Modeling User Preference and Priority using Bayesian RN-Metanetwork

When many users are present in a smart environment, and each user has his or her own preference about a certain service, the last decision to select the service is related to the priorities of the users. For example, grandparents have higher priority than their grandchildren, so when the grandfather and his grandson are present in a smart home, the room temperature is adjusted based on the grandfather's temperature preference rather than that of the grandson.

Therefore, when modeling the user preference, we also need to model the user priority.

The user priority can be categorized into 2 types: situation-independent priority and situation-de-pendent priority. Situation-independent priority means that the priority of a user does not change when situation changes, i.e. it does not change by time or by place or under any condition. In contrast, situation-dependent priority changes when situation changes, such as when user changes his place or activity. For example, normally the father has higher priority than his son, but when the son is sleeping, the audio volume and light are adjusted based on the son's preference at that time (audio OFF and light OFF).

The user preference in case of situation-independent priority can be modeled using the 2-level Bayesian RN-Metanetwork. Fig. 9 shows the Bayesian RN-Metanetwork for modeling user preference about Multimedia service in a 2-user system.

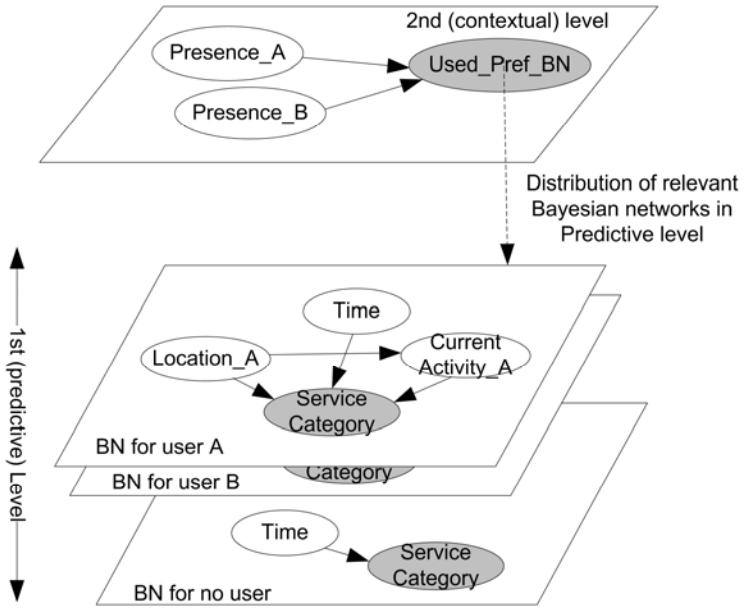


Figure 9: A Bayesian RN-Metanetwork to learn multimedia service preference of 2 users A, B in case of situation-independent priority.

The model is based on following idea: The priority of one user can be understood as how much that user can contribute into the final decision. In the model, the distribution of relevant Bayesian networks which model individual user preferences indicates the proportion of each user preference in the compound preference. Hence the meta-level of this Bayesian RN-Metanetwork also models the user priority. We can see that the user priority in this case depends only on the user presence. Table 1 shows an example of value of the conditional distribution of relevant Bayesian networks (or user priority).

Table 1
Conditional probability of Used_BN node in 2nd level Bayesian network in Fig. 4.

Presence_A	Y		N	
Presence_B	Y	N	Y	N
Use_A_pref	0.5	1	0	0
Use_B_pref	0.5	0	1	0
Use_Nouser_pref	0	0	0	1

From the conditional probability table, we see that the two users have same priority. When both users are present, each of them contributes 50% into the final preference decision.

In case of situation-dependent priority, the 2nd (or meta) level is replaced by a metanetwork. In other words, one more level for learning priority based on situation is added. Fig. 15 depicts the Bayesian RN-Metanetwork for the multimedia services preference in situation-dependent priority case.

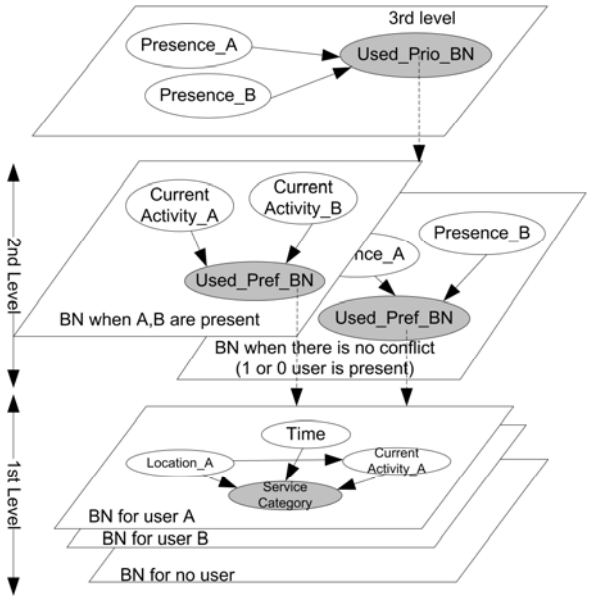


Figure 10: A Bayesian RN-Metanetwork to learn multimedia service preference of 2 users A, B in case of situation-dependent priority. Priority depends on not only user presence but also user activity.

Tables 2, 3 and 4 show the sample conditional probability of Used_Prio_BN node and Used_Pref_BN nodes. Note that the value range of Current activity nodes is reduced for easier demonstration (normally we have a lot of activities such as Reading, Walking, Eating, Working, etc.).

Table 2
Conditional probability of Used_Prio_BN node in 3rd level Bayesian network in Fig. 5.

Presence_A	Y		N	
Presence_B	Y	N	Y	N
Use_AB_prio	1	0	0	0
Use_single_prio	0	1	1	1

Table 3
Conditional probability of Used_Pref_BN node when both A and B are present.

CurrentActivity_A	Working		Sleeping	
CurrentActivity_B	Working	Sleeping	Working	Sleeping
Use_A_pref	0.5	0.3	0.9	0.5
Use_B_pref	0.5	0.7	0.1	0.5
Use_Nouser_pref	0	0	0	0

Table 4.
Conditional probability of Used_Pref_BN node in 2nd level Bayesian network when there is no conflict.

Presence_A	Y	N

Presence_B	Y	N	Y	N
Use_A_pref	-	1	0	0
Use_B_pref	-	0	1	0
Use_Nouser_pref	-	0	0	1

When the priority model is simple and we don't want to make it redundantly complicated, we can use a traditional Bayesian network for the priority model, instead of using 2 layers of Bayesian RN-Metanetwork. The traditional Bayesian network for priority model in Fig. 10 is illustrated in Fig. 11.

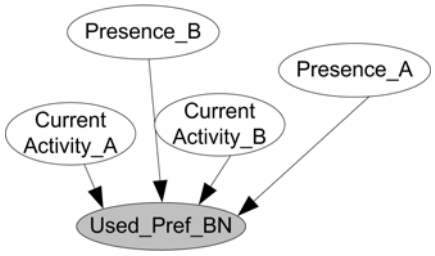


Figure 11: Simple Bayesian network for situation-dependent user priority

The algorithms for calculating the combined preference and learning user preference and priority are explained in next sections.

5.5 Probability Propagation and Adaptation for 2-layer Bayesian RN-Metanetwork

In this session, we will explain the algorithms for Probability Propagation and Adaptation for the basic 2-layer Bayesian RN-Metanetwork. The algorithms are illustrated by computations on the model in Fig. 9.

5.5.1 Notations and Definitions

The Bayesian RN-Metanetwork in Fig. 14 has parameters:

+ 1st level:

- This level has only one set of Bayesian networks: 2 Bayesian networks modeling the preferences of 2 users and one Bayesian network for the case of no user.

- The attributes of the Bayesian networks: Time (denoted T), Location_A/B (denoted Li, i=1, 2 corresponding to A and B), CurrentActivity _A/B (denoted CAi), ServiceCategory (denoted Sx, x=1, 2, 3 for its duplications in 3 networks) and has the values {SV1, ...SVj}.

- The prior probabilities: P(T); P(Li); P(CAi), P(Sx).

- The conditional probability P(Si|T, Li, CAi), i=1, 2 and P(S3|T).

+ 2nd level:

- The attributes: Present_A/B (denoted Pri) represents the Presence of user A/B with the values {yes/no}; Pr denotes the set of all Pri.

- The relevance node: Used_BN (denoted BN) holds the probability to have each Bayesian model in the predictive level with the values {BN1, ... BN3}.

- The prior probabilities: P(Pr) denotes the probability distribution of the set Pr and P(Pri) denotes that of each item in the set.

- The relevance probability: P(BN). The conditional probability P(BN | Pr)

5.5.2 Probability Propagation

Given the evidence P(Pr), P(BNx) is calculated as:

$$P(BN_x) = P(BN = BN_x \mid \Pr).P(\Pr)$$

Then the probability of the target attribute ServiceCategory can be estimated:

$$P(S) = \sum_{x=1}^{3} P(BN_x).P(S_x)$$

with P(Sx) calculated given the evidence T, Li and CAi

$$P(S_x) = \begin{cases} P(S_x \mid T, L_x, CA_x).P(T).P(L_x).P(CA_x) & x = 1,2 \\ P(S_x \mid T).P(T) & x = 3 \end{cases}$$

In other words, the probability of ServiceCategory preference of each user will be calculated separately, and then combined with the weight coefficients which are the distributed probabilities of the Bayesian networks in Predictive level.

The target ServiceCategory SVj with highest P(S=SVj) will be selected.

The Bayesian RN-Metanetwork provides an easy but efficient method for modeling many kinds of user preferences, from multimedia services such as music, television, radio, web page, public information, etc. to the environment parameter such as light, temperature, etc. All we have to do is identify the features which affect the preference of users, build the preference model for each user (a rather simple task if we consider the current development of Bayesian network research) and finally combine them by a RN-Metanetwork. When the system learns, each preference model is updated separately so that they can be reused in other systems.

5.5.3 Adaptation

Even though some approaches address learning issue as the initial of the conditional probabilities from example data sets, the true meaning of learning in a ubiquitous system is online learning. When a ubiquitous system starts working, there is no example data but only the domain knowledge and user-defined rules. For example, the contextual level network can

be initialized by the users' initial priorities. However, the task of online learning, or adaptation, is crucial.

Each time the system makes a decision about which service category to be selected, it then wait a time interval t_w for the response of users. There are 2 cases of user responses:

Approval: If there is no response, then the system assumes that the users are pleased with the decision. In this case, the decision together with the evidences will be considered a single sample and be used to update the Bayesian networks conditional probabilities.

Denial: If one user gives a control command to the system to change the selected service category, it means that the user may not satisfy with the decision of the system. This is a serious case, and the system should not make the same wrong prediction again. So the user's selection together with the evidences will be considered N samples (N>>1) and be used to update the Bayesian networks conditional probabilities.

The adaptation algorithm is based on two assumptions:

Assumption 1: *The contribution of one user preference in the combined preference is equal to his contribution in making a decision.*

The assumption means: When the system estimates the preference of many users, it gives each user's preference a weight. In the other hand, whenever a control command is given by a group of user, each user has his contribution in that command. The weight in first case and the contribution in second case are assumed to be equal.

Assumption 2: *Every user has the tendency of selecting the option which has highest probability calculated by his preference model. This option is called the most favorite option.*

The assumption means: Given the evidences, calculate the posterior distribution of a preference using the preference model of one single user (one of 2 Bayesian networks for 2 users A and B in the previous example, for instance), the option with highest probability can be considered that user's most favorite, and is most likely to be selected by the user himself.

