THE DESIGN AND IMPLEMENTATION OF FUZZY QUERY PROCESSING ON SENSOR NETWORKS

by

Marguerite Doman

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Information Technology

Charlotte

2009

Approved by:

Dr. Teresa Dahlberg

Dr. Yu Wang

Dr. Jamie Payton

Dr. Asis Nasipuri

Dr. Susan Sell

Dr. Cem Saydam.

ABSTRACT

MARGUERITE DOMAN. Design, implementation, and evaluation of fuzzy query processing for data management in sensor networks. (Under the direction of DR. TERESA A. DAHLBERG)

Sensor nodes and Wireless Sensor Networks (WSN) enable observation of the physical world in unprecedented levels of granularity. A growing number of environmental monitoring applications are being designed to leverage data collection features of WSN, increasing the need for efficient data management techniques and for comparative analysis of various data management techniques. My research leverages aspects of fuzzy database, specifically fuzzy data representation and fuzzy or flexible queries to improve upon the efficiency of existing data management techniques by exploiting the inherent uncertainty of the data collected by WSN. Herein I present my research contributions. I provide classification of WSN middleware to illustrate varying approaches to data management for WSN and identify a need to better handle the uncertainty inherent in data collected from physical environments and to take advantage of the imprecision of the data to increase the efficiency of WSN by requiring less information be transmitted to adequately answer queries posed by WSN monitoring applications.

In this dissertation, I present a novel approach to querying WSN, in which semantic knowledge about sensor attributes is represented as fuzzy terms. I present an enhanced simulation environment that supports more flexible and realistic analysis by using cellular automata models to separately model the deployed WSN and the underlying physical environment. Simulation experiments are used to evaluate my fuzzy

query approach for environmental monitoring applications. My analysis shows that using fuzzy queries improves upon other data management techniques by reducing the amount of data that needs to be collected to accurately satisfy application requests. This reduction in data transmission results in increased battery life within sensors, an important measure of cost and performance for WSN applications.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my committee chair and Ph.D. advisor Dr. Teresa Dahlberg. Her genius and commitment to excellence challenged me through the research work. Her support and enthusiasm encouraged me through the difficult moments. Dr. Teresa Dahlberg has been and will continue to be an inspiration to me. Without her guidance this dissertation would not be possible.

I would like to thank Dr. Jamie Payton, who has contributed greatly to my research work. Her technical expertise and advice played an important role in this dissertation.

I would like to thank my committee members, Dr. Susan Sell, Dr. Yu Wang, Dr. Asis Nasiburi, and Dr. Cem Saydam. I appreciate their advice and support.

Many thanks go to former and current members of Networking Research Laboratory, Lijuan Cao, Fan Li, Kashif Sharif and DingXiang Liu. I have enjoyed our collaboration very much.

I am especially grateful for my family: to my mother, Virginia, who celebrates me through all my successes and failures; to my sister, Kathleen, who supported me with encouragement and shelter through this endeavor; to my sister Barbara who drove me to campus that first day and continues to help me; to Cecilia and Tina for picking up the slack at home; to Karen and Kani for the gentle and 'not so gentle' pushes that kept me on track; to Linda, Jack, Matt, Joe, Anne and Sandy for generously supporting my mid-life decision; to my father, Jack, and my aunt, Marguery for their inspiration. Finally, to my sister Anne, I wish a "Happy Birthday".

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1: INTRODUCTION

Sensor nodes and Wireless Sensor Networks (WSN) enable observation of the physical world in unprecedented levels of granularity. Sensor nodes are compact, autonomous devices with a built-in processing unit, energy storage, communication capabilities, and limited data storage [28, 29, 77, and 15]. They also include sensing hardware to collect local environmental conditions. Through wireless communication, these sensor nodes dynamically form ad hoc networks and co-operate with each other allowing a real-time understanding of environmental changes and patterns. A growing number of environmental monitoring applications are being designed to exploit the data collection features of wireless sensor networks (WSN). Typical WSN application domains include [14]: 1) military tasks, such as intrusion detection or landmine detection; 2) environmental monitoring for ecological research, such as the deployment of a sensor network on Great Duck Island [23] and in ZebraNet [30, 32] to monitor wildlife and their habitat without the intrusion effects of human observation; and 3) commercial or human centric applications, such as managing energy efficiency of indoor heating and cooling systems [27]. In many monitoring applications, sensor nodes are deployed in the field, operating on batteries, with limited accessibility to recharge or replace batteries, and applications collect sensed data by interfacing with WSN data management middleware.

These WSN applications suggest that the utility of a sensor network is primarily

in its ability to gather and communicate collected data. As such, applications designed for WSNs tend to be data-centric, viewing the network as a single data space [2], and requiring sophisticated queries [1, 14]. To better understand the requirements and application view of the data collected in WSN, I developed a holistic classification of WSN middleware with respect to the varying approaches of how data is managed and queried. My survey identifies an opportunity to improve upon these approaches by taking into consideration the data uncertainty inherent in the physical environments where WSN are deployed.

Data collected from the physical world tends to be variable and dynamic. Sensor readings are sensitive and report even slight variations in the environment. Applications may not be concerned with the readings at each individual location, but are interested in the conditions of the whole environment in which the sensors are deployed. For example, micro-environments may exist in a large area where each has its own average or acceptable conditions of temperature. A user wants to request the node id of any sensor node in a hot area, without having to make specific requests to each node. It would be beneficial for the sensor node to determine if it is 'hot', where the semantic value 'hot' is interpreted as a temperature range specific to the area of deployment. Therefore, there is a need for a way to query a macro-environment without having to query each individual mini or micro environment that make up the area.

Leveraging aspects of fuzzy database, specifically fuzzy data representation and fuzzy or flexible queries, as a solution to manage and exploit the inherent variability of sensor data gathered from a large heterogeneous area. I developed a novel approach to querying WSN, in which semantic knowledge about sensor attributes is represented on

the node as fuzzy terms. These terms allow the node to respond to a query based on how closely the sensed value matches or approximates the fuzzy term. A flexible query establishes a target qualification of desired information and is concerned with data close to the target. To represent these queries, fuzzy sets provide a flexible way to define the query concepts linguistically. My research includes extending query processors to provide support for handling the linguistic concepts of fuzzy queries, thereby enabling more efficient data retrieval from WSN.

Furthermore, the performance of WSN data management middleware solutions can be measured by the efficiency with which needed data can be collected by an application, to minimize sensor node battery life consumed by transmissions, and the ease-of-use of the application interface to the WSN middleware. In many respects, fuzzy representation of sensor data allows sensor nodes more broadly interpret if a sensed reading is required for participation in the query result. The sensor node may then decide not to transmit the result data and thus save energy.

As the complexity and scale of WSN applications research increases, so does the need for effective comparative analysis. Deterministic benchmarks for middleware comparisons, as well as stochastic tests modeling the unpredictability of environmental phenomenon are needed. Approaches employed for ad hoc network simulation are not sufficient, since the network is typically modeled solely in terms of network topology. I have developed a simulation architecture which allows test scenarios to evolve independently of simulation models for network protocols and topology. This scenario informed approach to simulation, which incorporates dynamic changes in the underlying environment, provides techniques to evaluate middleware under unpredictable behaviors.

This architecture provides a generalized model that can be used to build a rich variety of models, to configure, simulate and evaluate middleware and applications. Various cellular automata (CA) models representing dynamic physical environments can be developed. Using established CA models, I developed 3 scenarios for evaluation of real world phenomena: earthquakes, fire spread and spread of gases.

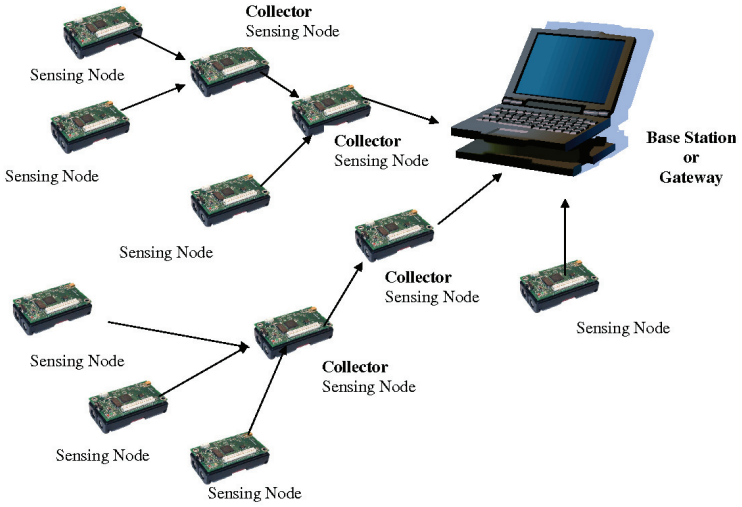
In chapter 2, I review the motivation for my research and outline my research proposal. In chapter 3, I describe my classification scheme and review related WSN data management literature. Chapter 4 describes the implementation of a fuzzy query processor. The dynamic scenario simulation architecture using cellular automata is described in chapter 5. In chapter 6, I describe the simulation scenarios I developed and the evaluation tests used to compare the efficiency of fuzzy and classic queries. Results of the evaluation and discussion thereof are present in Chapter 7. Chapter 8 concludes with remarks.

CHAPTER 2: RESEARCH MOTIVATION

2.1 Background

2.1.1 Wireless Sensor Networks

Sensor nodes and Wireless Sensor Networks enable observation of the physical world in unprecedented levels of granularity. Sensor nodes are compact, autonomous devices with embedded microprocessors, built-in limited energy supply, communication capabilities, and data storage [28, 29, 77, and 15]. They also include sensing hardware, built on Micro-Electro-Mechanical Systems, (MEMS) technology (the integration of mechanical elements, sensors, actuators, and electronics) [24, 17], to collect local environmental conditions. A sensor node may monitor temperature, humidity, barometric pressure, even acceleration if the node is mobile. The sensor node may also have an actuator device attached. For example, due to the limited power supply of sensor nodes which



Base Station Gateway: The base station is a larger computing node (such as a PC) that allows users to formulate queries and injects those queries into the network. There can be multiple base stations.
Collector: A Node can also have the task of collecting data from other nodes. This nodded will be referred to as a Collector.
Sensing Node or Node: This is the sensor node that monitors for sensing information.

FIGURE 1: Sensor Network

must be highly rationed, the node may utilize an actuator to put itself in sleep mode when it is not active.

A sensor network is composed of a large number of sensors nodes which are deployed in an area of interest around a phenomenon of interest. The positions of the sensor nodes are not necessarily pre-determined and there is no fixed networking infrastructure. Through wireless communication, these sensor nodes dynamically form ad hoc networks and co-operate with each other allowing a real-time understanding of environmental changes and patterns [13, 18, 23, 25, 26, and 27].

Wireless sensor networks are used to perform a variety of different application-specific tasks. The nodes may be randomly dispersed in difficult terrain or hidden in wildlife burrows, self-organizing when needed. Applications and even middleware for WSNs may define a specific networking environment for WSNs that best suits their tasks. For explanatory purposes herein, I refer to the simple network model depicted in Figure 1. A Collector Node is a specific sensor node that is elected or chosen to collect data from other nodes. A special type of a Collector Node is a Gateway Node, which is a higher powered device used to collect, process and distribute data. To formulate queries and inject those queries into the network, users rely on the use of a Base Station: a PC node that serves as the network's entry and exit point from the user's perspective. A network can have multiple base stations that are used to inject queries into and deliver results from the network.

Each sensor node operates independently and has small amount of memory for signal processing and task scheduling. Each sensor collects localized readings from a dynamic environment. Surrounding sensor nodes must collaborate to transmit that data

to the gateway node. The resulting WSN network formed by the ad hoc communication among nodes is dynamic. Sensor nodes may be mobile, or may malfunction causing the network to be reconfigured. Neighboring sensor nodes may correlate data or may aggregate or summarize collected data in order to minimize the amount of communication needed to return the information describing the sensed environment. Thus, limited data logging may be available at each sensor node. An underlying constraint to all these functions is the fact that each sensor node must contain its own limited energy supply. Energy conservation is an unavoidable consideration to any WSN solution. It is well known that communicating data over the wireless medium, even at short ranges, consumes far more energy than processing that data on the same sensor node [12]. Minimizing the amount of data sent through the network, yet still providing the gateway node the essential information, can significantly prolong the lifetime of a working sensor node. ***.As such, energy efficiency, as measured in the number of sensor node transmissions, is a critical performance metric for WSN solutions, including WSN middleware solutions.***

2.1.2 Data Management in WSN

A primary purpose of a sensor network is to collect information about a phenomenon of interest. This information may include sensor readings (e.g., temperature, light intensity, acceleration), sensor node location, or information about the energy (i.e., power) management status of the node. Wireless sensor networks are typically spread over a wide environmental area. The technology advances of WSNs bring with them new challenges for information processing. Sensor networks monitoring the physical world are subject to dynamic changes in environmental characteristics over time. As such, queries over sensor networks have both spatial and temporal considerations.