Assumption 2 leads to a definition:

Definition: *a decision matches user preference if it matches the most favorite option of that user.*

Based on the above assumptions and definition, the adaptation algorithm for RN-Metanetwork is introduced:

Adaptation Algorithm

Step 1: update the meta-layer network
- In Approval case, there is no need to update the meta-layer network (users satisfy with the current priority)
- In Denial case:

+ Find the most favorite option of each user

$$SV*_i = \arg\max(P(S_i))$$

where SV*i denotes the most favorite service of user i.
+ Count 1 for the value of relevant node related to the user's preference model if the final decision is the same with user's most favorite option. We use the sequential updating introduced in [30]. Do the following adaptation:

If SV*i = SV* then

$$P(BN_i \mid \Pr_e) = \frac{P(BN_i).s.q + 1}{s}$$

And

$$P(BN_j \mid \Pr_e) = \frac{P(BN_j).s.q}{s} \quad \text{with } j \neq i$$

with SV*: the finally selected services, Pre: the set of evidences of the presences of users, q: the fading factor, q (0,1), s: the effective sample size which is calculated by:

$$s = \frac{1}{1-q}$$

Step 2: update the preference model of each user based on the contribution of that user into the decision
For each user i: Calculate wi, the distribution probability of that user's preference model: $w_i = P(BN_i)$. Count wi for the final selected option and update the user's preference model 1 or N times, in approval or denial case respectively.

$$P_i(S_i = SV^* \mid T_e, L_{ei}, CA_{ei}) = \frac{P_i(S_i = SV^* \mid T_e, L_{ei}, CA_{ei}).s.q_i + w_i}{s}$$

And

$$P_i(S_i = SV_j \mid T_e, L_e, CA_e) = \frac{P_i(S_i = SV_j \mid T_e, L_e, CA_e).s.q_i}{s} \quad \text{with } SV_j \neq SV^*$$

where Te, Lei, CAei denote the evidences of the Time, user location, user current activity of user i, Pi(S | Te, Le, CAe) denotes the distribution of conditional probabilities in preference model of user i qi: the fading factor which is calculated separately for each preference model, to maintain the same experience size.

$$q_i = \frac{s - w_i}{s}$$

5.5.4 Probability Propagation and Adaptation for multi-layer Bayesian RN-Metanetwork

In nature, multi-layer Bayesian RN-Metanetwork is a 2-layer Bayesian RN-Metanetwork with the meta-layer is a Bayesian RN-Metanetwork itself.

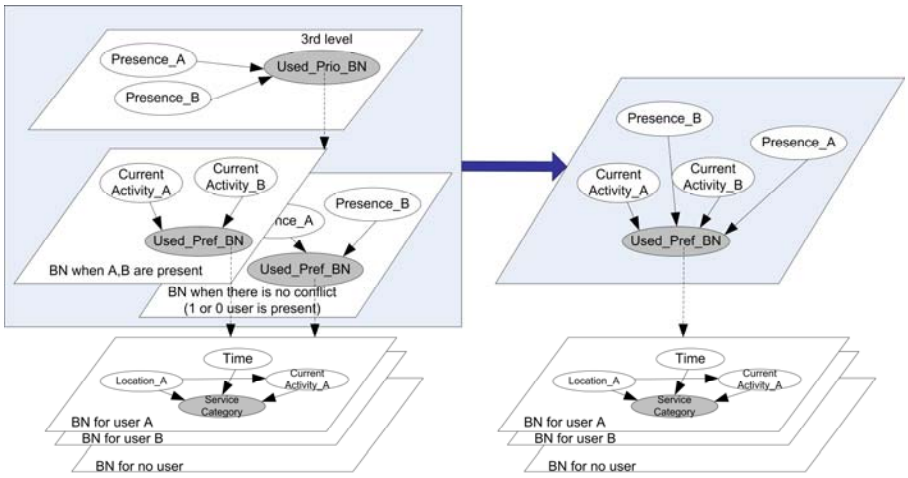


Figure 12: The first 2 levels of a 3-level Bayesian RN-Metanetwork equals with a Bayesian network with same input and output nodes and the intermediate node User_Prio_BN is omitted.

Therefore, the probability propagation algorithm and adaptation algorithm which were described in the previous section can be applied for multi-layer Bayesian RN-Metanetwork in a recursive manner.

The probability propagation process for the Bayesian RN-Metanetwork in Fig. 10:

Step 1: applying the probability propagation algorithm for the first 2 layers to calculate the marginal of User_Pref_BN

Step 2: use the marginal of User_Pref_BN as the distribution for the Bayesian networks in third layer to calculate the marginal for Service_Category.

The adaptation process for the Bayesian RN-Metanetwork in Fig. 10:

Step 1: in case of denial, applying the adaptation algorithm for the first 2 levels of the Bayesian RN-Metanetwork.

Step 2: calculate the marginal of User_Pref_BN and use it to do adaptation for the third level.

In general, the recursive probability propagation and adaptation mechanism for multi-level Bayesian RN-Metanetwork are illustrated in Fig. 13.

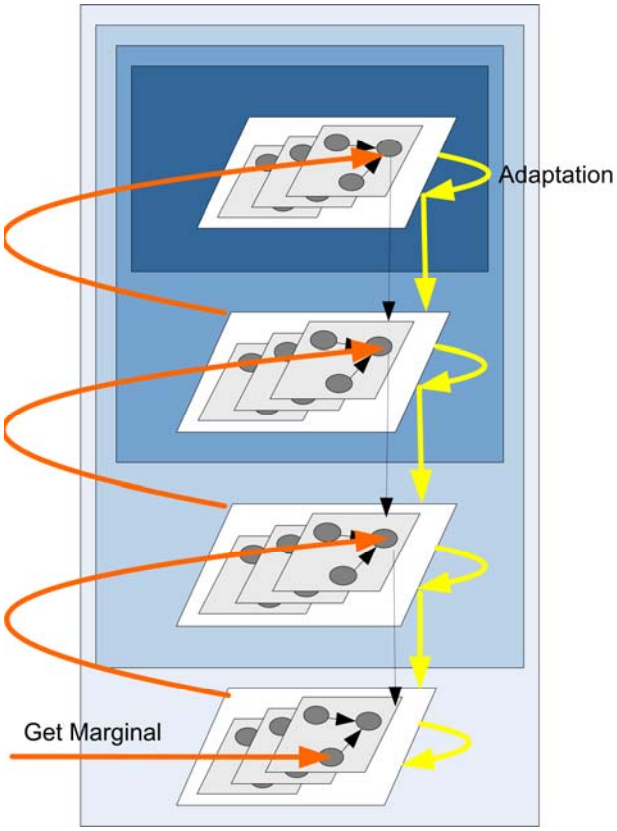


Figure 13: The recursive probability propagation and adaptation mechanism for multi-level Bayesian RN-Metanetwork.

5.6 Evaluation

One can argue that the traditional Bayesian network is still faster than the Bayesian RN-Metanetwork, because for Bayesian RN-Metanetwork we have the overhead of the meta-layer propagation, as well as all the preference models and the priority models should be propagated. In fact, the Bayesian RN-Metanetwork is slower just in case the distribution of every Bayesian network in the first layer is not equal to 0, or by other words, when all users are presented in the system.

Let estimate the calculation time for the Multimedia preference example above when using the traditional Bayesian network and Bayesian RN-Metanetwork.

Analysis in [31] shows the runtime for the brute force method of enumeration is $O(q^m)$, where q is the size of the alphabet (in our example: q = number of values for Service Category i for the preference model, or number of users + 1 in the meta network) and m is the number of unknown variables.

We have seen that Pearl's algorithm, for the special case of a polytree, has an efficient runtime of $O(Nq^e)$, where e is the maximum number of parents on a vertex [32]. It can be seen that in that in the case of the turbo-decoding algorithm [33], the runtime is linear in the size of the network, as evidence is propagated a constant number of times.

Here because both the number of unknown variables and the number of maximum parents will increase when the node number increase, we assume that in general the runtime is $O(q^x)$ with x is proportional to the size of the network.
From Fig. 12, the size S of the network in traditional case is proportional to the number of user. So with N = number of user and q = number of values for Service Category i for the preference model, the propagation time is $O(q^{\alpha N})$ with α is proportional to the number of node in a single user preference model.

From Fig. 13, the size S' of each preference network in Bayesian RN-Metanetwork remains the same for every user. S' = S / N. We have the propagation time of each preference network is $O(q^{\alpha})$ and the propagation time of meta network in Fig. 13 is O(1) given that we know the location of all the users.

Then the numbers of preference model should be propagated k has the binomial distribution, because this is the distribution of obtaining exactly k (Presence = Yes) out of N trials.

$$P(k \mid N) = \frac{N!}{(N-k)!k!}.p^k(1-p)^{N-k}$$

Where p is the mean probability of one user is presented in the location.
The expected value of k:

$$E[k] = N.p$$

Then the expected propagation time is: $O(Npq^{\alpha})$

We have $O(Npq^{\alpha}) < O(q^{\alpha N})$. The Bayesian RN-Metanetwork is still more efficient.

In case of situation-dependent priority, the propagation runtime is added with the time for calculating the priority given the evidences about situation.

The runtime for each priority model is $O(q^{\alpha N^*})$ with N* is the mean of the binominal distribution of taking k users from N users. We have N* = N.p. Hence the expected propagation runtime for a priority model is $O(q^{\alpha Np})$.

Then the total propagation runtime is:

$$O(Npq^{\alpha} + q^{\alpha Np})$$

In this case, it is hard to tell whether the Bayesian RN-Metanetwork is faster or slower. However, besides the speed, the advantages of Bayesian RN-Metanetwork come from at least two aspects:

- First, with the division of a large network into small and single-user models, the design of any Bayesian RN-Metanetwork becomes much easier, especially when we have to assign the conditional probabilities for the network.

- Second, we have the separate models for priority and preference. We also have separate models for each user. This dramatically increases the reusability of the models.

5.7 Implementation

To illustrate the use of this proposed approach, we will describe in details some models of user preference, which are implemented in Hugin Tool [39]. These are the most common user preferences in smart home systems, such as the preference for light, temperature, curtain, multimedia service category. The models go along with some numerical examples to show how they work.

This chapter will also contain the implementation details for the Bayesian RN-Metanetwork, as well as the user preference learning mechanism in Auto-CAMUS.

5.7.1 Implementation of User Preference models for Smart-Home Systems

Every system has its own input-output and hence has specific kinds of user preference. One of the famous scenarios is a smart home system where various sensors, controls (actuators) are installed, and many users are involved. In our implementation, we assume that there are 2 users in the house. The preference and priority models which are presented in this chapter are implemented in Hugin, a tool for building Bayesian networks [39].

Categories of User Preference in a Smart Home System

As previously said, user preference is what the user want the system to do in certain situation. In more details, it is the relation between the context of user, which consists of sensor data and other input information, and the control commands which the system give to the actuators inside it. Therefore, to know what kind of user preference we have in a system, we should know all the input information / sensor data as well as command controls a system should have.

The preference learning module in a Smart home system takes as it input the time, the location of user, home environment sensor data, including indoor/outdoor light intensity, audio intensity, indoor/outdoor temperature, current weather, etc. It also receives the current activity of the user from the activity reasoner, which can be implemented using a Bayesian network [8]. Light preference may consider the affection range of each light, while Temperature preference takes into account the duration of user staying in the room (for example, if the user has just enter the room and the outside is hot, he will need cooler temperature than the user who has stayed in the room for a long time.

The outputs of the preference learning module in a Smart home system are the preferred light intensity level (commands for the light control), temperature level (commands for air conditioner and heater control), multimedia service categories (such as television channel, music genre and audio broadcasting channel), curtain control (close / open), etc. Because the multimedia service has already discussed above, in this section we will only describe the models for curtain control, light and temperature preference.

There is an assumption that there is no body sensor to tell about the user health status or emotions. Otherwise, the preference will depend a lot on user emotion and health.

Curtain Control Preference

In some systems, a curtain can have only 2 statuses CLOSE and OPEN. In other system, the curtain control can open the curtain to some levels. Assume that we have 3 levels for curtain control: CLOSE, OPEN and HALF_OPEN.

Normally, user wants to open or close the curtain depending on the outside weather, light intensity, sunlight direction and the activity which is happening in the room. For the weather, we can have some rain sensors to detect rain, and sunlight sensor to detect whether the sunlight enters the room or not.

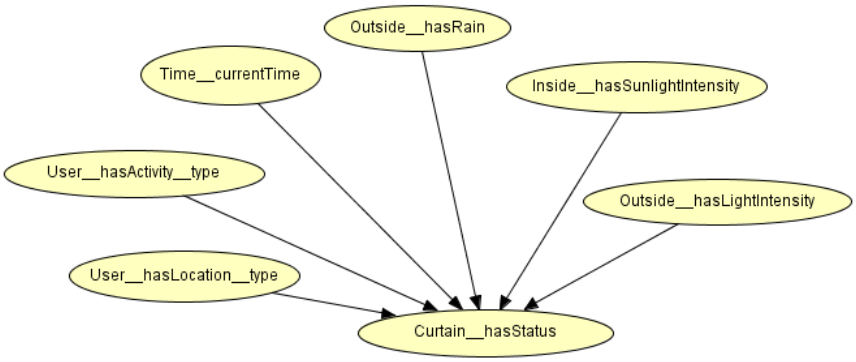


Figure 14: The curtain control preference model of one user

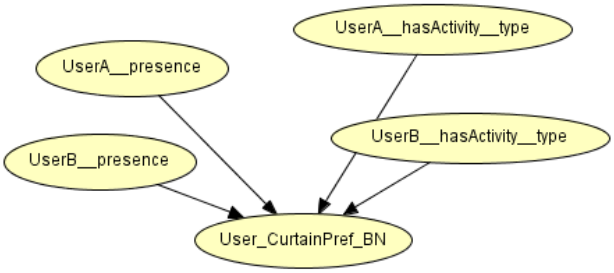


Figure 15: The priority model for curtain control preference (for 2 users) using traditional Bayesian network.

The priority of curtain control is related to the presence and the current activity of the user.

Initial the priority model:

To initialize the CPT, we can learn the conditional probability from the history data, or enter the probability manually, or use expression. This is an example of Hugin expression for the CPT of User_CurtainPref_BN node:

```
if (and (UserA__presence == "NO", UserB__presence == "NO"),
Distribution (0, 0, 1), Distribution (0.5, 0.5, 0))
```

The Distribution function in this expression is the distribution of the states of User_CurtainPref_BN node, which are BN1, BN2 and BN0. These states are the preference models of user A, user B and the model in case no user is presented.

This expression gives the equal priority for both users without concerning the current activity of each user. If we allow the user who is sleeping to have higher priority, the expression would be like this:

```
if (and (UserA__presence == "NO", UserB__presence == "NO"),
Distribution (0, 0, 1),
if (and (UserB__hasActivity__type == "SLEEPING",
 not (UserA__hasActivity__type == "SLEEPING")),
Distribution (0.2, 0.8, 0),
if (and (not (UserB__hasActivity__type == "SLEEPING"),
UserA__hasActivity__type == "SLEEPING"),
Distribution (0.8, 0.2, 0),
Distribution (0.5, 0.5, 0))))[2]
```

Using the function "Transfer to table" of Hugin [39], we will have a CP table, a part of which is showed in the following figure.

UserB__h...	WORKING								SLEEPING							
UserA__h...	WORKING				SLEEPING				MOVING				WORKING			
UserA__pr...	YES		NO		YES		NO		YES		NO		YES		NO	
UserB__pr...	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO
BN1	0,5	0,5	0,5	0	0,8	0,8	0,8	0	0,5	0,5	0,5	0	0,2	0,2	0,2	0
BN2	0,5	0,5	0,5	0	0,2	0,2	0,2	0	0,5	0,5	0,5	0	0,8	0,8	0,8	0
BN0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 16: CPT of the curtain control priority model

Light Control Preference

Most lights have 2 common states: ON and OFF. Assume that there are 4 levels of light intensity in the room: BRIGHT, NORMAL, DIM and DARK. The intensity levels which each light can produce range among BRIGHT, NORMAL and DIM.

The light control preference of one user will be related to the current activity of all the users involved, current light intensity in the room, outside light intensity and the curtain status. All the lights which have the same affective range with the user location will be considered in the model.

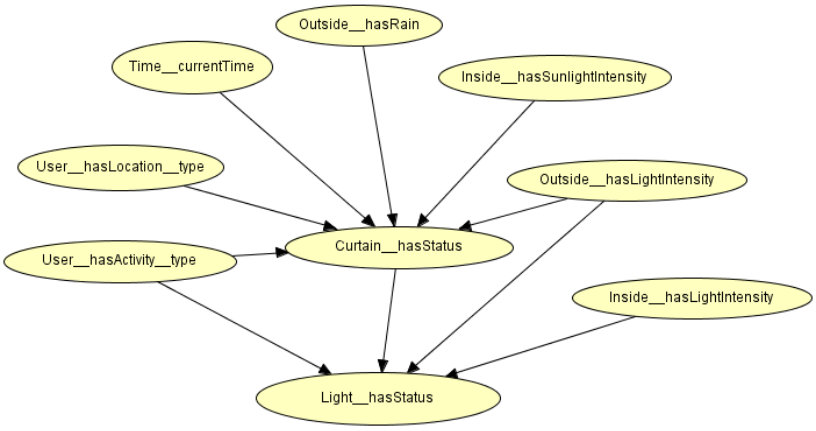


Figure 17: The Bayesian network models for the light control preference of one user

[2] The expression is in the format of Hugin expression, building by the Hugin tool [39].

The priority of the light intensity control also depends on the current activity of user. For example if the user is sleeping, light should not be bright.

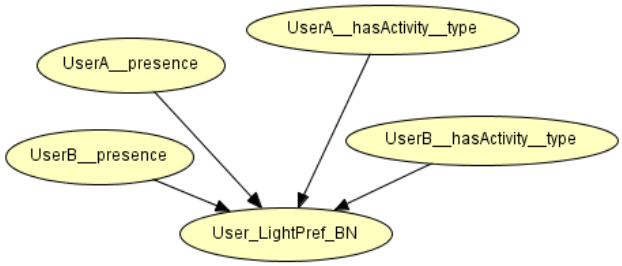


Figure 18: The priority model for curtain control preference (for 2 users) using traditional Bayesian network.

The initial of priority model is similar to the previous session.

Temperature Preference

Every air conditioner has its own types of control. However, the control can be mapped into some certain level of temperature. Assume that there are 3 levels: High, Moderate and Low.

The temperature preference of user depends on inside and outside temperature, user activity and the occupation duration. The temperature preference priority depends on the occupation duration. A user who has just been walking outside needs the help of the air conditioner or the heater more than a user who has stayed in the room for a long time.

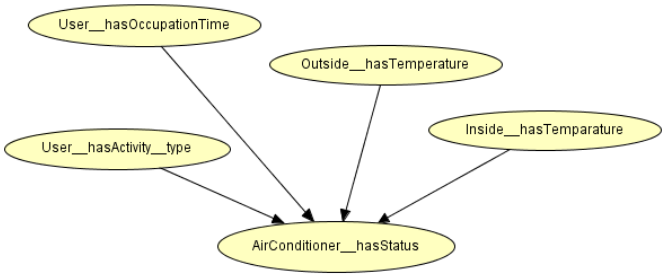


Figure 19: The Bayesian network models for the air conditioner control preference of one user

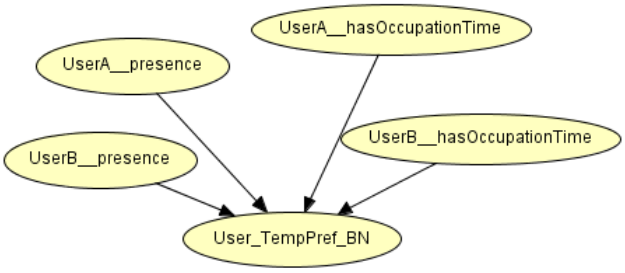


Figure 20: The priority model for air conditioner control preference (for 2 users) using traditional Bayesian network.

5.7.2 Adding more user preference models into a Bayesian RN-Metanetwork

When a new user joins the system, the preference model should be updated. The process for integrating a user into a multi-user preference model has 3 steps:

Step 1 (optional): create the preference model for that user based on the common preference model for a single user.

From the common preference model, we make a copy and then rename the nodes related to user to the name of that user.

For example, the air conditioner control preference model for UserA will be:

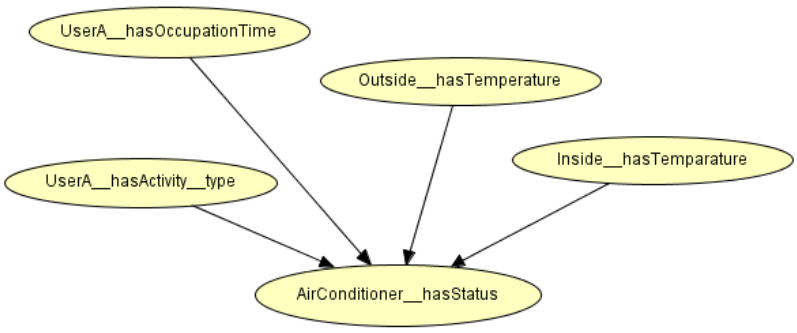


Figure 21: Air conditioner control preference model of UserA

This step can be omitted if the user preference model has already built before.

Step 2: add the new preference model into the set of preference model in the predictive level (first level) of the Bayesian RN-Metanetwork.

Step 3: update the priority model:

- Add the nodes related to the new user into the priority model

- Add one more state which is equivalent to the new user preference model

- Update the CPT:

+ Where new user presence has value NO, the conditional probability can remain the same, and 0 is filled for the probability of the new state.

$$P(BNN|X,Y,PrN=NO) = 0$$
$$P(BNi|X,Y,PrN=NO) = Po(BNi|X)$$

with N denotes the current number of users,
X denotes the set of nodes which related to old users
Y denotes the set of nodes which are related to new user except the presence,
PrN denotes the Presence of user number N,
P: new conditional probability

Po: old conditional probability

+ Where the new user presence has value YES, the conditional probability is recalculated. Normally the calculation is based on the priority mechanism of the system. If the system is first-equal-priority, i.e. all at the beginning all the users have same priority, the calculation will be:

New state:

$$P(BN_N|X,Y,Pr_N=YES) = \frac{1}{N}$$

Old state: the old priority order among the old users is maintained by:

$$P(BN_i|X,Y,Pr_N=YES) = P(BN_i|X)*\frac{N-1}{N}$$

5.7.3 Bayesian RN-Metanetwork Reasoner Implementation

The Bayesian RN-Metanetwork is implemented as a reasoner inside Auto-CAMUS.
The programming language is Java. The library of Hugin [39] is utilized as a core for Bayesian network computations.