A query can request a bird's eye view of the environment or can zoom in on a particular area of interest. For example, to understand comfort level one may sample temperature data from a particular meeting room or one may sample the average temperature of each floor or the entire building. A snapshot query will monitor network attributes for a limited time period. Queries may also be long running, periodically sampling environmental properties. A snapshot query would sample the temperature of a room during a specific board meeting, while a long running query would request periodic sampling of the room temperature over the course of a month. In addition to measuring ambient environmental characteristics, WSNs can monitor movement or intrusion in the network. Tracking objects moving through the network can be achieved by requesting data about that object's location. These varied roles of sensor network applications have resulted in different solutions to address the management of sensor data. Because WSNs are deployed in specific environments to solve unique problems,

proposed middleware target solutions singular to the desired application. This requires efficient integration of all components of the network. Managing data in WSN is a collaboration of localized storage, routing, in-network processing, and middleware. Data can be stored and retrieved by name instead of by the node location. Data generated at one node can be stored at another node [77]. The routing techniques necessary to facilitate data management in WSN is not simple end-to-end routing by node address. While addressing nodes by location is sometimes necessary, the identity of the node is less relevant than the data it provides [12, 13, 125, and 132]. Therefore, routing solutions are often integrated with and influenced by the application. Sensor networks limited energy resource demands in-network processing to reduce data as much as possible. Instead of blindly routing data, many applications do some data processing at each hop, for example: aggregating similar data or filtering redundant information [126,127,128,129,130, and 131]. These methods are suited for different types of queries. ***Numerous WSN data management techniques are being proposed in the literature. However, not enough attention has been given to the energy efficiency of WSN data management techniques.***

2.1.3 Leveraging Fuzzy Database Approaches

As data collected from the physical world tends to be inexact, dynamic and, varying, the data acquired by the sensor network is unlikely to be a precise representative sample of the physical environment. This is due to a number of reasons including: micro-environments existing within the physical space; non-uniform placement of sensors in space; or readings from sensor nodes with inexact calibration. Sensor nodes are prone to failure and can generate wrong data due to malfunctioning. Sensor readings from a singular area may vary due to influences of localized interference. Therefore, applications concerned with the macro-environment must be prepared to deal with the occurrence of varied data collected in WSN applications, some imprecisely representing the macro-environment.

Consider sensors recording temperature in a forest. One is placed in heavily shaded area. The other is placed in a rocky area in the sunlight. Their readings, while reflective of the immediate surrounding temperature, may not represent the ambient temperature of each separate area. To determine if higher than expected values indicate the presence of a fire, for example, these readings would have to be interpreted at the base station. A reading of a high temperature from the sensor in the heavily shaded area may be indicative of a spread of a fire or other anomaly. While a reading of a high temperature from the sensors in the rocky location may be the result of the midday sun and reflected by rocks. This may result in sending unnecessary values through the network that do not indicate an anomaly. In this situation, it may also be meaningful to express the anomalous condition using higher-order data concepts in a less precise but descriptive semantic language. For example, a user may want to gather information on

sensors in a region where the temperature is significantly high for that region, without specifying an exact temperature value. A useable query expression would be one that allows the request to return temperatures that are ‘close to’ a dangerous level. Emerging work on the use of imprecise data and queries suggests deeper study is needed to determine the appropriateness of these query approaches for extracting data from a sensor network, given that the real-world data is inherently imprecise.

There are two ways to consider the problem of querying dynamic and imprecise information from sensor networks: uncertainty in the data or ambiguity in the query [75]. Current WSN research has addressed some areas of data uncertainty and suggested methods to allow more flexible queries. Storing sensor readings locally at a base station to manage uncertainty has been proposed by current research. Statistical modeling techniques [58] are used at the base station to interpret and predict sensor readings. Queries are then directed to the model at the base station. This shields the query from faulty sensor readings. A specific probability model based on time-varying multivariate Gaussians is suggested by [4]. Other solutions may collaborate with neighboring sensors or aggregate sensor data from a region to address the problem with misreads in sensor or other imprecise data [22, 26]. Imprecise queries or flexible queries have been addressed in [6]. This approach requires all data be transmitted to a central processor, treating the sensor network as an input data stream device. At the central processing location, the data is processed by an inference engine using ECA (event-condition-action) rules making deductions and drawing conclusions. Their work requires all sensor data be transmitted to the central processor and may not be energy effective in large networks.

In classical databases, research on the management of imprecise queries and imprecise data has included fuzzy databases which represent data in terms of fuzzy logic [10, 78]. Fuzzy database research is an area that can be beneficial in wireless sensor networks, specifically if used to model and reason about sensor data. Fuzzy database provides mechanisms to define imprecise data and describe imprecise queries. A concept in fuzzy databases that may benefit sensor networks is the use of fuzzy terms. A fuzzy term describes a function that defines membership in a set based on some measure of how closely the data meets the query constraints. A simple example, shown in Figure 2, uses fuzzy terms 'cold', 'warm', and 'hot' to represent outside temperature. In Minnesota, where winter temperatures reach degrees way below freezing, even -15°F or -26°C, a 'warm' Spring day can be felt at 40°F or 5°C. In North Carolina, where winter temperatures are less severe, 40°F or 5°C is a cool day and is still 'sweater' weather. A 40°F or 5°C day can be considered as somewhat cool and somewhat warm. By representing data as fuzzy terms, semantic knowledge about sensor attributes may be defined on a node. This provides a layer of abstraction similar to the spoken language when requesting information from a sensor network.

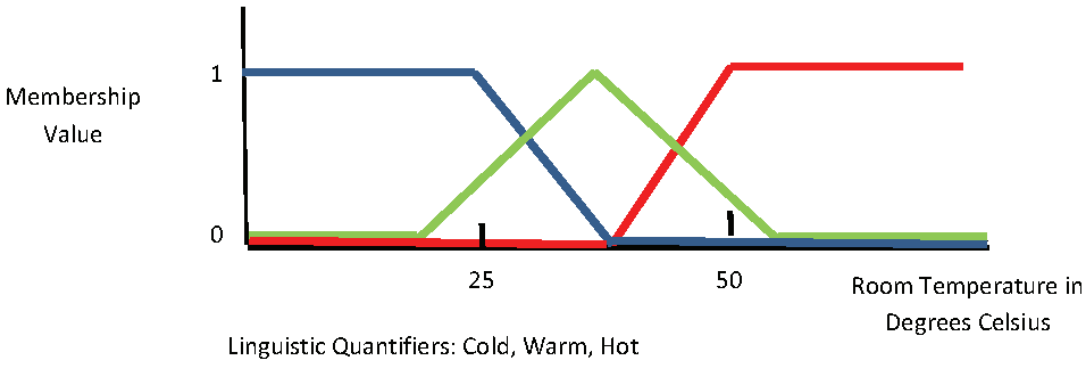


FIGURE 2: Illustration of fuzzy terms

Fuzzy database research is an area that can benefit wireless sensor networks, specifically if used to model and reason about sensor data. In many respects, fuzzy representation of sensor data depicts the physical world more realistically than crisp numbers, taking into account that all phenomena in the physical universe have a degree of inherent uncertainty. Though work on more flexible queries and representation of imprecise data is necessary in the study of wireless sensor networks, most data management approaches do not largely cover data uncertainty.

Fuzzy sets [78] are the central idea of fuzzy query processing. Where as in classical or crisp sets membership is defined by exact inclusion and exclusion criteria, in fuzzy sets membership is a degree of inclusion. A fuzzy membership function measures the how closely the data belongs to a specified set [10]. Formally, fuzzy set A, defined over domain U, known as the 'universe of discourse' is characterized by the membership function,μ:

EQUATION 1: Fuzzy Set Definition

$$\mu A: U \rightarrow [0, 1]$$

which maps elements from the universe of discourse U to a number in the interval [0, 1]. This interval known as the membership grade or membership degree (also called a possibility) indicates a continuous increase from non-membership, where 0 represents data outside the set, to complete membership, where one represents data that is a certain member of the set. That is, the grade of membership indicates the compatibility of the data value with the fuzzy set in terms of values between [0, 1].

Fuzzy membership functions proposed in [91] utilize similarity relations of fuzzy set theory [92]. Permitting data types of finite scalar (label) sets, finite number sets, and finite fuzzy number sets, the user provides values representing a similarity

relation. These similarity values in the relations are standardized in the [0, 1] interval, where 0 corresponds to “totally different” and one to “totally similar”. A similarity threshold can be established in order to retrieve values whose similarity is in some way relative to the threshold.

Using the possibility theory of fuzzy logic, models have been proposed to represent and process imprecision in databases. The possibility theory is based on the idea of linguistic variables and how they are related to fuzzy sets [93, 94]. In this way, one can evaluate the possibility of an attribute value belonging to a set, like a degree of membership in the set [10]. Some important possibilistic models in this group are [10]: (1) Prade-Testemale Model [95, 96], (2) UmanoFukami Model [84], and [83] Zemankova-Kaendel Model [79, 80]. The Prade-Testemale Model allows the integration of incomplete or uncertain data in the possibility model. That is, for an attribute A, its domain D is extended to include a special element denoting the case where A is not applicable to a tuple in question, example NULL or UNKNOWN. The domain of A is now $D' = D \cap \{UNKNOWN\}$ for the tuple. The possibility distribution for A in the tuple is an application that maps D' to the [0, 1] interval. From this formulation all value types, certain and uncertain can be represented. The Umano-FukamiModel also utilizes the possibility distributions to model information knowledge. This model suggests different mapping representation for attribute types that have unknown but applicable values, have undefined or non-applicable values, or have a lack of information whether the value is applicable or non-applicable (NULL). The Zemankova-Kandel Model consists of 3 parts: a value database similar to the previous possibilistic models, an explanatory database, in which fuzzy relations are

stored, a set of translating rules for handling adjectives or modifiers (ex. ‘most’ or ‘very’). In this model the result of the query also presents both possibility and certainty values for each instance returned.

The GEFRED Model [81] is a synthesis of the previous models. Being a possibilistic model, it particularly refers to generalized fuzzy domains of attributes, thus admitting the possibility distribution in the domains. It also includes the case where the underlying domain is not numeric but scalars of any type. It includes unknown, undefined and null value having the same sense as that in Umano-Fukami. That is, the GEFRED model is based on the generalized fuzzy domain and generalized fuzzy relations, but which also includes classic domain and classic relations.

Proposals for fuzzy object oriented database models have also emerged. Fuzzy Object-Oriented Data Model (FOOD) [82, 83] allows the instantiation and representation of vague attributes and relations. The FOOD Model manages certain and uncertain information by using fuzzy set theory and possibility theory.

I hypothesize that importing a fuzzy database model on WSN data management will reduce the number of transmissions required between sensor nodes in order to fulfill requests for information from a WSN application, and will have the added benefit of a familiar, easy to use applications interface.

2.1.4 WSN Simulation Environments

Wireless sensor networks are designed to monitor the natural environment. Middleware and applications that extend the functionality of WSNs are an active area of research. Evaluation of the middleware must be done with consideration of the deployment of the WSN application being addressed. Current simulators for WSN focus on network simulation [45, 112,113, and 114]. While some function is provided to update sensor values, most generally generate random values to be used as sensor readings. Therefore, in order to evaluate WSN middleware, research has deployed the network in an actual physical setting or has developed a lab experimental test bed.

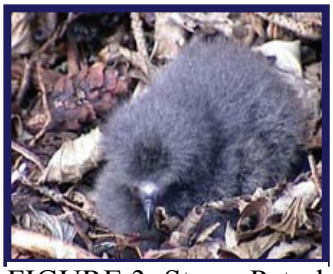


FIGURE 3: Storm Petrel

The Great Duck Island Project [24] and ZebraNet [30, 32] are research projects which deployed WSN in a natural environment. The Great Duck Island (GDI) Project run by the University of Berkeley, CA, dispersed sensor nodes in the nests of the Storm Petrel. The Storm Petrel is a nocturnal seabird, relatively unknown, and have a large nesting habitat is on Great Duck Island, ME. By measuring the temperature in the nests, the researchers were able to monitor the occupancy patterns during the nesting season and environmental changes occuring in the nests and their vicinity during the breeding season.


FIGURE 4: Zebra

ZebraNet, run by Princeton University, is another inter-disciplinary study using WSN to research important long-standing

questions about long-range migration, inter-species interactions, and nocturnal behavior. Data is stored on sensor nodes belted around the animal's neck. Periodically, a research drives by the herd with a gateway node to collect the stored data.

Agilla [35, 36] middleware can be used to detect and form a perimeter around a fire. A fire tracking agent is injected into the network. The fire tracking agent initially wanders around the network looking for fire. Each time it arrives at a node, it checks whether any neighbors are on fire. If so, it clones itself onto nodes within two horizontal and vertical gridhops of the fire. This process is repeated by the clones, eventually forming a perimeter. Fire detection agents continuously check the fire and proactively adopt to breaches in the perimeter.