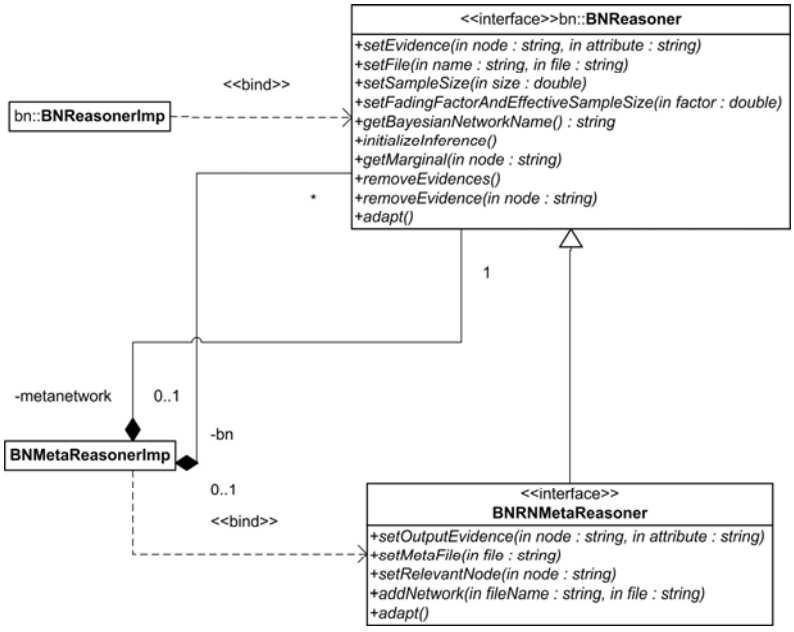


Figure 22: Class diagram of Bayesian RN-Metanetwork

By wrapping the Bayesian RN-Metanetwork in an interface which inherits the BNReasoner interface, the Bayesian RN-Metanetwork reasoners as well as the traditional Bayesian reasoner can be accessed in a unified manner. Because a Bayesian RN-Metanetwork includes the BNReasoners as its metanetwork and prediction-level networks, each of the sub-network can be a Bayesian RN-Metanetwork itself, or in other words, the Bayesian RN-Metanetwork can be multiplied into many layers.

5.7.4 User Preference learning implementation in Auto-CAMUS

We have a ReasonerManager service to take care of creating the reasoners as well as managing those reasoners. A ContextDataManager manages the context repository, receives the queries and registrations for context data. Inside the ContextDataManager, the rule-based reasoning is implemented to infer the high-level context data from low-level context data. When new context data comes from sensors, the ContextDataManager service does the inference for high-level context data, then matches the registrations and notifies the handler services if the registrations are matched.

A PreferenceContextAggregator service works as the preference learning module for the middleware. It reads the model files, calls the ReasonerManager service to create the Bayesian RN-Metanetwork reasoner, and registers to the ContextDataManager service for the needed context data. When it is notified about the new context data, it will give the new evidence for the Bayesian RN-Metanetwork reasoner by setEvidence method, then asks the reasoner to do inference and finally gets back the marginal for the nodes related to the preferences.

Any application can send request to PreferenceContextAggregator service to get the preference of user. After making the decision and receive the feed back from user, the application will tell the PreferenceContextAggregator service the result of feedback. Based on user feedback (approval or denial), PreferenceContextAggregator service will set the evidence and call the Bayesian RN-Metanetwork reasoner to do adaptation.

Fig. 23 shows the sequence diagram of all process.

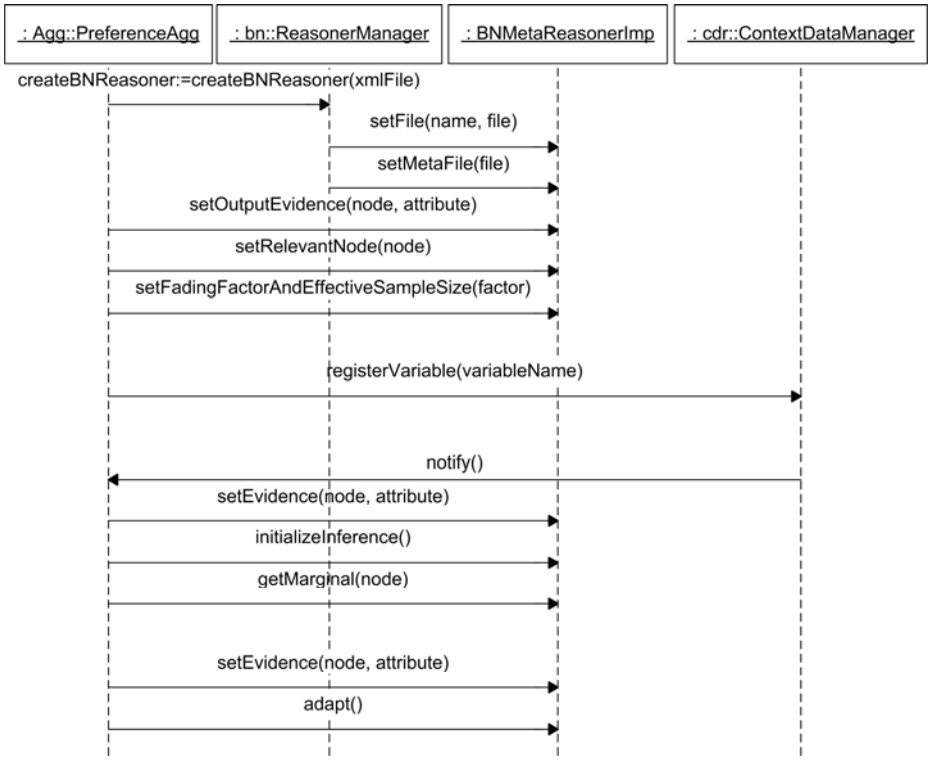


Figure 23: The sequence diagram of user preference learning process in Auto-CAMUS

5.8 Conclusion

Context-aware computing poses interesting issues for information system researches. Among those issues, learning user preference in order to adapt the system automatically to the need of user is a crucial task. Many challenges have risen in this are due to the uncertain, heterogeneous, distributing characteristic of a context-aware system. Especially when there are many users involve in an intelligent environment, the system has to cope with conflict resolution and distinguishing among the user preferences. A solution for learning user preference in a multi-user context-aware environment which can efficiently resolve the above mentioned problems is the main contribution of our work.

We have first presented the Bayesian RN-Metanetwork which can be used to model the user preference as well as user priority for many users, while still maintain separate preference model for each user. The propagation algorithm showed how to calculate the composite preference of all the users in the system and make decision of service to provide. To actively and continuously adapt the models to the newest preferences and priorities of the users, the adaptation algorithm for Bayesian RN-Metanetwork was described. To illustrate the use of Bayesian RN-Metanetwork, some common models for various kinds of user preference in a smart home system were presented.

However, this is just the first step to make the system intelligent. There are still a lot of challenges such as user behavior routine learning, in which the prediction of the future actions of user is the most important task.

Chapter 6

LOCATION AWARE COMPUTING

6.1 Introduction

Position of client is very important parameter for mobile computing applications. Positioning using Wireless LAN received signal strength (RSS) is increasingly popular choice. Reliability and performance of WLAN based positioning systems is directly related to the correctness of radio map of the site. We present a distributed component architecture based radio map generation system to facilitate development of accurate and comprehensive radio maps for Location Aware systems. Besides, our system allows real time visualization of RSS of all available wireless base stations, giving more insight into the nature of wave propagation at different locations. Thus, providing an efficient way to analyze the properties of WLAN and achieve optimal coverage, enhancing the positioning system performance.

Widespread deployment of Wireless Local Area Networks (WLAN), so called WiFi (Wireless Fidelity), has given a rise to development of WLAN based Location awareness in modern context-aware ubiquitous computing environments. Most significant advantage of this method is pervasive coverage of WiFi and no special hardware requirment, enhancing the value of WLAN.

WiFi location awareness applications include, but not limited to, a wide range of services to the end user like automatic call forwarding to user's location, robotic global localization, and exploration and navigation tasks. [Andrew], Finder, Guiding and Escorting systems, liaison applications, location based advertisement and positing of entities in large warehouses.

WiFi signal strength (RSS) based Systems for Location Awareness require a mapping of spatial points of interest with received signal strength at that point. This technique is gaining increasing attention for indoor applications mainly because other popular techniques e.g. GPS do not perform well in indoor environments. This mapping is referred to as location map, radio map, location fingerprints or calibration data in various Location Awareness systems. [RADAR][Horus], [Pehlvan]. We will refer this entity as "radio map" in rest of this paper. Figure 1 shows typical methodology for generating radio map.

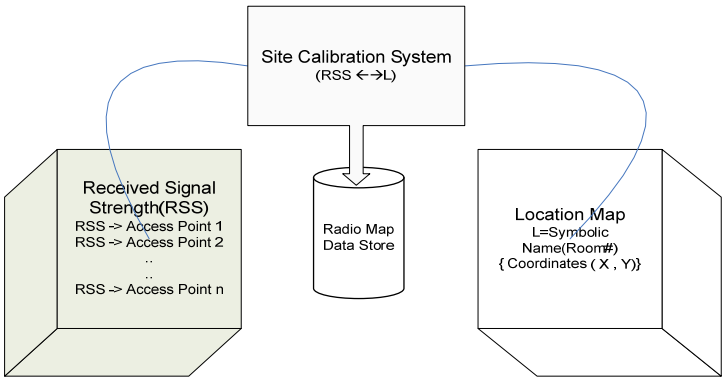


Figure1: Typical generation of radio map

The Received Signal Strength (RSS), carrier signal Phase of Arrival (POA), and Time of Arrival (TOA) of the received signal are the metrics used for building a radio map for later estimation of location. As a matter of fact, the accuracy and precision of location estimation is directly linked with the correctness of radio map. As the measurements of metrics become less reliable, the complexity of the position algorithm increases. Despite an extensive research being put into location estimation techniques, no dedicated effort is made to ensure correctness and evaluate the reliability of radio map.

6.2 Related Work

Recently, some researchers have proposed statistical manipulation of RSS data to reduce the effort and time required for building a radio map. [Microsoft] [Trento] [Trento2]
Binghao Li *et al* employ kriging interpolation used in mining variograms to reduce the number of calibration locations for building a radio map and used 12 samples for each location. [Binghao] [Horus] [HKUST] proposed a method of using unlabeled samples for reducing the sampling rate at each location and number of locations.

Despite reducing the number of calibration points, these techniques still need to calibrate the area. One subtle dependency of these approaches is that, they calibrate the denser set of reference points first, and then remove less influential points out them. It is still hard to calibrate only the few reference points without the knowledge of signal strength at un-calibrated points. Since radio map is foundation of any positioning technique, no work is suggested a systematic approach for site calibration using distributed component technology. Furthermore, since number of samples taken at each location and number of locations calibrated affect the accuracy of positioning technique, an extensive calibration of each point should be made possible unlike current approaches. Still, the need for having a reliable system to build such map is not yet met.

6.3 Our Approach

Nature of indoor radio wave propagation is very complex and non-deterministic due to multi-path effects, building structure, materials, and other devices using the same frequency spectrum as 802.11 protocol devices (2.4, 5 GHz). Moreover multiple temporal factors cause the RSS to fluctuate like activity, people, even opening and closing of doors during the day hours. RSS observations show that the signal strength tends to vary even at the same location due to above mentioned factors. Taking few samples at location is not sufficient enough for building a reliable location estimation technique for real world scenario. Insufficiently calibrated point might lead to misleading conclusions about the location at the time of positioning.

Most of the work has oversimplified the complex nature of radio wave propagation by using few samples of signal strength for a target location. It gets more hypothetical when one tries to interpolate un-calibrated locations using the insufficiently calibrated set of locations. Following table shows the extent to which different location estimation techniques have employed signal strength sampling.
Performance of Every location positioning system is tightly coupled with propagation characteristics in particular setup. That makes the positioning accuracy of a system strongly relative to particular environment. If any of the environmental factors or change in WLAN

layout parameters, rendering the reference radio map ineffective, can directly reduce the accuracy of positioning system. Thus, besides developing a positioning system, maintaining its precision and accuracy for long time, even in the face of significant environmental changes, becomes a critical issue. Therefore an accurate and comprehensive radio map is not only unavoidable and means to build such map should be easy so that it can be reconstructed when ever environmental factors change in a significant manner. Creating a denser radio map by interpolating sparsely calibrated locations implies critical trade offs between accuracy and precision, how much right and how often right, of Location estimation technique and complete radio map. This paper suggests facilitating the radio map building by using a sophisticated distributed system that can support multiple roaming devices to take part in calibration effort simultaneously.
In this paper we first elaborate the key requirements that such a system should provide to the Location aware system developers and secondly we present our Location-Ware tool kit for building such a system to achieve those requirements.