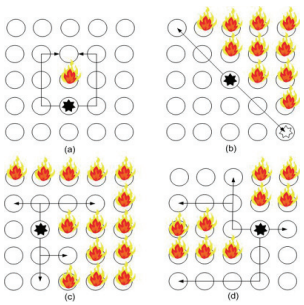


FIGURE 5: Agilla middleware forming a perimeter [124]

Agilla research deployed a test bed used of Mica2 Motes in a 6x9 grid with a single basestation. This lab environment, though effective, is difficult to reproduce. The calibration of the sensors, setting the mote in the layout, the sensitivity of prototype hardware, all contribute to this difficulty. A new middleware would not be able to effectively compare or contrast its contributing enhancements by duplicating this experiment.

FIGURE 6: Agilla experiment test bed [124]

Numerous simulation environments exist for evaluating wireless network protocols [112, 113, and 114]. Typcially, these environments include a network topology model that models the location and interconnection between mobile nodes. The physical environment is implicit in the network topology model. Comprehensive

simulation environments are needed to evaluate WSN solutions, including data management solutions. ***Since environmental monitoring is an important application for WSN, it is important to model the physical environment, and events that occur within that environment, indepent from the WSN that is deployed to monitor the physical environment.***

2.2 Research Proposal: Fuzzy Query Processor for Wireless Sensor Networks

Herein I present my research contributions, which include the following. I provide classification of WSN middleware to illustrate varying approaches to data management for WSN and identify a need to better handle the uncertainty inherent in data collected from physical environments, in order to increase the efficiency of WSN monitoring applications I present a novel approach to querying WSN, in which semantic knowledge about sensor attributes is represented as fuzzy terms. I develop an enhanced simulation environment that supports more flexible and realistic analysis by using cellular automata models to separately model the deployed WSN and the underlying physical environment.

2.2.1 Classification of Data Management Approaches

WSN applications suggest that the utility of a sensor network is primarily in its ability to gather and communicate collected data. There have been surveys proposed to compare and contrast these and other components of WSN middleware that include aspects of managing data [1, 119, 120, and 133] These surveys address the scalability, interface, as well as sub-components of the network used to build the solution. There have been surveys addressing data processing [118]. However, they focus on the application implementation of the middleware. Comparing to existing surveys, I propose a holistic classification scheme of data management approaches, providing a taxonomy that organizes WSN middleware by this classification scheme. By surveying the current literature, I consider four abstractions to classify the data management of the query requests found in the literature: 1) event detections 2) data acquisition, 3) acquisition of semantic information, and 4) tracking movement through mobile code systems. These are distinguished by the types of queries issued by the application and the kinds of information requested.

2.2.2 Fuzzy Query Processing in WSN

In sensor networks, which collect real world data, it is hard to exact a clear cut-off value to gather all the relevant data when querying data from a sensor network [39]. Sensing technology acquires samples of physical phenomena at discrete points in time and space. The data acquired by the sensor network is unlikely to be a precise representative sample of the physical environment, for a number of reasons including non-uniform placement of sensor in space or faulty sensors. Current research has addressed imprecision of the sensor data from the aspect of the uncertainty and variations of the experience of monitoring real world phenomenon [38, 39, 4, 5, 6, and 59]. However, the current sensor network query processing techniques do not provide for queries to request flexible selection criteria over the search space. While queries can delineate data retrieval by exact threshold boundaries, it is not possible to define semantic concepts to express vague queries directly, ex. 'Return node IDs of those sampling temperature significantly higher than average.' Consequently, these vague retrieval requests must be emulated with specific queries. This means that the user is forced to retry a particular query repeatedly with alternative values, until it returns all data that are satisfactory. Multiple queries may be required to drill-down to phenomena of interest, such as regions where the temperature is/was unusually high.

Flexible queries can be more appropriate to better understand the context of the overall information presented by the set of values returned by individual sensor nodes. Taking the meaning of "flexibility" from [71], "a system is flexible in so far as it allows imprecise terms in user queries". By providing a method to define semantic concepts about an attribute, fuzzy sets can enable flexible query processing. In fuzzy query

processing, the selected records are ranked according to their compatibility with the semantics – the intent – of the query. This provides a record of how well the record fits into the complete set of results received. This provides a measure of inclusion; a way to know which records are strongly a representative of a concept and which are only weakly characteristic of the concept. Inspired by Zadeh's Fuzzy Sets Theory [78], research in fuzzy database models has been devoted to extending databases with mechanism to represent and handle information in a flexible way [10]. Fuzzy database literature has addressed data representation, data handling and fuzzy querying [65,72,73,74, 75]. Most research in fuzzy database techniques are typically in reference to relational databases. Though sensor networks are not a relational database, the ideas fuzzy query processing are interesting in respect to how they can provide insight to handling and querying data in sensor networks. While in standard query processing, a query is answered only by data that completely satisfies the selection criteria, in fuzzy query processing data may have a degree of satisfaction with respect to the selection criteria. The user may request to return data that only 'somewhat' satisfies the selection criteria.

Using a database approach to acquisition of data, viewing the network as single data store, my research will develop a fuzzy query processor over WSN. This novel query processor will be designed to define data representation, query interface and in-network processing to handle selection criteria using fuzzy concepts.

2.2.3 Simulation Environment Scenarios Using Cellular Automata for Wireless Sensor Network Analysis

Middleware and applications that extend the functionality of WSNs are an active area of research [22, 26]. This research includes software engineering techniques that simplify the construction of sensor network applications [21]. These generally facilitate high-level design to address the management of the limited resources inherent in WSN hardware, and to mask the complexities of query dissemination and result collection. As the complexity and scale of sensors network applications increase, so does the need for effective comparative analysis [1]. To aid in comparing the sophistication of queries supported by different middleware solutions, it is necessary to develop benchmarks for comparison [26], for example through repeatable simulations of WSN applications and WSN network topologies operating within dynamically changing environments [1]. Simulation approaches used for analysis of ad hoc networks are not sufficient for WSN because an environment model separate from the network topology model is not typically employed.

The need for high-fidelity simulation modeling of both the sensor network and the underlying driving environment scenarios is demonstrated by Park, et al. [98] This work models the plume propagation of air-borne chemical/biological agents, using data generated by the Second-order Closure Integrated Puff (SCIPUFF) [34] modeling tool for atmospheric dispersion applications. SCIPUFF generates grid values indicating the evolution rate of the dispersion at various time steps. Depending on the time step, grid position and geographical distance modeled between the nodes, the appropriate SCIPUFF grid values are injected into each node. This tightly couples the simulated

scenario with the node placement, limiting the possibility of handling dynamic state changes and node movement.

It is advantageous to employ a cooperation of both the network application simulatin and a separate environment simulation to evaluate WSN solutions. Cellular automata provide a natural way of studying the evolution of large physical systems. In spite of its simplicity, CA exhibit a wide range of behavior [16]: stationary uniform states, periodic states in time and space, spatially disordered states and turbulent behavior in continuous evolution [17]. Cellular automata systems have been extensively used as models for complex systems, including research in the field of biology [28], logic circuit design [29], physics [9, 17], ecology [15] and earth science [17, 23, 25]. In [10], Cas model the behavior of natural hazards including a forest-fire model; an earthquake model [23, 25]; and a sand-pile model with landslides [23].

I developed an architecture that can be used to extend existing simulators to include various CA models representing dynamic physical environments. My architecture allows an environment scenario to evolve independent from models for network protocols and topology. I extend Park's work [1, 2], which injects environmental data into each mode as a scenario evolves, to allow sensor nodes to be dispersed, deleted or moved throughout the simulation space and continue receiving updated, relevant environment state data. Because of the flexibility of CA models, scenarios can be developed that include randomness in environment state data.

I implement three cellular automata rules to model different physical systems. The Olami-Feder-Christensen (OFC) model has also been shown to qualitatively exhibit features of seismic aftershocks and stress loading and ruptures cycles. The Frisch,

Hasslacher and Pomeau (FHP) model describes the motion of particles traveling in a discrete space colliding with each other. I use these rules to effectively model scenarios such as spreading of gas, earthquakes, and bridge or building rupture. Mathematical CA model in [32] describes the spread of a fire. I use that model, incorporating weather (wind) and land topology conditions to build a forest fire scenario.

CHAPTER 3: DATA MANAGEMENT APPROACHES IN WSN

WSN applications have been built to monitor the dynamic changes in environmental characteristics over time. In addition, WSNs can monitor movement or intrusion in the network. Tracking objects moving through the network can be achieved by requesting data about that object's location. These middleware implement the specific requirements for the description of the data considered and the form of the queries to collect that data. Comparative studies and analysis of WSN middleware, including data processing, have been presented in [1, 118, 119, 120, and 133]. However, they focus on the application implementation of the middleware. My insterest is in a holistice view of the manner which the middleware defines the use or intent of the data coleccted. I determine this by the description of the information retrieved and by the approaches used for query abstractions.

In section 3.1, I present an classicification scheme for data management approaches with respect to data requested through querying the WSN. I present a review of literature cataloged by this classification in section 3.2.

3.1 Classification of Data Management Approaches

I consider four abstractions to classify the query requests of data processing middleware applications found in the literature: 1) event detections 2) data acquisition, 3) acquisition of semantic information, and 4) tracking movement through mobile code systems. These are distinguished by the types of queries issued by the application and the kinds of information requested, as described next, and illustrated in Figure 7.

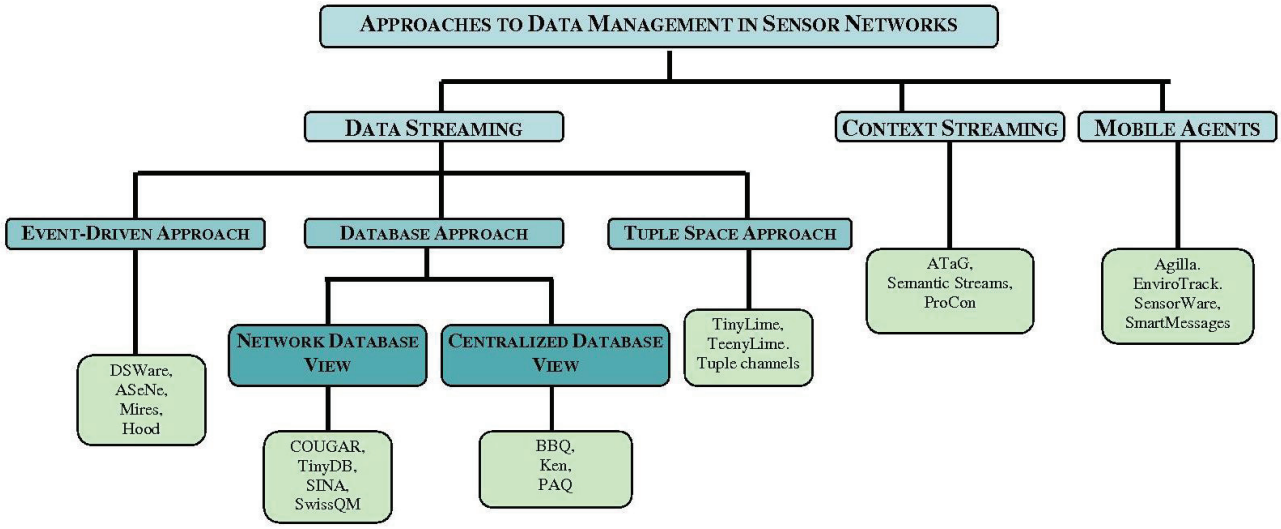


FIGURE 7: Classification of Data Management Approaches

Event Detection: This approach is characterized by an application's subscription to specific data or state changes in the network or by a continuous query over the WSN. Examples of queries used in this approach include "when toxicity of the environment reaches a certain threshold, return the location of the node" or "when a window temperature is between certain values, close the blinds". In this approach, often a publish/subscribe scheme is used to query and extract data. Each node announces a set of attributes that describe the data types it monitors. The application subscribes to a

subset of the attributes or registers an event of interest. The sensor node may remain inactive until an anticipated event is sensed in the environment. At that point, the sensor node directs the data collected by the event to the application subscribers. Events may be detected by a single node or cooperatively by detection in multiple neighboring nodes. A notification of the event is sent towards an interested sensing node or gateway, which initiates appropriate action. An application can request monitoring for multiple events or complex events involving the detection of one event in the absence of another.

Active Database technology [55] is included in this approach. An active database offers a reactive computational model that falls under the event-based category. This model exploits event-condition-action (ECA) Rules (or active rules) that can be mapped to sensor networks that contain both reactive sensing components and actuator components [54]. These rules provide a mechanism to cause the sensor nodes to react to a monitored event rather than to proactively test for an event's occurrence.

Data Acquisition – This classification takes into account that sensor nodes have well-defined attributes (temperature, humidity, etc). Applications monitor or collect sensed data over various intervals of time. The data acquisition approach involves querying the sensor network, typically using an SQL-inspired approach such as "SELECT *." Queries may be issued to request the current data values or to request the sampling of data values over a specified time period. Queries used in this approach may be single-shot "retrieve the current average temperature of the first floor of the building," or periodic "retrieve the average temperature of the first floor every 4 hours." The data may be aggregated or otherwise optimized during collection, but it is the resultant values that are returned to the gateway.

I further classify the data acquisition category into two approaches: database approach and tuple space approach.

Database Approach: In this approach, the sensor network is viewed as a single data store where each sensor node has its own query processor [40]. When networked, sensor nodes may collaborate with or participate in the query process of neighboring sensor nodes. Unlike traditional databases where large amounts of information are stored in one location and updated periodically, sensor networks store small amounts of data, often widely distributed, and the data values can change continuously. Some of the work in this area describes techniques to collect the sensor data at a base station, where the collected data represents the sensor network locally [43]. The sensor network is tasked to collect data only when the local data store requires updates. Other work describes techniques that view the network as a pure sensor database solution [3, 11]. These essentially provide a distributed database solution that is appropriate for the resource-constrained devices in the sensor network. Middleware approaches that treat sensor networks as a database can borrow techniques from both database management and data stream management research.

Tuple Space Approach: In this approach, the network is viewed as a set of shared memory spaces. Data is represented as elementary data structures call tuples which contain the values read by the node's sensors [61]. A tuple space is a virtually shared memory space among collaborating sensor nodes. It provides the appearance of shared memory but does not require a physically shared memory. Data residing on different sensor nodes within the tuple space appear to be stored in one single global memory to the requester. When a sensor node in a tuple space (most often, the gateway

node) is queried, it represents the collected data of every other node in the tuple space.

Acquisition of Semantic Information – In this classification, knowledge gathered from sensor data is used to obtain contextual information about the environment or about the state of the node itself. Middleware or applications will then infer semantic information by combining data from different sensors, correlating elements such as timestamps, acceleration, or proximity. It is the semantic conclusion that is queried and returned to the gateway [43]. This approach allows users to pose queries over semantic interpretations of sensor data, such as "I want the speed of the cars driving in front of the parking lot elevator". Sensor nodes must generate semantic inferences about the environment. For example, if an object is a certain size, it is a vehicle and the speed is returned.

Mobile Code Systems – This classification includes techniques for collecting and processing distributed data within a sensor network by dynamically relocating the collection and processing code throughout a sensor network. Mobile code technologies include the design of both programming languages and their corresponding runtime support. Here, I consider the Mobile Agent paradigm [44]. The mobile agent is a computational unit that contains state and data information. Instead of moving data from the node to the requester, mobile agents move through the sensor network to both collect and process local data. The program moves itself through the network, retaining its own state information as well as data. Sending code as well as data through the network implies a paradigm for data management which is more general than sending data alone.

3.1 Survey of Current Literature

The following is a comparative study of several middleware solutions for sensor networks, as categorized by the data management classifications in Figure 7.

EVENT DETECTION SOLUTIONS

Data Service Middleware (DSWare): DSWare [56] is an event-driven data service abstraction that provides an easy-to-use SQL-like interface to the application. DSWare provides services to detect both atomic and compound events. Compound events are those that describe two or more atomic sub-events. Compound events implement the notion of confidence to address the possibility that not all sub-events have been detected. Data is processed and aggregated through the network to limit transmissions, sending only the result to the base station. The group management component controls the cooperation among sensor nodes to accomplish robust and spatial objectives.

Mires: Mires [53] is a message-oriented middleware for sensor networks that allows applications to communicate through a publish/subscribe design paradigm. In the Mires architecture, sensor nodes are responsible for advertising the sensor data they provide. The network application subscribes to data selected from the advertised service. Sensor nodes transmit data according to these subscriptions. Mires focus on networking and architecture issues, rather than on the data model or subscription semantics. It is probable that the subscription language is similar to traditional distributed applications.

Active Sensor Networks (AseNe): AseNe [54] provides an active database approach to sensor networks. Active databases define an appropriate computational model for event-condition-action (ECA) rules, also referred to as active rules. Active rules can be sent to the sensor nodes using SQL-like statements. AseNe is built on event channels which are

viewed as data object primitives. The event channels are responsible for maintaining lists of subscribers and for sending notifications of published events to the subscribers. When an event occurs with a condition requiring a subsequent action as defined in an active rule, event channels distribute the events following the publish/subscribe paradigm.

The data model for this approach requires that it distinguishes between different types of events being monitored. This model will provide a mechanism identifying a set of sensor nodes involved in monitoring for the same event and for the coordination of their operations. Each node, then, encapsulates a description of the model(s) it manages.

DATA ACQUISITION SOLUTIONS

Data Acquisition Solutions: Database Approach

The database approach to data acquisition solutions are further classified by two differing views for modeling the data. In the centralized database view, the data is collected and managed at the base station. The sensor data previously collected and stored at the base station is queried first. Network sensor data is again queried only if needed to complete or update the centralized database at the base state. In the network database view, the entire network is considered as the data store for each query. Queries are distributed throughout the network for each request. Each node decides whether it will participate in the query by doing any of the following: returning data, forwarding the request or processing returning data.

Data Acquisition Solutions: Database Approach – Centralized View

Barbie-Q (BBQ): A Tiny-Model Query System (BBQ) [62] employs a central data model on a base station to represent the state of the network. This model is built from previously stored sensor readings. The centralized database is used to examine stored

sensor readings to form the logical estimates. Requests are made to the sensor network only when the centrally stored data is insufficient to answer the query. BBQ uses probabilistic modeling techniques to optimize data acquisition for sensor networks. Using a correlation-aware probabilistic model, SQL queries are answered utilizing the correlation of values from nearby sensors.

Ken: Ken [58] also employs a central data collection on a base station to represent the state of the network. This differs from BBQ in that both the base station and sensor nodes maintain a synchronized and dynamic probabilistic model of how data evolves. The base station uses the predicted data models as the true data. The sensor nodes are queried to ensure the predicted data values satisfy the required approximation guarantees.

Probabilistic Adaptable Query System (PAQ): PAQ [63] represents the state of the network with a central data collection on a base station. Using a probabilistic approach, statistical techniques are used to model the recent history of a sensor reading to predict the most likely future values. This model is used both globally at the base station to predict the readings of individual sensors and locally at each sensor to detect outlier readings.

Data Acquisition Solutions: Database Approach – Network View

COUGAR: One of the early models of the database approach for sensor networks, COUGAR [3, 8] defined a platform for query processing over ad hoc sensor networks. In COUGAR, the network itself is modeled as a virtual relational database. Data is represented as tuples, which include a sensor data sample including the node ID, sensor name, sensor value, and timestamp. The query processor, (or "query proxy"), runs on each sensor node to interpret and execute queries. Nodes are clustered at deployment by

specifying their physical boundaries. All nodes within a boundary are considered to be within a certain cluster. A node within the cluster is elected 'leader.' This node may perform some optimization on the query request and may aggregate data before it is sent to allow optimization of query processing within the cluster and lessen the amount of data transmitted.

TinyDB: Like COUGAR, TinyDB [9, 11] models the network as a virtual relational database. Data is represented as tuples, which include a sensor data sample including the node ID, sensor name, sensor value, and timestamp. TinyDB implements an acquisitional query processing system to allow sensor nodes to make runtime decisions about the value of an individual data item. Sensor nodes are networked through 'semantic routing trees' providing a tree-like structure built during query dissemination which connects all nodes with data specified by the query. Data is processed and aggregated at the connective tree nodes as it is returned through the network, limiting the transmission size of the result sent to the requester.

Sensor Information Networking Architecture (SINA): SINA [57] provides a data view that models a virtual spreadsheet database and provides mechanisms to create associations among sensor nodes. Hierarchical clusters of sensor nodes are formed based on power levels and proximity. It makes use of the property that near-by nodes have similar data and optimized data collection, by returning only the data that the other nodes don't have.

Spatial Queries (SPIX): SPIX [37, 64] is a distributed spatial index over WSN that is used in processing spatial queries. Spatial queries are subsets of queries in which the sensor network is queried by location. Spatial queries are used to answer questions such

as *find the average temperature in an area within one mile of the point of interest.* In this solution, participating sensors are located through a distributed spatial index that is built on top of a sensor network. It essentially forms a routing tree that determines a communication path among the nodes where each sensor node maintains information about a bounded area which covers itself and other nodes below it (other nodes also in the area appearing later in the routing tree). When a node receives a query, it determines if it should forward the query request to its children. The query arrives at the location of interest and is then answered by the nodes in that area.

Data Acquisition Solutions: Tuple Space Approach

TinyLIME: TinyLIME [61] is a data sharing middleware which extends the LIME [60] middleware by providing features and components specialized for sensor networks. Data is represented by data structures on each node similar to the Lime construct of a tuple space. Network data is managed as a federated tuple space where a base station shares and integrates tuple space, provided by its connected nodes within the network. In order for sensor data to be available to the network, the node must be directly connected to a base station.

TeenyLIME – TeenyLIME [59, 60] also is based on the LIME middleware. TeenyLIME differs from TinyLIME in that TeenyLime applications can operate without relying on an external base station. Tuple spaces are distributed among the sensor nodes, transiently sharing the tuple spaces contents with one-hop neighboring nodes. Because each node has a different set of neighbors, the tuple space view varies for each node. Sensed data is stored locally as tuples and is made available for queries from neighboring devices.

ACQUISITION OF SEMANTIC INFORMATION SOLUTIONS

Semantic Streams: The framework suggested by semantic streams [43] allows non-technical users to pose queries through semantic interpretation of sensor data. The programming model contains two fundamental elements: event streams (representing real world events with properties such as time, location, movement, and value) and inference units (processes that operate on event streams). Inference units infer semantic information from incoming events, combining events to form new event streams or adding information to existing event streams. Queries are issued on semantic values using a logic programming language based on Prolog.

Proactive Context-Aware: (ProCON) [41] is a context detection mechanism where data is defined through a structure called a context descriptor. With information about events, conditions, and actions given by the user, context descriptors are generated and delivered to sensors. Using context descriptors, context decisions are made within the sensor network to more efficiently deliver information to the requester. Sensor networks can deliver context-level information, not raw sensor data, to the proper actuators. Cooperating nodes are connected through a network overlay, through which event notifications are made.

MOBILE CODE SYSTEMS SOLUTIONS

EnviroTrack: EnviroTrack [43] is designed for applications that need to monitor environmental events or track objects. In EnviroTrack, environmental events are defined as addressable, user-defined entities. Sensors which detect similar entities form groups around them. A context label is associated with each group to represent the tracked entity. These labels then follow the tracked entity through the network. Tracking objects are

attached to context labels to perform context-specific computation. As the entity moves, sensor nodes join and depart the sensor group formed by a context. Membership changes occur automatically. The data attributes of the entity (e.g., location) are gathered and averaged with other sensors in the group.

Smart Messages: Smart Messages [33] are execution units which migrate through the network to execute on nodes of interest to collect data. They route themselves at each node in a path toward another node of interest. At each node, the payload is analyzed to evaluate the path and process the data. Smart messages can dynamically assemble new, possibly smaller Smart Messages, incorporating only necessary information to reduce transmission size. The Smart Messages project proposes a cooperative distributed computation model for sensor networks based on the migration of Smart Messages.

Agilla: Agilla [35, 36] applications consist of mobile agents that will move and clone themselves. Agilla maintains a neighbor list for each device that agents use to discover neighboring devices to which they may migrate. The agents coordinate through tuple spaces. One agent may insert a sensor reading into a tuple space and another can retrieve it without reciprocal awareness or co-location. Agilla architecture allows new agents to be injected into the network and allows old agents to die. Multiple agents can coexist, as can multiple applications.

Each classification presents different approaches to data management in sensor networks with respect to the information queried by the applications solutions. These queries include detection of changing events, queries to retrieve sensor data from the entire query or from a distinct spatial region within the network, queries that inferred semantic knowledge from sensor data collected and queries that were used to track

phenomena through the network. In each of the solutions provided, the context of the information queried is required to be expressed using exact parameters. That is, the query will make arbitrary determinations about what does and does not fit the criteria for selection based on crisp threshold values. However, in sensor networks, which collect real world data, it is hard to exact a clear cut-off value to gather all the relevant data when querying data from a sensor network. Therefore, each classification of data management approaches could benefit from fuzzy queries.

CHAPTER 4: FUZZY QUERY PROCESSING

Leveraging aspects of fuzzy database, specifically fuzzy data representation and fuzzy or flexible queries, as a solution to manage and exploit the inherent uncertainty of sensor data in WSN, I developed a novel fuzzy approach to querying WSN wherein semantic knowledge about sensor attributes is represented on the node. This fuzzy approach must demonstrate a level of accuracy in the set of data returned and must consider the limited resources of storage and energy on the nodes comprising the WSN. The storage capacity of sensor nodes will increase as research continues to address this. However, storage space requirements will continue to be a treasured resource on small sensor nodes. Therefore, I considered storage requirements as a design consideration. My objective in employing a fuzzy query approach is to enable retrieval of intended information with a lesser amount of message passing among wireless sensor nodes as compared to that required for a more traditional query approach. The limited energy resource available on each sensor node must be conserved. Most energy is lost in the transmission of messages as data is returned to the gateway node. Therefore, I considered message passing as a key criteria.