6.3.1 RF analyzer

Commercially available and open source like netstumbler and ministumbler [NetStumb] meet this requirement but these systems are more focused on network traffic analysis, intrusion detection and layout management. While radio map building require a subset of there capabilities, still these systems fall short of providing for capture, store and analysis requirements in a distributed Client Server architecture.

6.3.2 Distributed Architecture:

Most of the commercial WiFi surveying tools and are designed to run on single machine. Typically mobile terminals possess limited processing storage capabilities to support Site Surveying and Calibration components. Secondly, our concept of enabling multiple devices to simultaneously calibrate the site can be realized only in distributed component architecture.

6.3.3 Real Time Capture

Capturing wireless signal from available Access Points remains the fundamental task in building site calibration systems. Reliability of source of signal strength is vital for both building radio map and positioning technique. Careful attention should be payed to the signal capturing component's interfacing with Wireless NIC. Most of the commercially available wireless NICs provide received signal properties to the user mode applications. A standard library for signal capture should be employed that can hide different hardware vendor specific details of signal capture.

6.3.4 Real Time Visualization of RF Propagation

Visualizing the radio wave properties like Signal to Noise Ratio (SNR), Received Signal Strength (RSS), transmitter id (BSSID), transmitter channel etc is important for not only site surveying but calibration also. A very unique feature of indoor positioning applications is that the size of the coverage area is much smaller than outdoor applications. This makes it possible to conduct comprehensive planning of the placement of Wireless Access Points. Careful planning of a Wireless network can significantly reduce measurement errors of location metrics caused by NLOS propagation. The structural information of the Access Points can also be employed in intelligent positioning algorithms. Furthermore adjustment of frequency channels to be used by each Access Point in order to minimize the adjacent channel noise is very helpful

for maximum coverage. Furthermore graphical representation of radio wave shall give visual clues to mapping function between location and received signal properties. Therefore a real time visualization of captured radio signals presents a very useful capability of such systems.

6.3.5 Storage

When the area of coverage becomes large and a large number of sensors are involved, the size of the location signature database increases dramatically. [Pehlvan] Simultaneous calibration of site shall produce large amount of data. This magnitude depends on how detailed location map is required to be built. Two fundamental factors are the number of locations to be calibrated and the time period for which is location is calibrated. This requires a database server system to collect incoming data from all mobile devices and assemble it into a radio map. Schematic view of the calibration data that Location-Ware uses is presented in next section.

6.3.6 Support for Statistical Analysis of RF data:

Inside the building Radio Frequency (RF) propagation follows in a very complex model that depends on geometry, infrastructure, neighboring devices and activity going inside the building. Location predictive element of received radio signal is not as simple as triangulation in case of GPS positioning. Still at different locations, radio signal exhibits certain properties special to that location. Therefore only capturing the received signal strength is not enough for location estimation inside the buildings. It requires an extensive analysis of the properties and patterns that radio wave exhibits at a particular location. A sophisticated statistical analysis component of site calibration system is important property.

6.3.7 Interactive

As site surveying is physical task that requires a person to take the handheld device and note down the received signal strengths from all APs while roaming around the area. This whole process should be as easy possible. Interactivity of Site Calibration system is as important feature that effects quality of remote wireless data collection and integration into one whole. As all reference matrices at each location are going to be used by positioning techniques, the foundation for building tools should be user friendly and interactive.

6.4 System Architecture

Our System architecture follows distributed component approach to realize the concept of distributed simultaneous site calibration. Following block diagram shows entities of the system and core components of Mobile Unit (MU) devices and Stationary Unit (SU) workstation. Since component based approach is used in building up the whole system and each component provides specific feature while stand alone. This feature allows on demand addition or removal of the components from the system. That means MU can use only WiFi scanning component as classical site surveying tool. And combined with Visualization component charting of the WiFi signal patterns is possible. Similarly at the SU end, Statistical analysis component can be used for statistical analysis of any time series. And SU Visualization component provides charting of the radio map stored in the database. All components expose their functions through standard API (Application Programming Interfaces) that allows the programmer not only to plug in/out these components from there applications but also use its features without knowing the internal details of the implementation.

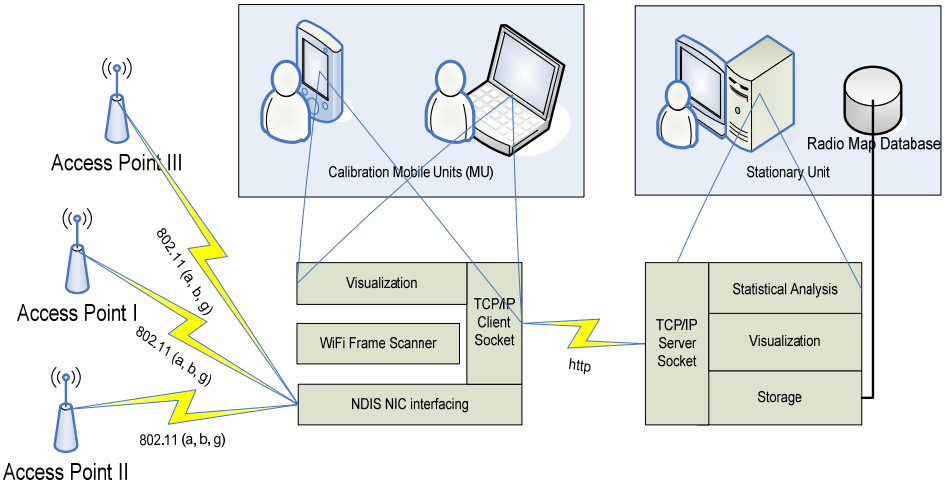


Figure: System Architecture of Distributed Wireless LAN Calibration System

In the next section we will present details about the core components involved in entities of this system at both ends.

6.5 Core Components

6.5.1 Mobile Unit System

For actual calibration task, we used TOSHIBA M-30 note book with built in Intel PRO/Wireless 2200 BG Network card. But these components can be deployed on other small handheld devices also. MU system is provided with NIC (Network Interface Card) interfacing, WiFi signal scanning capability, and device specific visualization of radio wave propagation components.

Wireless NIC Interfacing

All Mobile Units involved in calibration process need to interact with Wireless NIC in order to capture the wireless signal. We used NDIS (Network Driver Interface Specification) API for detecting the installed hardware on MU and load a suitable driver for it. Following figure shows the available drivers for installed NIC hardware on MU being used.

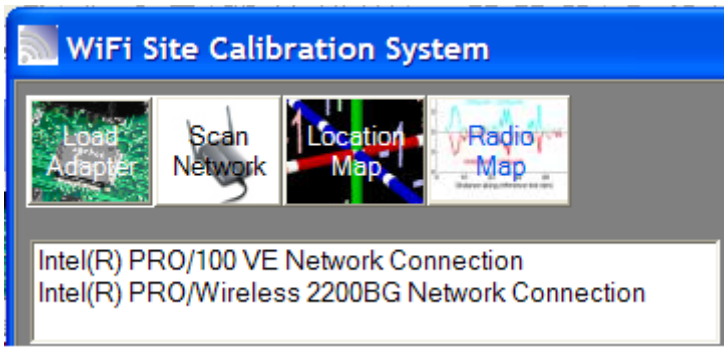


Figure: Available drivers for installed NIC hardware

WiFi Signal scanner

802.11 MAC (Media Access Control) layer specifies the mechanism for the radio NIC to scan the channels. This scanning can be done passively, by listening to the periodic broadcast or beacons of wireless access points, or actively by initiating the probe process itself. Our WiFi signal scanner component uses active mode of scanning that allows MU to initiate the probe and receive the response immediately. Radio NIC of MU broadcast a probe frame and all Access Points within range send probe response frame. The probe response frames contain information about the access point, including service set identifier, supported data rates, etc.

MAC driver then interprets the response frame and takes note of the corresponding signal strengths. Following figure shows the list of Wireless Access Points in range of MU after a probe.

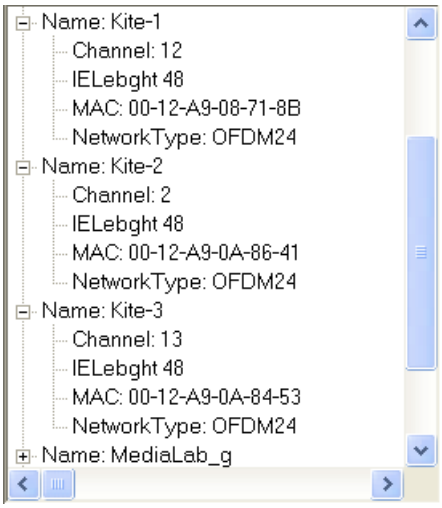


Figure: List of Wireless Access Points and their Configuration properties

We used rawehter API for Windows XP for capturing the wireless packets. It allows a user-mode application to "directly" access NDIS network interface card (NIC) drivers from Win32 applications. Rawether uses a NDIS protocol driver that acts as a "relay" between an application and the NDIS miniport driver. Network Driver Interface Specification (NDIS) is a common programming interface. Media-access controller (MAC) device drivers "directly" access NDIS network interface card (NIC) drivers from Win32 applications using a NDIS protocol driver.

Scanning Rate Controller

Active scanning mechanism allows MU to scan network at desired rate. Our scanner component exposes this capability through standard programming interface. Client mode applications can choose their preferred rate of network scanning. This capability is implemented in our system as GUI control.

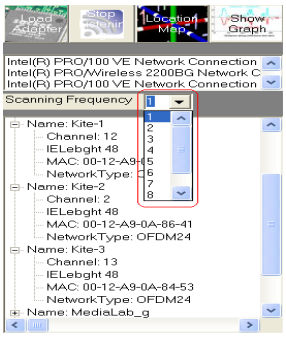


Figure: Controlling WiFi signal scanning rate

Visualization

An open source charting library ZedGraph is employed to visualize received signal strength of each Wireless Access Point. [ZedGraph] Following figure shows the pattern of received signal strength of WiFi signal. Each point of the graph correspond to the scanning frequency set by the user.

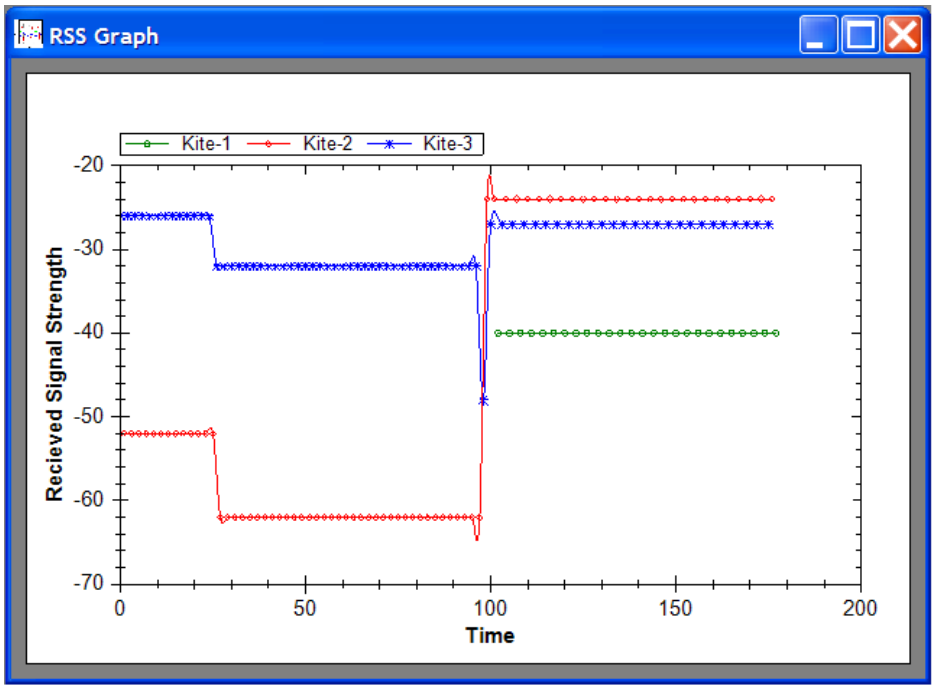


Figure: Received Signal Strength visualization

Location Map

Every MU calibration application is featured with active location map of the site. This map is provided to allow the user to pinpoint the current location of observation. Following figure shows experimental Location map used in our system. Location of all reference Access points is marked as red circle. And different calibration points are marked with green circles.

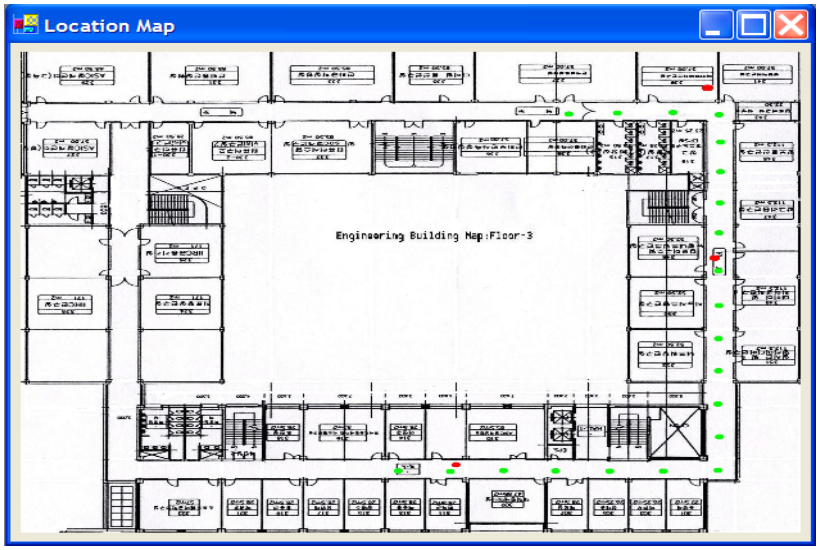


Figure: Location Map

Network Communication

WinSock is used to implement the MU side and Stationary system side network components [WinSock]. Peer to peer network model is considered for collaboration of MU and server work station. Both ends communicate using TCP/IP protocol using WinSock API. Network communication component on MU side registers itself with the SU and get and MU-ID response from SU. Later this ID is used to report WiFi signal information, acquired from all Access Points in range, back to the SU. At the SU side, network communication component registers every MU roaming in site and manages all the incoming information using multithreaded client management system. Asynchronous Communication model is implemented to allow multiple MU work simultaneously.

6.5.2 Stationary Unit workstation

On the other hand, stationary unit performs the important task of managing simultaneous incoming information of every MU.

Radio Location Map Schema

For representation of spatial co-ordinate's information with respective radio wave propagation characteristics in integrated form, we used following schema model.

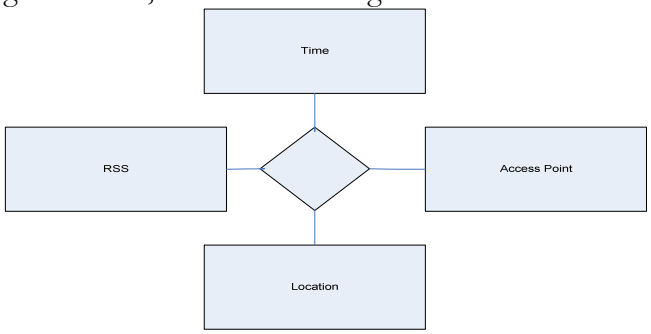


Figure: Schematic view of Radio location map

Statistical Analysis

As radio map does not serve any objective of Location-Aware System. Analysis of radio map data for, developing a mapping function between location and signal properties is an important task. Statistical Analysis component produce most of the important statistical measures and allows graphic representation of these measures. Measures of dispersion are used for analyzing the range and pattern of signal strength spread on a location and between two locations. Similarly measures of central tendency are used for generating smoothing filters, building different histograms and removing outliers. Moving average Smoothing Filters are applied to remove the temporal variations due to transitory environmental conditions like people movement. One important measurement that we compute is Fluctuation frequency of every Access Point signal at one location. This measure allows measuring the stability of signal strength relative to moving or stationary mobile unit and generating sophisticated probability distribution functions (p.d.f) for location to signal mapping function.

6.5.3 Data Storage

Currently the radio map data is stored in Excel spread sheets. As the volume of radio map increases, same schematic representation can be implemented in any DBMS like MySql.

6.6 Conclusion

Wireless site calibration for location awareness is fundamental task in the process of WiFi based positioning systems. Time and effort required for site calibration is major tradeoff for accuracy and precision of such systems.

Chapter 7

ARTIFICIAL NEURAL NETWORKS FOR LOCATION ESTIMATION

7.1 Introduction to Neural Network for Location Estimation

Interacting with noisy data or data from the environment, Non-stationary processing of data from a complex (e.g. chemical, manufacturing, or commercial) process, is very difficult using classical mathematical and engineering methods. Neural networks provide massive parallelism, fault tolerance, adaptation to circumstances. There may be an algorithm, but it is not known nor has too many variables. It is easier to let the neural network learn from examples.

7.1.1 Draw Back

It is virtually impossible to "interpret" the solution in traditional, analytic terms, such as those used to build theories that explain phenomena.

7.1.2 Issues in ANN

Two main training paradigms have emerged: batch learning, in which optimization is carried out with respect to the entire training set simultaneously, and on-line learning, where network parameters are updated after the presentation of each training example (which may be sampled with or without repetition). Although batch learning is probably faster for small and medium training sets and networks, it seems to be more prone to local minima and is very inefficient in the case of training large networks and for large training sets. On-line learning is also the more natural approach for learning non-stationary tasks, whereas batch learning would require re-training on continuously changing data sets.

- The main difficulty with on-line training is the sensitivity of most training methods to the choice of training parameters. This dependence may not only slow down training, but may also have bearing on its ability to converge successfully to a desired stable fixed point.
- Most advanced optimization methods (e.g., conjugate gradient, variable metric, simulated annealing etc) rely on a fixed error surface whereas on-line learning produces an inherently stochastic error surface.
- The Bayesian approach provides an efficient way of training and has been applied quite naturally and successfully within the framework of batch learning. Extensions to the on-line case, where explicit information on past examples is not stored, have been limited so far.

7.2 Candidate NN Structures

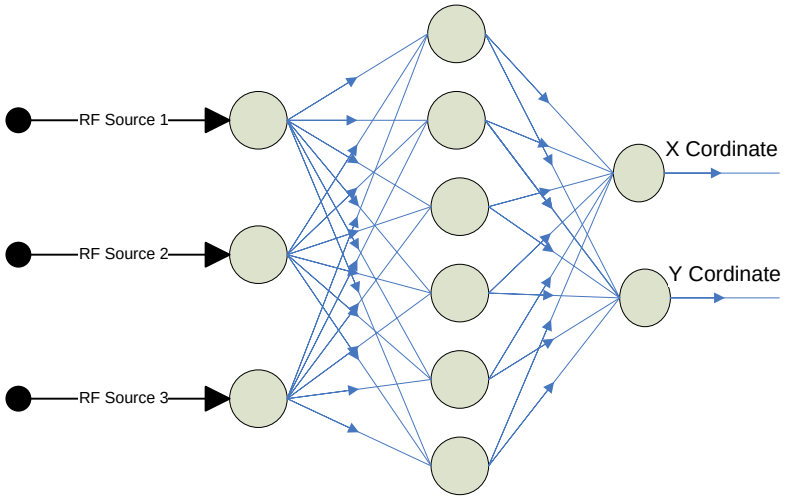


Figure: Candidate Neural net Structure 3-6-2

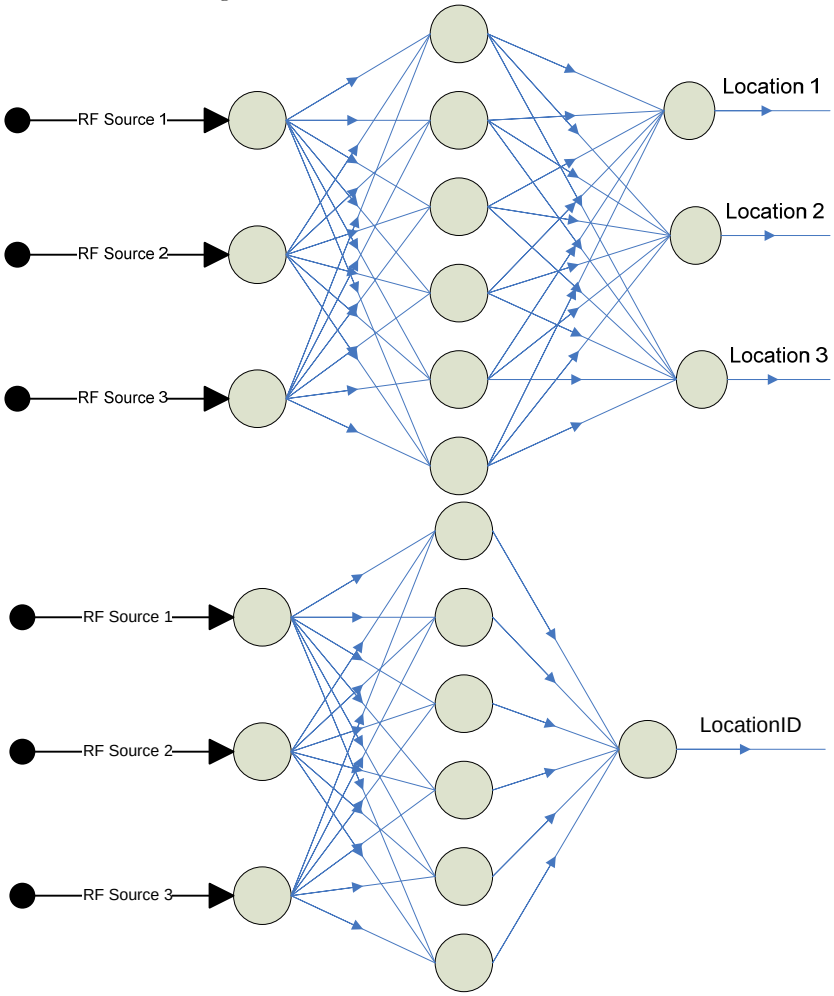


Figure: Candidate Neural net Structure 3-6-1

7.2.1 Activation Function

$$\mathbf{Sum} = \mathbf{Bias} + \sum_{i=0}^{(M-1)} \left(\mathbf{Weight}[i] * \mathbf{Input}[i] \right)$$

$$\mathbf{Output} = \mathbf{ActivationFunction}(\mathbf{Sum})$$

7.2.2 Feed Forward Back Propagation

"Back-Propagation" is a mathematical procedure that starts with the error at the output of a neural network and propagates this error backwards through the network to yield output error values for all neurons in the network.