In order to implement fuzzy query processing for WSN, I needed to apply fuzzy terms to the domain. I define these terms in Section 4.1. Fuzzy membership functions are required be stored on each node and used by the fuzzy query. These are discussed in Section 4.2. The implementation changes are described in Section 4.3.

4.1 Fuzzy Terms

As I extend fuzzy query processing to WSN, I will use the following concepts common in the fuzzy database area: Fuzzy Term, Fuzzy Attribute, and Fuzzy Value, Fuzzy Qualifier (modifier or hedges), and Fuzzy Membership Function:

Fuzzy Term: A fuzzy term is an imprecise value over a sensor attribute. It ties a specific semantic concept to an attribute. A fuzzy term can be considered as a linguistic (semantic) label defined over an attribute representing a subset of possible attribute values. Fuzzy terms are defined by a tuple (linguistic name, fuzzy membership function). For example, fuzzy terms may be: (cold, fuzzy membership function with temperatures between 0 and 5 as certainly cold and temperatures between 5 and 15 as somewhat cold) and (hot, fuzzy membership function where temperatures from 25 to 35 are somewhat hot and temperatures from 50 to 100 are certainly hot).

Fuzzy Attribute: A sensor attribute can be considered a fuzzy attribute when it is associated with a set of fuzzy terms. A fuzzy attribute will contain the attribute name and a set of fuzzy terms used to describe the particular fuzzy concepts of the attribute. For example a fuzzy attribute may be (temp {cold, hot}).

Fuzzy value: A fuzzy membership function can be viewed as a curve that defines how values are mapped to a membership value (or fuzzy value) between 0 and 1. This degree of membership describes how a value (for the purposes of this work, the value is a sensor value) ranks in the fuzzy membership set. A fuzzy value is a value between [0, 1] that indicates the compatibility of the sensor value with the parameters of the fuzzy term. Fuzzy value 0 means that the sensor value is not a member of the fuzzy set; fuzzy value 1 means that the sensor value is fully a member of the fuzzy set. The fuzzy values between

0 and 1 characterize sensor values, which belong to the fuzzy set only partially.

Fuzzy Membership Function: The fuzzy membership function defines the requirements or characteristics of members within a fuzzy set. It represents a valuation of the degree of truth of an element's membership in the fuzzy set. A simple membership function is a shifted unit step function. Sometimes, this is called a 'heaviside' step fuzzy membership function [76]. This membership function defines a sharp boundary between values that are in the set (have a fuzzy value of 1) and those not members of the set (having a fuzzy value of 0). A heaviside_right fuzzy membership function is graphed as Figure 8 where the value is shown as 'a'. Any value equal to or greater than the value represented by 'a' has a fuzzy value of 1. Any value less than the value represented by 'a' has a fuzzy value of 0. The fuzzy membership function is shown in Equation 2.

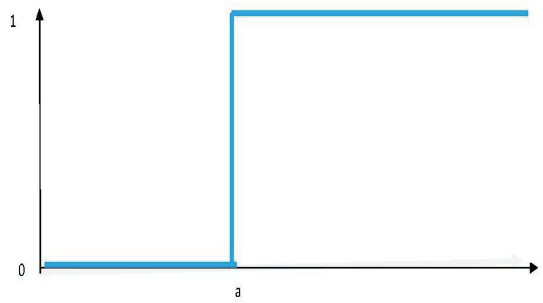


FIGURE 8: Heaviside_right fuzzy membership function

EQUATION 2: Heaviside_right fuzzy membership function

$$fuzzy(x) = \begin{cases} 0 & if\ x < a \\ 1 & if\ a \leq x \end{cases}$$

Inversely, a heaviside_left function is graphed in Figure 9. Its membership function is shown in Equation 3. An example of using this membership function is a requirement of different solvents being at boiling temperature before being mixed. Each

solvent has its own boiling point and will have its own membership function, specifying a value for ‘a’ (boiling point temperature) that is appropriate for that solvent. A fuzzy query could be made to all solvents requesting notification when the boiling point is reached.

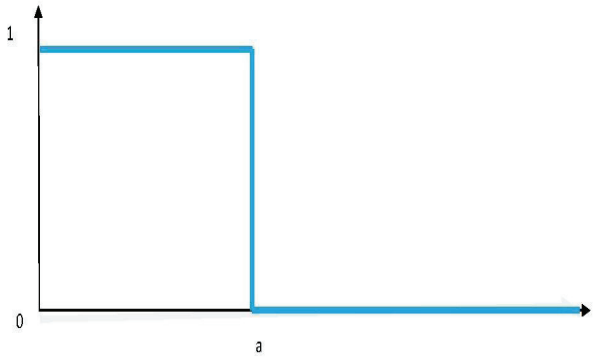


FIGURE 9: Heaviside_left fuzzy membership function

EQUATION 3: Heaviside_left fuzzy membership function

$$fuzzy(x) = \begin{Bmatrix} 0 & if\ a > x \\ 1 & if\ x \leq a \end{Bmatrix}$$

Fuzzy Qualifiers: In fuzzy terms, the selected results are ranked according to their compatibility with the semantic or intention of the query. It is possible to add fuzzy qualifiers to fuzzy terms. Fuzzy qualifiers are phrases such as “very,” “slightly,” “significantly” or “weakly.” Adding a fuzzy qualifier to a fuzzy term (e.g. ‘very cold’), changes the selection criteria to incorporate the intent of the qualifier. It is basically another fuzzy term.

To explain these terms in reference to each other, I refer to the outdoor temperature example from Chapter 2. In Figure 10, the fuzzy terms are 'cold,' 'warm,' and 'hot.' These are the linguistic labels used to describe the temperature values. The fuzzy attribute is the temperature reading. Temperature values are mapped onto the fuzzy terms ('cold,' 'warm,' and 'hot') using a fuzzy membership function. Each temperature listed on the x-axis is mapped to one or more values between 0 and 1. This mapping is shown by the lines of each fuzzy term as an algebraic function used to map the temperature reading to the fuzzy values. In this example, a temperature below the fuzzy membership function for 'cold' is when the temperature is 32°F or 0°C, it is definitely cold. If the temperature higher than 32°F or 0°C but lower than 50°F or 10°C, it becomes less cold, but is still somewhat cold. If the temperature is greater than 50°F or 10°C, it is

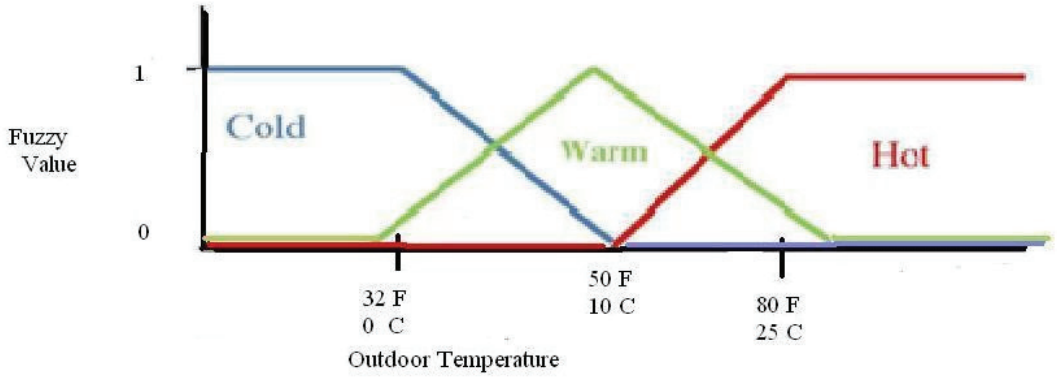


FIGURE 10: Fuzzy Terms Example

EQUATION 4: Fuzzy membership function example

$$fuzzy\ value(temperature): \begin{cases} 1 & if\ reading \leq 32°F\ or\ 0°C \\ \frac{a+b}{2} & if\ 32°F\ or\ 0°C \leq reading\ is \leq 50°F\ or\ 10°C \\ 0 & if\ reading\ is \geq 50°F\ or\ 10°C \end{cases}$$

definitely not cold. The fuzzy value is then determined by the line between those values. The fuzzy membership function for cold is defined by equation 4:

There are similar fuzzy membership functions for both ‘warm’ and ‘hot.’ Therefore a single temperature reading can have a different fuzzy value depending on the fuzzy membership function used.

Fuzzy qualifiers are not identified in Figure 10, but are suggested by how close the temperature value is mapped to the extreme fuzzy values of 0 or 1. There is no definitive definition describing ‘very’ or ‘somewhat’ values with regard to a fuzzy membership function. Instead the definitions are left for the application to define.

4.2 Fuzzy Membership Functions

To investigate a general approach to using fuzzy queries on WSN, I consider standard fuzzy membership functions that are concise and economical with respect to the limited resources of the sensor motes. The membership functions I selected address two common query goals within WSN applications: monitoring a value as it approaches a threshold; and monitoring a value within an interval.

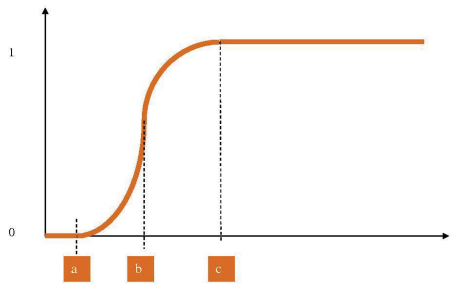


FIGURE 12: S membership function

Applications for WSN include monitoring the environmental phenomena that approaches a threshold. Consider sensors monitoring the stress along a levee. The levee strength is determined by domain experts and may vary in areas where the levee is older or along a seam between sections.

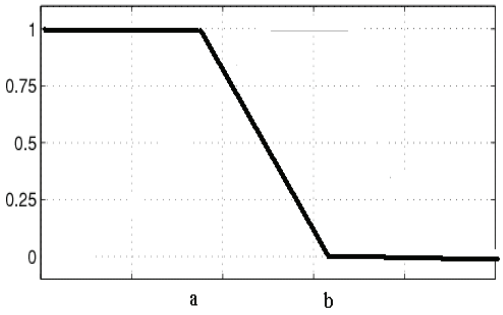


FIGURE 12: L-membership function [52]

These strength parameters can be stored in each L-membership function sensor, representing the stress threshold of the area it monitors. The user can then query the network for any sensor measuring stress that is approaching the local limit.

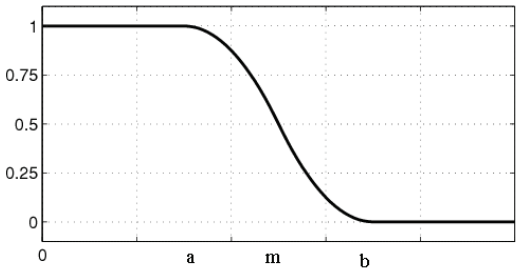


FIGURE 11: Z-membership function [52]

The S-shaped membership function is defined by a lower limit, an upper limit and by a point of inflection between the two. This provides a non-linear progression of membership grade. The growth of the membership grade can vary by the point of inflection. The Z-function and the L-function are also

named after the shape the curve assumes. The Z-function is a reverse of the S-function. The L-function is a linear variation of the S-Function

Queries are issued to retrieve information from sensors about how close their readings are to the pre-determined limit value. In the example of stress a query issued may be:

SELECT node_id, stress, FUZZYTERM(rupture) FROM sensors
WHERE FUZZYTERM(rupture) > 0.8

This query returns the node identifiers, their stress values and how close each value is to the pre- determined rupture value based on the selection criteria where the stress value has a membership degree of .8 or higher in the fuzzy set defined by the term 'rupture.'

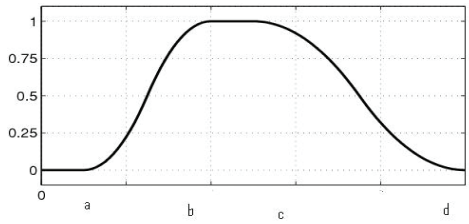


FIGURE 13: PI membership function [52]

Other applications monitor the network to determine if a set of sensors are within a certain range set by domain experts. Information desired from the sensor network where the measured value is in a defined range can be linguistically defined, ex. 'hot,' 'loud,' 'low light,' or 'bright light.' However, associating a precise set of values to define the interval may not accurately achieve the intended objective. For example associating a high temperature range with 100˚C and higher [100, …), suggests that 99˚C is not 'hot.' In this case querying a network in which all sensors

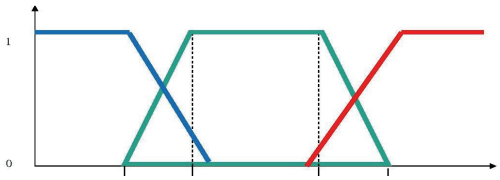


FIGURE 15: Trapezoidal membership function

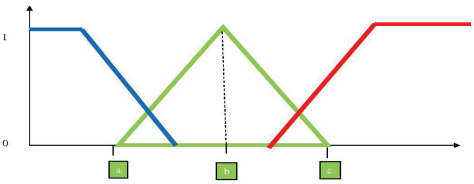


FIGURE 14: Triangle membership function

registered between 89˚C and 95˚C would not result in the intended objective, which is notifying the user that the region was indeed hot. Expanding the scope of the query to 93˚C will result in more sensor nodes returning an indication of a hot region, but it still does not capture the intent of the query. Defining these intervals using fuzzy terms provides a more flexible query request and returns results more closely matching the intention of the query.

The TRIANGLE, TRAPEZOIDAL, and PI membership functions define interval ranges as well as functions for values "close to" the range fuzzy set membership functions. Required parameters are: TRIANGLE [a,b,c] , shown in Figure 15, where a is the lower limit of the range, b is the single value completely satisfying the set, and c is the upper limit; TRAPEZOID [a,b,c,d], where a is the lower limit, d is the upper limit, and b and c are the lower and upper limits of its nucleus, shown in Figure 16; and PI [a,b,c,d] similar to the trapezoidal function, but with non-linear progression, shown in Figure 14.

Queries can be issued to retrieve information from sensors to register a temperature that has been defined as hot within a membership degree:

SELECT node_id, temp, FUZZYTERM(hot) FROM sensors
WHERE FUZZYTERM(hot)> veryHot

This query returns the node identifiers, their temperature values and how close each value is to the defined range 'hot' (FUZZYVALUE) based on the selection criteria, veryHot, is defined as temperature value which has a membership degree of .7 or higher in the fuzzy set defined by the term 'hot.'

I now give functional descriptions of each membership function covered. Storage requirements for each function is listed

4.2.1 Fuzzy Membership Function: Triangular Function

The triangular membership function of a vector that defines a triangle and is depends on its lower limit *a* and its upper limit *c,* and the parameter defining the peak of the curve, b.

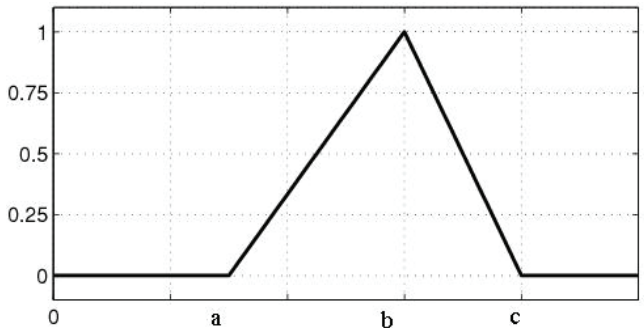


FIGURE 16: Fuzzy membership function: triangular function [52

The membership function implemented to determine the value representing the membership degree of the element x is as follows:

EQUATION 5: Fuzzy membership function – triangular function

$$f(x,a,b,m) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x < b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0 & x > c \end{cases}$$

Storage requirements for this membership function are integer variables for the parameters: a, b, c.

4.3.2 Fuzzy Membership Function: Trapezoidal Function

This shape of this fuzzy function resembles a trapezoid. TRAPEZOID [a,b,c,d] where a is the lower limit, d is the upper limit, and b and c are the lower and upper limits of its nucleus.

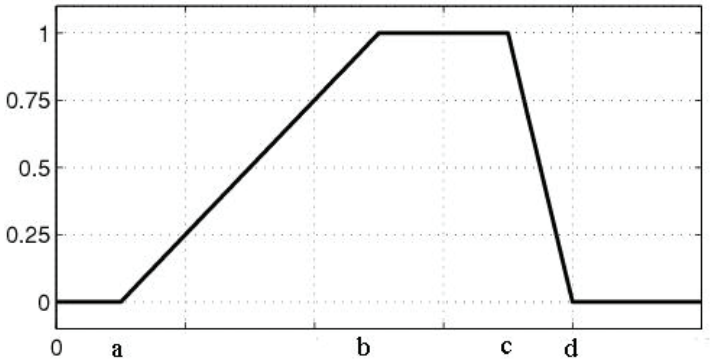


FIGURE 17: Fuzzy membership function – trapezoidal function [52]

The membership function used to determine the value representing the membership degree of the element x is as follows:

EQUATION 6: Fuzzy membership function for Trapezoidal Function

$$f(x,a,b,c,d) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x < b \\ 1, & b \le x < c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0 & x > c \end{cases}$$

Storage requirements for this membership function are integer variables for the parameters: a, b, c, d.

4.4.3 Fuzzy Membership Function: Pi- Function

This membership function is named because of its Π (pi) shape. The PI membership function is defined by four (4) parameters. The parameters a and d locate the "feet" of the curve, while b and c locate its "shoulders."

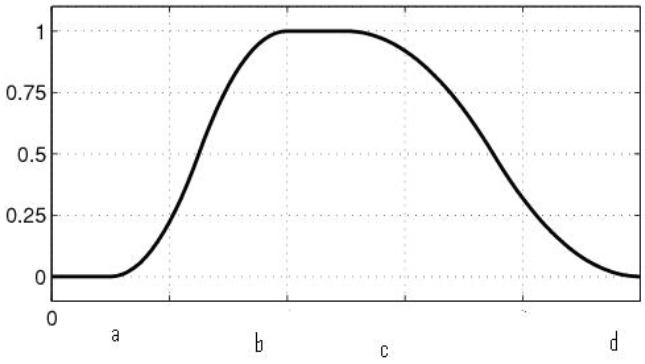


FIGURE 18: Fuzzy Membership Function: PI-Function [52]

The membership function used to determine the value representing the membership degree of the element x is as follows:

EQUATION 7: Fuzzy membership function for PI function

$$f(x,a,b,c,d) = \begin{cases} 0, & x<a \\ 2\left(\frac{x-a}{b-a}\right)^2, & a\le x<\frac{a+b}{2} \\ 1-2\left(\frac{x-b}{b-a}\right)^2, & \frac{a+b}{2}\le x<b \\ 1-2\left(\frac{x-c}{d-c}\right)^2, & b\le x<\frac{c+d}{2} \\ 2\left(\frac{x-d}{d-c}\right)^2, & \frac{c+d}{2}\le x\le d \\ 0, & x>d \end{cases}$$

Storage requirements for this membership function are integer variables for the parameters: a, b, c, d.

4.2.4 Fuzzy Membership Function: S-Function

The S-membership function is defined by its lower limit a, its upper limit b, and the value of m or the point of inflection so that a<m<b. A typical value for the point of inflection is $m = \frac{a+b}{2}$. Growth is slower when the distance from a to b increase..

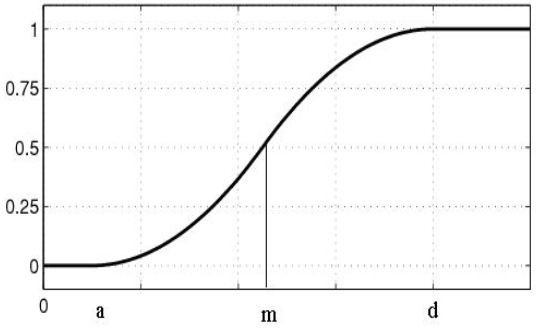


FIGURE 19: Fuzzy Membership Function: S-Function [52]

The membership function used to determine the value representing the membership degree of the element x is as follows:

EQUATION 8: Fuzzy membership function for S-curve

$$f(x,a,b,m) = \begin{cases} 0, & x < a \\ 2\left(\frac{x-a}{b-a}\right)^2, & a \le x < m \\ 1-2\left(\frac{x-b}{b-a}\right)^2, & m \le x \le b \\ 1, & x > b \end{cases}$$

Storage requirements for this membership function are integer variables for the parameters a,m, and b.

4.2.5 Fuzzy Membership Function: L- Function

This membership function is named after its L shape and is depends on its upper limit *a* and its upper limit, b.

The membership function used to determine the value representing the membership degree of the element x is as follows:

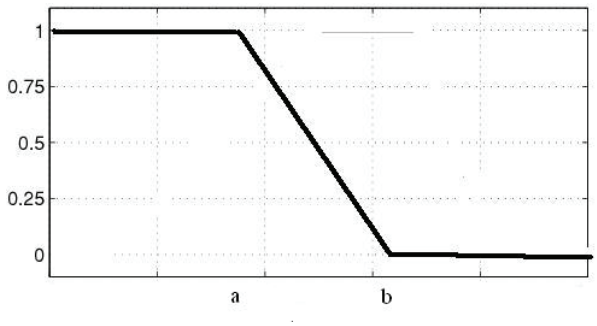


FIGURE 20: Fuzzy Membership Function: L-Function [52]

EQUATION 9: Fuzzy membership function for L- function

$$f(x,a,b) = \begin{cases} 1, & x < a \\ \dfrac{b-x}{b-a}, & a \le x \le b \\ 0 & x > b \end{cases}$$

Storage requirements for this membership function are integer variables for the parameters a, b.

4.2.6 Fuzzy Membership Function: Z-Function

This fuzzy function has a Z-shape. Similar to its mirror, the S-Function, it is defined by its lower limit b, its upper limit a, and the value of m or the point of inflections so that a<m<b. A typical value for the point of inflection is $m = \frac{a+b}{2}$. Growth is slower when the distance from a to b increase. .

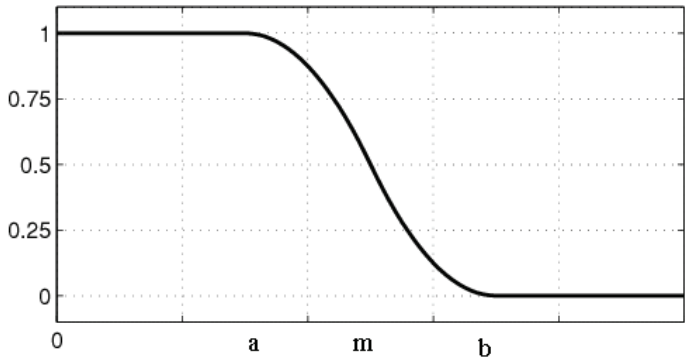


FIGURE 21: Fuzzy Membership Function: Z-Function [52

The membership function used to determine the value representing the membership degree of the element x is as follows:

EQUATION 10: Fuzzy membership function for Z-Function

$$f(x,a,b,m) = \begin{cases} 1, & x < a \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2, & a \le x < m \\ 2\left(\frac{x-b}{b-a}\right)^2, & m \le x \le b \\ 0, & x > b \end{cases}$$

Storage requirements for this membership function are integer variables for the parameters a, m, b

4.3 Implementation of Fuzzy Query Processing

My implementation extends the SwissQM [66, 67] query processing engine which runs upon the TinyOS [47] WSN operating system on each node. TinyOS is a component based operating system targeting sensor nodes and wireless sensor networks. Designed by UC Berkeley, it is a small, open source, event-based system where each system module is designed to continually respond to incoming events. An imbedded operating system, it is written in nesC [48], a C language extension that reflects the encapsulation and execution model of TinyOS design. TinyOS is a well established operating system for sensor nodes. However, it provides a low-level interface for application development.

SwissQM [66, 67] (Scalable Wireless Sensor Network Query Machine) is a virtual machine that runs on TinyOS and is designed to provide application development at a cleaner and higher level interface than TinyOS. SwissQM provides a declarative programming model and imposes no data model. With its integration with TinyOS, SwissQM has the added benefit that the same implementation can be executed in the TOSSIM [45] simulator or deployed on physical sensors. SwissQM has two basic components: 1) the WSN interface manager running on the gateway node; 2) the query manager running on each node in the network. Queries are written as specialized programs that are sent over the network and run as part of the query manager. Because of its performance advantage, SwissQM executes a program in two phases, first compiling the source code into bytecode, and then passing the bytecode to the virtual machine. Using this design, the interface manager parses and assembles the user query program into a bytecode program that is interpreted by the query manager on each node.

The gateway node disseminates the query bytecode program through the network and collects the results. The query manager on each node is written in nesC and is integrated with the TinyOS operating system. Interpreting the query bytecode program, the query manager collects data from the sensors, processes the query constraints and returns the results.

To support fuzzy query processing, I extended SwissQM by adding 3 features: persistent variables, an update function to alter program state, and a new bytecode instruction 'get_fuzzyvalue.' These are required to support the fuzzy membership functions in the queries and to provide means to store, update and retrieve data necessary for the implementation of the fuzzy membership functions. These are described next.