A common form of learning is "trial and error". A "trial" is the output of a system in response to particular stimuli. An "error" is the external reaction to the output of the system that is supplied to the system as some other kind of stimulus. A system capable of "trial and error" learning relies on receiving feedback that describes the nature and severity of mistakes. The system can use the error information to make corrections in the way it responds to that particular combination of stimuli in the future. Back-Propagation yields neuron error values throughout a neural network. Learning occurs when neuron input weights and bias values are adjusted in an attempt to reduce the output error for the same stimuli. It should be noted that defining a mechanism for learning implicitly defines the nature of phenomena that will frustrate learning.

7.2.3 Back-Propagation Formulae

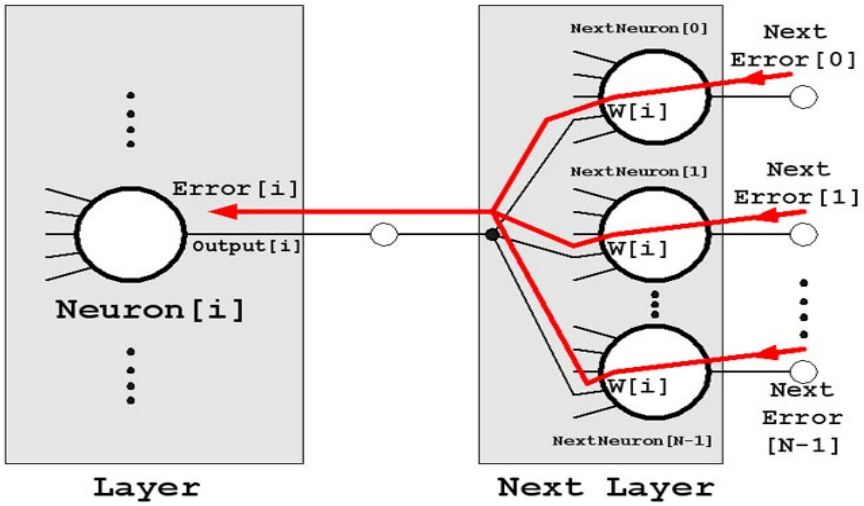
The goal is to compute output errors for every neuron in a network. The output errors for neurons at the output layer of a network is particularly easy to compute:

$$\mathbf{Error[i]} = (\mathbf{Output[i]} - \mathbf{Desired[i]})$$

The error value for a neuron in an arbitrary layer that is not an output layer is computed using the following formula:

$$\mathbf{Error[i]} = \mathbf{Output[i]} * (1 - \mathbf{Output[i]}) * \left(\sum_{k=0}^{(\mathbf{NextLayerNeurons}-1)} \left(\mathbf{NextNeuron[k].Weight[i]} * \mathbf{NextError[k]} \right) \right)$$

The following diagram illustrates concept expressed by the formula:



In a sense, all of the output errors at the next layer leak backwards through the input weights and accumulate at the output of a neuron in a previous layer. This accumulated value is multiplied by a value that is greatest when the current output of the neuron is most neutral (most "undecided") and is least when the output of the neuron is most extreme (very "certain").

7.2.4 Weight Change and Bias Change Formulae

The basis of learning is the adjustment of weights and bias values in an attempt to reduce future output errors. "Learning Rate" is a numerical value that essentially indicates how quickly a neuron adjusts weight and bias values according to error values. The following formula indicates how to change the weights of a neuron with a particular set of input values and its output error value:

```
WeightChange[i] =
  (-(Rate * Input[i] * Error))
```

FIGURE: Weight changes for a neuron.

The following formula indicates how to change the bias of a neuron given the current output error for the neuron:

```
BiasChange = (-(Rate * Error))
```

FIGURE: Bias change for a neuron.

7.3 Modeling Data and Training Vectors

Training Vectors are a subset of calibration data. Training Vectors are selected very carefully to make Neural Network generalize for cases of signal strength instances.

7.3.1 Artificial Noise

Noise in the actual data is never a good thing, since it limits the accuracy of generalization that can be achieved no matter how extensive the training set is. On the other hand, injecting

artificial noise (jitter) into the inputs during training is one of several ways to improve generalization for smooth functions when you have a small training set.

7.3.2 Permutations

Probability distributions of RSS of each AP at each location are available as a result of calibration process. Permutations are generated from these probability distributions to cover all the possible combinations. This allows the neural network to generalize for all possible RSS values at a given location.

Normalization

Before training, it is often useful to scale the inputs and targets so that they always fall within a specified range. The function premnmx can be used to scale inputs and targets so that they fall in the range [-1,1]. The following code illustrates the use of this function.
[pn,minp,maxp,tn,mint,maxt] = premnmx(p,t);
net=train(net,pn,tn);
The original network inputs and targets are given in the matrices p and t. The normalized inputs and targets, pn and tn, that are returned will all fall in the interval [-1,1]. The vectors minp and maxp contain the minimum and maximum values of the original inputs, and the vectors mint and maxt contain the minimum and maximum values of the original targets. After the network has been trained, these vectors should be used to transform any future inputs that are applied to the network. They effectively become a part of the network, just like the network weights and biases. If premnmx is used to scale both the inputs and targets, then the output of the network will be trained to produce outputs in the range [-1,1]. If you want to convert these outputs back into the same units that were used for the original targets, then you should use the routine postmnmx. In the following code, we simulate the network that was trained in the previous code, and then convert the network output back into the original units.
Algorithm: pn = 2*(p-minp)/(maxp-minp) - 1;
Another approach for scaling network inputs and targets is to normalize the mean and standard deviation of the training set. This procedure is implemented in the function prestd. It normalizes the inputs and targets so that they will have zero mean and unity standard deviation. The following code illustrates the use of prestd.

7.4 Training / Tuning

Following block diagram shows the system components involved in training and tuning of neural network.

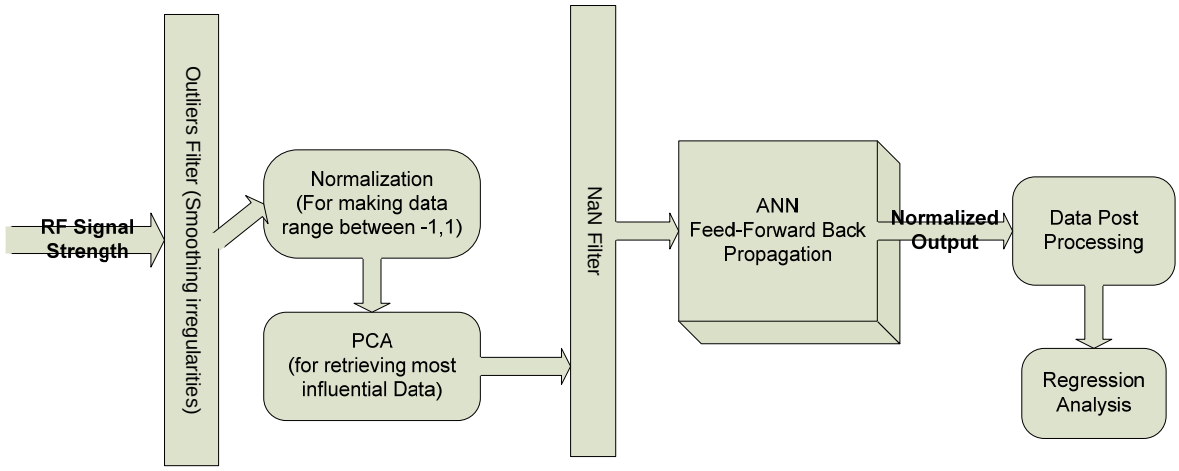


Figure: Location-Aware ANN Training Phase

7.4.1 Learning Methodology

Two different methodologies are suggested in literature; Offline and Online. Since we collect training data in a separate phase of development and it is available as a component, we apply offline training phase to train neural networks.

7.4.2 Variable Learning Rate (traingda, traingdx)

With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface. The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning rate will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface. An adaptive learning rate requires some changes in the training procedure used by traingd. First, the initial network output and error are calculated. At each epoch new weights and biases are calculated using the current learning rate. New outputs and errors are then calculated. As with momentum, if the new error exceeds the old error by more than a predefined ratio max_perf_inc (typically 1.04), the new weights and biases are discarded. In addition, the learning rate is decreased (typically by multiplying by lr_dec = 0.7). Otherwise, the new weights, etc., are kept. If the new error is less than the old error, the learning rate is increased (typically by multiplying by lr_inc = 1.05). This procedure increases the learning rate, but only to the extent that the network can learn without large error increases. Thus, a near-optimal learning rate is obtained for the local terrain. When a larger learning rate could result in stable learning, the learning rate is increased. When the learning rate is too high to guarantee a decrease in error, it gets decreased until stable learning resumes. Try the Neural Network Design Demonstration nnd12vl [HDB96] for an illustration of the

performance of the variable learning rate algorithm. Backpropagation training with an adaptive learning rate is implemented with the function traingda, which is called just like traingd

7.5 Post Processing

The network output 'an' will correspond to the normalized targets 'tn'. The un-normalized network output 'a' is in the same units as the original targets't'. If premnmx is used to preprocess the training set data, then whenever the trained network is used with new inputs they should be preprocessed with the minimum and maximums that were computed for the training set. This can be accomplished with the routine tramnmx. In the following code, we apply a new set of inputs to the network we have already trained.

```
pnewn = tramnmx(pnew,minp,maxp);
anewn = sim(net,pnewn);
anew = postmnmx(anewn,mint,maxt);
```

7.6 Testing Validation

For testing and Validation extensive site calibration data is used. A permutation generating component is planned to allow generating all possible sets of data in required amount. Once this component is available, more exhaustive testing of Neural Net shall be possible.

7.7 Execution

Neural network is developed and trained in MATLAB. After testing and validation of results, it is required to be implemented as a software system. Following is the block diagram of how different components shall collaborate for making such system.

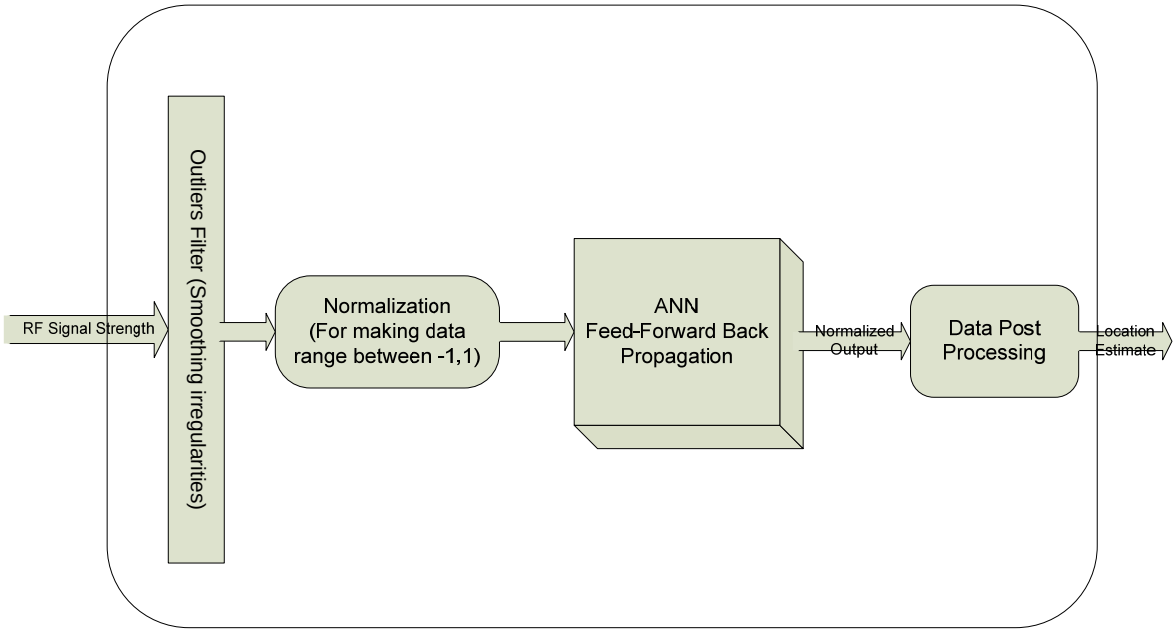


Figure: Location-Aware ANN Execution Phase

7.8 Conclusion

This work is in progress. After implementing the software ANN system, different other neural network architectures, recurrent models, shall be considered.