4.3.1 Shared Variables for Query Programs

Memory in sensor nodes typically consists of persistent memory (Flash) and volatile data memory (SRAM). The memory use of SwissQM is differentiated between the SwissQM query manager application and the query programs that are executed by query manager. The SwissQM query manager application resides in the persistent memory. The state information of the SwissQM application is stored in the volatile data memory in volatile memory. The query programs or bytecode programs and their data structures are loaded into a heap space that SwissQM allocates within the volatile memory. When a program is stopped, the heap memory is deallocated. Therefore, there is no way for data (parameters for the fuzzy membership function) to persist across separate query programs

In order for SwissQM to manage the limited space on the mote, it does not permit dynamic allocation of memory within the query manager or query program. All memory is allocated at the time the program is loaded. To facilitate persistent data, I created eight (8) new SwissQM instructions. The instruction modules, existing within the SwissQM application in the persistent memory, store the data as local variables. Query programs can then store and access the variables. In this way, these data are persistent across query programs. The instructions are:

iload_gl1 : loads a global variable onto the operand stack

istore_gl1 : stores the top element on the operand stack as a global variable

iload_gl2 : loads a global variable onto the operand stack

istore_gl2 : stores the top element on the operand stack as a global variable

iload_gl3 : loads a global variable onto the operand stack

istore_gl3 : stores the top element on the operand stack as a global variable

iload_gl4 : loads a global variable onto the operand stack

istore_gl4 : stores the top element on the operand stack as a global variable

4.3.2 Communicating State Changes To Network Nodes

In SwissQM, the query or bytecode program structure is divided into 3 sections: initialization, delivery, and reception. These are combined to process the different phases of data acquisition. The initialization section is executed once after the program is loaded into memory and can be used to initialize the local data structures. The delivery section is executed periodically in response to a time. The timer is set by an execution interval, called sampling period. The intended use of the delivery section is to sample the sensors and send the information towards the gateway node. This data is sent as a result message and contains an epoch number that associates the data with the current sampling period. The reception section is executed when a node receives a result message from any of its children. It is used to intercept the message in order to aggregate data of a particular epoch. If no reception section is present, a node simply forwards the data to its parent. Each program is identified by a program number. Messages generated by the program contain this number in order to allow nodes to correlate messages with programs. This correlation is managed by the SwissQM query manager application.

In order to affect changes to either the local data variables within an active query program or to change the values of the global variables, I implemented a new section to the query or bytecode program, *update section*, and a new message type, *update message*. The update message is broadcast through the network by the gateway node and is given the program number of the associated query program listening for this update message. When present, the update section is executed when a node intercepts an update message.

An example of a SwissQM query progam demonstrating these additions is one

where the program samples the temperature every minute. If the temperature is greater than a base value, the program reports the node ID. After some time or as the result of a trigger condition, the gateway node sends an update message to change the base comparative value. The corresponding query program is as follows:

```
1   .section delivery, "@60's
2   get_temp            # sample the temperature sensor
3   iload_gl1           # read global variable 1
4   if_cmpgt   send     # send if temp> global variable
5   goto_exit           # if not exit the program
6   send:  get_nodeid          # read the node's ID
7   istore 0            # store it in transmission buffer, pos. 0
8   send_tb             # send the tran2mission buffer

9   .section update
10  iload 0             # read first byte of message data
11  istore_gl1          # store that value in global variable 1
```

Lines 1, 11, and 14 declare the sections defined in the query program. The delivery section is scheduled to execute one every 60 seconds. At that point, the temperature is read and compared to the value stored in global variable 1 (Lines 2,3). If the temperature is greater than the stored value, the node id is read (Line 7) and copied to the first byte of the transmission buffer (Line 8). The transmission buffer is sent to the parent node (Line 9). When an update message is received, the update section is invoked. The data from the update message is stored in the transmission buffer. The first byte of the transmission buffer is read and stored in the global variable (Lines 12, 13). The update section can execute any instructions available to all other nodes. An update message can be sent to trigger the sample of sensor data and the transmission of the result message to the parent node.

4.3.3 'Get_Fuzzyvalue' Instruction

The SwissQM query manager application is a stack-based, integer virtual machine. It uses a small subset of the integer and control instructions of the Java Virtual Machine (JVM) specification [16]. The current instructions provide stack manipulation, arithmetic, logic, and control instructions such as conditional and unconditional jumps, access to the sensors and access to the local storage buffer and the local message transmission buffer. SwissQM employs a modular implementation for its instructions. The modularity is achieved by organizing instructions into groups according to their general function, such as buffer instructions or attribute instructions. Each group is implemented as a separate nesC component.

To support fuzzy queries, I implemented a new group for fuzzy instructions. Fuzzy membership evaluations are implemented with the get_fuzzyvalue instruction.

Get_fuzzyvalue(sensorValue, mbrFunc, $parm_1$, ... $parm_m$)

The first parameter is the sensor value to be evaluated. The second parameter identifies the membership function to be used. The number of remaining parameters varies depending on the arguments required by the membership function. An example of coding the instruction follows:

```
1  #  S Function:
2  .section delivery, "@60's
3          iload_gl3          # push fuzzy membership parameter b on the stack
4          iload_gl2          # push fuzzy membership parameter m on the stack
5          iload_gl1          #  push fuzzy memership parameter a on the stack
6          ipushb  4           # push  fuzzy membership function – S-function
7          get_temp           # push  senor value on the stack
8          get_fuzzyvalue    # calculate and push  the fuzzy value on the stack
9          store 1             # store in transmission  buffer
10         get_nodeid         # push the node id on the stack
11         store 0            #store node id in transmission buffer
12        send_tb            @send the trannsmission buffer
```

In this example, the delivery section is scheduled to execute one every 60 seconds (line 2). Lines 3, 4, 5 read the values stored in the global variables. Line 6 pushes an encoding value on to the stack to indicate that the fuzzy membership function, S-function, is to be used. Line 7 caused the temperature sensor to read the current temperature value. Line 8 executes the fuzzy membership function using the previous values as arguments. The resulting fuzzy value is stored in the transmission buffer, line 9. Lines 10, 11 copy the node ID into the transmission buffer. Line 12 sends the tranmission buffer to the parent node.

Fuzzy membership functions implemented are:

TABLE 1: Implemented Fuzzy Membership Functions

Membership Function	Encoding value	Parameters	Operand Stack
Triangular	1	[sv,1,a,m,c]	If the current stack is α:, then adding the fuzzy value parameters and function identifier the stack becomes: *fuzzy function encoding value* *sensor value* *parm-1* ... *parm-m* α After the membership function completes, the stack becomes: *fuzzy value* α
L-Function	2	[sv,2,a,b]	
Trapezoidal	3	[sv,3,a,b,c,d]	
S-Function	4	[sv,4,a,m,b]	
Z-Function	5	[sv,5,a,m,b]	
PI-Function	6	[sv,6,a,b,c,d]	

The SwissQM query manager application provides only a single data type: a signed 16-bit integer type. This is sufficient for now as the raw data returned by sensors is typically an integer value from an Analog/Digital converter. To compensate, I altered

the fuzzy universe range from [0, 1] to [0,100]. New functions can be easily added to this implementation. However the integer restriction limits the functions that can currently be implemented.

SwissQM runs optimized programs written in a lower level interface than standard query languages. Query compilers can be directed to emit SwissQM programs. As a result, SwissQM does not make any assumptions about the query language used (e.g., SQL or Xquery). By extending the base support for bytecode programs to process fuzzy queries, I provde an interface for any query language compiler running optimized SwissQM bytecode programs to represent fuzzy queries. Extensions to the industry standard Structured Query Language (SQL) have been proposed to support fuzzy query processing. These include SQLf [70 and FSQL [10,49]. XML schemes [49] have also been proposed for fuzzy querying. To accommodate sensor networks, extensions to the high-level query language SQL [8, 9, 11, 58, anc 59] have been used as other specialized interfaces for WSN querying. These interfaces for WSN can be extended to include the previously proposed interface support of fuzzy queries. Noting that, here is an example of how an SQL query can be implemented in SwissQM:

SELECT nodeID, fuzzyTemperatureValue
FROM sensors
WHERE fuzzyTemperatureValue > 0.7

The corresponding SwissQM query code follows. The example is a variation of the previous coding example:

```
1  .section delivery, "@60's
2        iload_gl3        # push fuzzy membership parameter b on the stack
3        iload_gl2        # push fuzzy membership parameter m on the stack
4        iload_gl1        #  push fuzzy memership parameter a on the stack
5        ipushb  4        # push  fuzzy membership function – S-function
6        get_temp         # push  senor value on the stack
```

```
7         get_fuzzyvalue   # calculate and push  the fuzzy value on the stack
8         dup              # repeat the fuzzy value on the stack
9         ipushb   7       # push the comparator value 7 on the stack
10        if_icmplt  exit  # if the fuzzy value < 7 go to the exit label
11                         # otherwise, prepare the transmission buffer
12        store 1          #  store in transmission  buffer (remove from stack)
13        get_nodeid       # push the node id on the stack
14        store 0          # store node id in transmission buffer
15        send_tb          # send the transmission buffer
16  exit:                  # exit label
```

In this example, the delivery section is scheduled to execute one every 60 seconds (line 1). Lines 2,3,4 read the values stored in the global variables. Line 5 pushes an encoding value on to the stack to indicate that the fuzzy membership function, S-function, is to be used. Line 6 caused the temperature sensor to read the current temperature value. Line 7 executes the fuzzy membership function using the previous values as arguments. Line 8 duplicates the fuzzy membership value on the stack. One value is removed in the comparision operation on line 10, which comparies the fuzzy value with 0.7. Since SwissQM only allows integer values both the calculated fuzzy values and the query fuzzy values are adjusted. If the fuzzy value is less than 0.7 (adjusted to 7.) then the program exits. Otherwise, the fuzzy value is stored in the transmission buffer at offset 1 in line 12. Lines 13,14 read and store the node ID in the transmission buffer. Line 15 sends the tranmission buffer to the parent node.

CHAPTER 5: SIMULATING DYNAMIC ENVIRONMENTS

It is advantageous to employ a cooperation of both the network application simulation and a separate environment simulation to evaluate WSN solutions. I have developed a novel simulation architecture [97] that allows environment scenarios to evolve independently of simulation models for network protocols and topology. This scenario informed approach to simulation, which incorporates dynamic changes in the underlying environment, provides techniques to evaluate WSN middleware under unpredictable behaviors. Using cellular automata (CA) to represent dynamic physical environments, this architecture provides a generalized model that can be used to build a rich variety of models, to configure, simulate and evaluate middleware and applications.

In the following sections, I introduce common CA concepts and explain decisions concerning my CA implementation (Section 5.1). I document the design of the architecture I built for the environment scenario as an extension of the popular WSN simulator, TOSSIM (Section 5.2). In Section 5.3, I describe how environment scenarios can be built using this architecture and three scenarios I built to display the versatility of the architecture. The scenarios used to evaluate fuzzy query processing are described in Section 5.4.

5. 1. Overview of Cellular Automatae

Cellular automata, first introduced by von Neumann (1966), represent simple mathematical idealizations of the natural world. It consists of a lattice or grid of identical regions, where the physical quantities or measurement values of each region are represented by a finite set of values [123,124]. Values evolve in discrete time steps with respect to established system rules. Rules are well-defined functions and given an initial configuration, generate a consistent evolution of state changes in terms of the values of the neighboring regions, as well as internal conditions. The discrete time evolutionary changes of interdependent regions result in a model where the complexity of the system as a whole exceeds the complexity of each automaton [11].

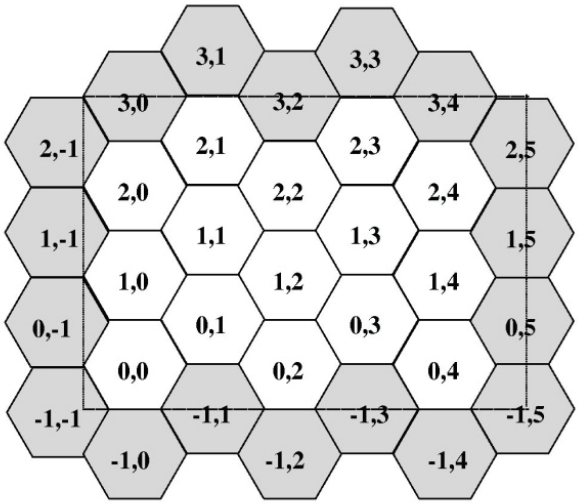


FIGURE 22: Hexagonal cellular automata grid

A Cellular Automata can be formally defined [105, 123] as a 4 –tuple (*L,S,N,f)* where *L* is a regular grid or lattice whose elements are called cells; *S* is a set of finite states which can be assigned to any cell; *N* is a finite set of indexes representing the neighborhood of a cell; and f is the transition rule which specifies the time evolution of the states of a given cell.

In defining a grid or lattice for a CA, cell geometry must be selected. The cell geometries determine how a neighborhood is established. The geometry of a

neighborhood will define how environmental events are propagated and how the influence of neighboring cells is determined. There is no restriction to the size or radius of a neighborhood. However, for simplicity it is generally made up of adjacent or near adjacent cells only.

The basis for my simulation model is a 2D grid of regular hexagonal regions. Cells or environmental regions are identified by both a row, column coordinates of the region with respect to the other hexagons in the grid and by the (x,y) grid coordinates of their center as shown in Figure 22. Each region manages and maintains the state of a set of attribute values which reflect the environment conditions of that cell. Each region is assigned a behavior object that includes the transition function for the cellular automata system. Therefore each region is able to examine the state of the neighboring regions and evolve accordingly.