APPENDIX

Appendices A: VB Component code for Statistical Analysis

```
Attribute VB_Name = "Module1"
Sub Frequency()
'
' Macro recorded by Uzair Ahmad
'
    Dim pre, nex, i, Min, Max, Var, SD, Jmpngfrqncy

    Cells(1, 8).Value = "JF"
    Cells(1, 9).Value = "Variance"
    Cells(1, 10).Value = "StDev"
    Cells(1, 11).Value = "Min"
    Cells(1, 12).Value = "Max"

    Cells(5, 7).Value = "Bins_Kite_1"
    Cells(5, 8).Value = "FD_Kite_1"
    Cells(5, 9).Value = "Bins_Kite-2"
    Cells(5, 10).Value = "FD_Kite_2"
    Cells(5, 11).Value = "Bins_Kite_3"
    Cells(5, 12).Value = "FD_Bins_Kite"

    i = Range("Kite_1").Rows.Count
    For Count = 2 To i
      pre = Cells(Count, 4).Value
      nex = Cells(Count + 1, 4).Value
      If (pre <> nex) Then
        Jmpngfrqncy = Jmpngfrqncy + 1
      End If
    Next Count
    Cells(2, 8).Value = Jmpngfrqncy / i * 100

    Cells(2, 9).Value = "=Var(Kite_1)"
    Var = Cells(2, 9).Value
    Cells(2, 10).Value = "=Stdev(Kite_1)"
    SD = Cells(2, 10).Value
    Cells(2, 11).Value = "=Min(Kite_1)"
    Min = Cells(2, 11).Value
    Cells(2, 12).Value = "=Max(Kite_1)"
    Max = Cells(2, 12).Value

    Cells(6, 7).Value = Max
    i = (Abs(Min) - Abs(Max)) / SD
    For Count = 7 To 18
      If (Cells(Count - 1, 7).Value <= Min) Then
        GoTo Here
      Else
Cells(Count, 7).Value = Cells(Count - 1, 7).Value - SD
      End If
    Next Count
```

```
Here:
    Range("G6:G" & Count - 1).Name = "kite_1_Bins"
    Range("H6:H" & Count - 1).Name = "kite_1_FD"
    Range("H6:H" & Count - 1).FormulaArray = ("=Frequency(kite_1, kite_1_Bins)")

    i = Range("Kite_2").Rows.Count

    For Count = 2 To i
      pre = Cells(Count, 5).Value
      nex = Cells(Count + 1, 5).Value
      If (pre <> nex) Then
        Jmpngfrqncy = Jmpngfrqncy + 1
      End If
    Next Count
    Cells(3, 8).Value = Jmpngfrqncy / i * 100
    Cells(3, 9).Value = "=Var(Kite_2)"
    Cells(3, 9).Value = "=Var(Kite_2)"
    Var = Cells(3, 9).Value
    Cells(3, 10).Value = "=Stdev(Kite_2)"
    SD = Cells(3, 10).Value
    Cells(3, 11).Value = "=Min(Kite_2)"
    Min = Cells(3, 11).Value
    Cells(3, 12).Value = "=Max(Kite_2)"
    Max = Cells(3, 12).Value

    Cells(6, 9).Value = Max
    i = (Abs(Min) - Abs(Max)) / SD
    For Count = 7 To 18
      If (Cells(Count - 1, 9).Value <= Min) Then
        GoTo There
      Else
        Cells(Count, 9).Value = Cells(Count - 1, 9).Value - SD
      End If
    Next Count

There:
    Range("I6:I" & Count - 1).Name = "kite_2_Bins"
    Range("J6:J" & Count - 1).Name = "kite_2_FD"
    Range("J6:J" & Count - 1).FormulaArray = ("=Frequency(kite_2, kite_2_Bins)")

    'Selection.FormulaArray =
End Sub
```

Appendices B: Capturing the wireless Signal Code

```
private void listen() {
// Make a call to get a BSSID list
int excelRowCount = 2;
int graphcount = 0;
if(btnListenRadio.Text == "Scan Network")
{
// change btn label
btnListenRadio.Text = "Stop Scanning";
stopListening = false;
// listen till stoplistening is false
```

```
do
{
BssidList bssidList = new BssidList();
uint returnValue = adapter.QueryInformation( bssidList );

if( returnValue == 0 )
{
foreach( BssidListItem bssidItem in bssidList )
{
string apName = bssidItem.Ssid;
string ss = bssidItem.Rssi.RSSI+ "";
if (worksheet == null)
{
MessageBox.Show("Excel sheet can not be created");
return;
}
if( (apName == "Kite-1") |
(apName == "Kite-2") |
(apName == "Kite-3") )
{
worksheet.Cells[excelRowCount,1] = excelRowCount+"";
worksheet.Cells[excelRowCount,2] = DateTime.Now.Hour +":"+ DateTime.Now.Minute +":"+
DateTime.Now.Second;
if(locationChanged == true){
locationChanged = false;
worksheet.Cells[excelRowCount,3] = mapForm.getLocation();
}
if(apName == "Kite-1")
worksheet.Cells[excelRowCount,4] = ss;
if(apName == "Kite-2")
worksheet.Cells[excelRowCount,5] = ss;
if(apName == "Kite-3")
worksheet.Cells[excelRowCount,6] = ss;

graphWnd.updateGraph(Convert.ToDouble(graphcount++),Convert.ToDouble(ss),apName);
if (graphcount == 200)
{
graphcount = 0;
graphWnd.clearGraph();
}
}
}
}
else
MessageBox.Show("  BSSID List: Query Failed; Status: {0}"+" PcaUsa.Ndis.Status" +" "+ returnValue );
excelRowCount ++;

System.Threading.Thread.Sleep(Convert.ToInt16(comboSamplingFreq.Text)*1000);
}while(! stopListening);
return;
}
}
```

Appendices C: Data Storage Component

```
private void intializeExcel(){
```

```
try
{
workbooks = excel.Workbooks;
workbook = workbooks.Add(Microsoft.Office.Interop.Excel.XlWBATemplate.xlWBATWorksheet);
sheets = workbook.Worksheets;

worksheet = (Microsoft.Office.Interop.Excel.Worksheet) sheets.get_Item(1);
if (worksheet == null)
{
Console.WriteLine ("ERROR: worksheet == null");
}
}
catch(Exception ex)
{
Console.WriteLine(ex.Message);
}
```

Appendices D: Loading the NDIS Driver Code

```
// Create an Instance of the Desired Rawether Driver Object
// -------------------------------------------------------
// This may throw exceptions if the specified driver is not present or has faults.
PcaUsa.Ndis.BssidList bssidList = new PcaUsa.Ndis.BssidList(500);
uint itemsCount = bssidList.NumberOfItems;
bssidList.GetEnumerator();
PcaUsa.Rawether.Drivers.DriverObject ndisDriver = null;
try
{
ndisDriver = new PcaUsa.Rawether.Drivers.PcaMprProtDriver();
}
catch( PcaUsa.Rawether.Exceptions.DriverNotPresent )
{
MessageBox.Show( "NDIS driver is not present");
return;
}
// Create The Adapter Collection
PcaUsa.Rawether.Adapters.AdapterCollection adapterList = ndisDriver.GetAdapterCollection();
// Perform The Adapter Enumeration
foreach( BindingName adapterName in adapterList )
{
// Display AdapterName Information
listNDISDrivers.Items.Add((object)adapterName.InstanceName);

PcaUsa.Rawether.Adapters.AdapterObject adapter;
// Open an AdapterObject on the found AdapterName
try
{
adapter = ndisDriver.OpenAdapter( adapterName );
}
catch( PcaUsa.Rawether.Exceptions.OpenAdapterFailedException )
{
MessageBox.Show( "OpenAdapter: FAILED." );
continue;
}

if (adapterName.InstanceName == "Intel(R) PRO/Wireless 2200BG Network Connection")
{
```

```
this.adapter = adapter;
PcaUsa.Ndis.MediaState connectState = PcaUsa.Ndis.MediaState.Unknown;
adapter.QueryInformation( ref connectState );
// NOTE: Rawether .NET reports "Disconnected" only if the adapter actually supports
// the OID_GEN_MEDIA_CONNECT_STATE query. If the OID is not supported the state
// will be "Unknown".
if( connectState == PcaUsa.Ndis.MediaState.Disconnected )
{
MessageBox.Show("  This AdapterObject is in Disconnected state" + "  It is probable that no packets will be
received" +
"   * Skipping test receive test on this adapter *" );
//adapter.CloseAdapter();
continue;
}
else if( connectState == PcaUsa.Ndis.MediaState.Unknown )
{
MessageBox.Show( "  This AdapterObject is in UNKNOWN connect state" + "  It is possible that no packets
will be received  Continuing receive test on this adapter, but it may hang..." );
}
//adapter.CloseAdapter();
break;
}
// Get Media Connect State
}
```

References:

[NetStumb] http://stumbler.net/index.php

[Microsoft] John *et al* ,Minimizing Calibration Effort for an Indoor 802.11 Device Location Measurement System, November 13, 2003, Technical Report, MSR-TR-2003-82 Microsoft Research

[Trento] Roberto Battiti *et al*, Statistical Learning Theory For Location Fingerprinting In Wireless Lans, October 2002 Technical Report # DIT-02-0086 University of Trento Italy

[HKUST] Reducing the Calibration Effort for Location Estimation Using Unlabeled Samples, Proceedings of the 3rd IEEE Int'l Conf. on Pervasive Computing and Communications (PerCom 2005)
0-7695-2299-8/05

[Trento2] Mauro Brunato, Csaba Kiss Kalló, "Transparent Location Fingerprinting for Wireless Services," in Proceedings of Med-Hoc-Net 2002.

[Binghao] LI, B., WANG, Y., LEE, H.K., DEMPSTER, A.G., & RIZOS, C., 2005. A new method for yielding a database of location fingerprints in WLAN. *IEE Proc. Communications*, 152(5), 580-586

[RADAR] Paramvir Bahl *et al,* A Software System for Locating Mobile Users:Design, Evaluation, and Lessons Microsoft Research http://research.microsoft.com/~padmanab/papers/radar.pdf

[Pehlvan] K. Pehlavan *et al,* "Indoor Geolocation Science and Technology", IEEE Communications Magazine, 2002

[Andrew] Andrew *et al*, "Using Wireless Ethernet for localization" 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Lausanne, Switzerland), September 2002.

Mauro Brunato, Csaba Kiss Kalló, "Transparent Location Fingerprinting for Wireless Services," in Proceedings of Med-Hoc-Net 2002.

[Horus] *Moustafa et al,* **The Horus WLAN Location Determination System ,**Proceedings of the Third International Conference on Mobile Systems, Applications, and Services (MobiSys 2005), Seattle, WA, USA, June 2005

[ZedGraph] http://zedgraph.sourceforge.net/

[NDIS] http://www.ndis.com/

[WinSock]http://msdn.microsoft.com/library/?url=/library/ens/winsock/winsock/windows_sockets_start_page_2.asp