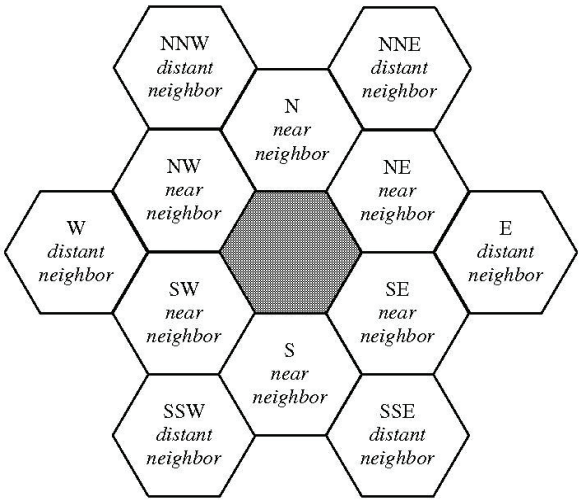


FIGURE 23: Layout of neighboring regions for hexagon cell.

Hexagonal geometry was chosen because this geometry allows isotropic influence of environmental conditions around each cell. Hexagons are oriented as shown in Figure 23. With this orientation, hexagonal neighborhoods are addressed as north, south, north-east, south-east, north-west and south-west. The neighboring region can be identified by simple calculations on the region row, column coordinates. For example the North

neighbor is found by adding one cell position to the row of the row, column coordinates of a region.

When simulating a given CA, it is impractical to consider an infinite grid. The gray cells in Figure 21 are identified as 'boundary' cells. These may extend outside of the grid space. Therefore, transition rules pertaining specifically to the behavior of the boundary cells of the environment scenario must be established. A cell defined as a boundary cell, can determine a state for its virtual neighboring cell by one of several solutions [101], each with a different behavior (Figure 24): Periodic or cyclic boundary conditions suppose the grid is embedded in a torso-like topology. Boundary cells select neighbors by assuming the left and right sides of the grid are connected, as are the upper and lower edges. Fixed boundary conditions are defined so that the neighborhood is completed with cells having pre-assigned values. Adiabatic boundary conditions are obtained by duplicating the value of the site to the extra virtual cells. Reflecting boundary conditions copy the value of the other neighbor to the virtual cell. The environment scenario model allows boundary cells to be defined with either as periodic/cyclic or the adiabatic. That is, when a boundary cell is addressed, the environment model actually refers to the non-boundary cell that would be adjacent were the grid a continuous circle.

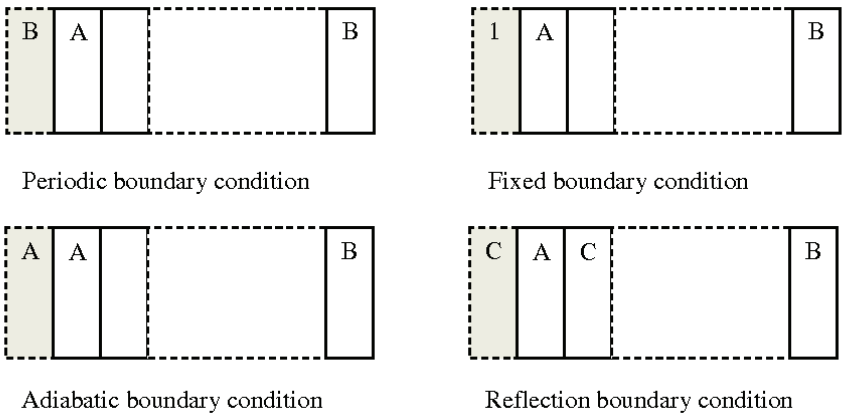


FIGURE 24: Cellular automata: Types of boundary conditions

Simulated sensor nodes or motes are registered to the region they are monitoring. The position of a mote is identified by its x,y coordinate. When the cellular automata simulation model detects the presence or the movement of a mote, the proximity of the mote's position to the cell's x,y center determines in which cell the mote will be registered. If the mote is in a boundary cell, it is registered to the non-boundary cell whose center x,y coordinates are the closest to the mote's position. In this way, I create a variation of the adiabatic boundary condition where the state of is obtained by duplicating the value of the contiguous non-boundary cell. When attributes representing environmental values change within a region to which a mote is register, that region issues a simulator event to the event queue, which causes the values to be set in the respective analog-to-digital conversion (ADC) port of the mote object. When the application or middleware running on the simulated mote queries environment data sensed through that port, the value assigned to the ADC port is returned.

Transition rules are implemented in the behavior of the cell. The transition rule or function specifies the time step evolution of the state of a given cell. It is dependent on the cell geometry, the neighborhood and the state set. There are several considerations involved in determining a transition rule for a given cell or the rules as they may vary between cells. A transition rule can be direct (pre-selecting the outcome for each possible configuration of states in a neighborhood), or it may be formulaic (requiring a detailed rule to be evaluated at each cell).In classical Cellular Automata Theory [85] a rule can depend only on the states of its neighboring cells (totalistic rule) or it can also use the cell's immediate state (outer totalistic rule). Another classification to consider regarding the transition function is to distinguish between deterministic or probabilistic rules [101, 85]. A deterministic rule will have exactly one outcome for each neighborhood configuration. Given an initial configuration, the CA will always evolve the same way. A probabilistic rule selects an outcome from several possible states associated with some probability function. Probabilistic cellular automata offer a way to adjust rule parameters to accommodate a continuous range of values despite the discrete nature of the cellular automata world. This allows a better modeling of physical systems in which particles are annihilated or created or attributes are otherwise changed at some given rate

5. 2 Environment Scenario Architecture

The architecture of the dynamic environment scenario model can be built over any discrete WSN simulator that allows applications to interface to the event queue, create application specific events and update the analog to digital converter (ADC) sensor port on the simulated motes. To create environment scenario models for simulation analysis, I implemented an environment scenario model simulation model as an extension to the TinyOS [47] Simulator (TOSSIM) [45**].** TOSSIM is a discrete event simulator based on TinyOS wireless sensor networks. TinyOS is an operating system designed specifically for sensor motes. The TOSSIM architecture is composed of 5 parts: support for compiling TinyOS components into the simulation infrastructure; a discrete event queue; mechanisms for extensible radio and ADC models; communication services for external programs to interact with a simulation; and a small number of re-implemented TinyOS hardware abstraction components. TOSSIM takes advantage of the TinyOS structure and whole system compilation to generate discrete-event simulations. It runs the same code that runs on the sensor hardware. By replacing a few low-level components, TOSSIM translates hardware interrupts into discrete simulator events; the simulator event queue then delivers the interrupts that drive the execution of the application. Sensors are modeled in detail at the level of ADC ports. TOSSIM provides scalability: handling large networks of sensor nodes in a wide range of configurations, completeness: running the same code that is compiled for the actual hardware ensures the simulator closely captures complete application behavior, and fidelity: providing a high fidelity bit-level simulation of the communication radio

stack, it captures the reactive nature of sensor networks, including the subtle timing interactions between motes. The simulation is managed by external built using the provided scripting language, Tython [68]. Tython is based on Jython [154], a Java implementation of the Python language and exploits the full expressive power of Python to access the TinyOS tool chain.

Figure 25 illustrates the enhancements that I made to the TinyOS/TOSSIM architecture to implement my dynamic environment scenario simulation model. My dynamic environment scenario simulation model is implemented in 5 class objects:

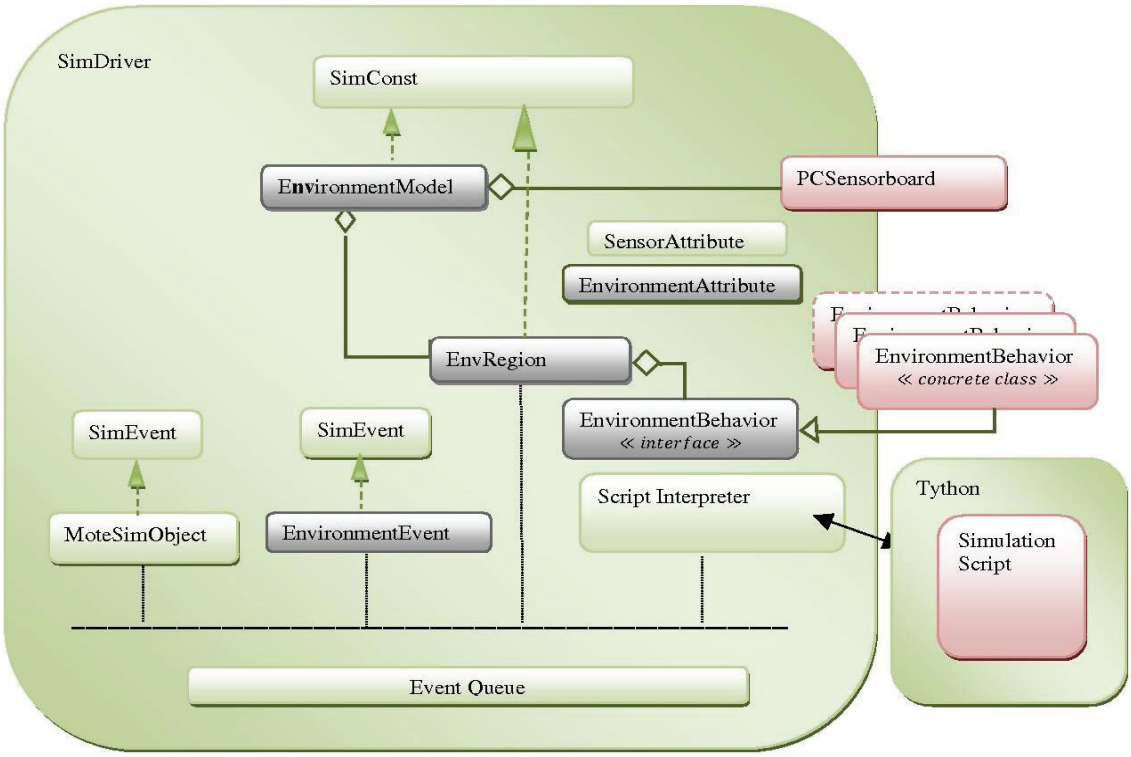


FIGURE 25: Architecture of Environent Simulation Design

EnvironmentModel, *EnvRegion*, *EnvironmentBehavior*, *EnvironmetAttribute,* and *EnvironmentEvent,* as shown in 23. The *EnvironmentModel* class takes as input the length of one edge of each regular hexagon. The height and width of the environment space is determined by the size of the simulator 'world' established by TOSSIM parameters The environment grid is created instantiating and positioning the hexagon cell objects. Once created, *EnvironmentModel* maintains addressability to each hexagonal cell comprising the grid.

EnvironmentEvent objects implement the TOSSIM event interface for event structure and accessibility. *EnvironmentEvent* objects are classified by type and subtype. Specific types are defined for control in the simulation architecture. An application type is also defined for use by simulation scenarios. This allows application scenarios the ability to alert regions to global conditions or to signal a neighboring region when an internal state change in one cell dictates action by another.

Hexagonal cell objects are instantiations of the *EnvRegion* class. The *EnvRegion* class requires the edge length of regular hexagon it is to represent, and the row, column coordinate position of the cell within the environment grid. The *EnvRegion* object maintains addressability to its behavior object, a list of associated environment attributes and values defined and regulated by the behavior object, and a list of any motes to this region. Each region monitors the event queue for any simulation events that indicate a mote has been re-positioned, registering or unregistering the mote as required. Each region also monitors the event queue for all environment events issued. If the environment event indicates that the region

requires action, its associated behavior object is invoked. *EnvironmentAttribute* objects define the environment attributes that are specific to a region. Each region object stores the current state of the sensor attributes and relevant environment attributes.

The Strategy design pattern [69] is used to implement the behavior object. An *EnvironmentBehavior* interface class is provided defining the methods required for region behavior. A behavior object is then assigned to each hexagonal cell object. The same behavior object implement the environment actions with all cells or different behavior objects can be used to define an environment topology over the grid. The behavior object must: (1) define the initial state of all environment attributes for the cell; (2) provide a method to update the environment attributes; (3) provide a method to handle any *EnvironmentEvent* objects that require activity by this cell. Scenarios can be developed by creating concrete class object for the *EnvironmentBehavior* that define the logic of the cell behavior.

5. 3 Building Environment Scenario Applications

An advantage of the Environment Scenario architecture that I developed is that it allows researchers and developers to create interesting environmental simulation scenarios tailored to the needs of their work. It also provides a mechanism for developing reproducible experiments. Herein I describe the components and interfaces provided to assist in creating a customized simulation environment scenario: 1) description of the simulated mote sensor board; 2) attributes objects to define sensor and non-sensor attributes for each region; 3) interface object to define the behavior of each region, 4) user events to communicate between regions.

When building a cellular automata simulation of an environment scenario, a description of the sensor board of the mote being simulated and concrete behavior objects for each hexagonal cell are required. A script or other program can be used to initiate and control the runtime of the simulation *PCSensorboard*. *PCSensorboard* class object defines the attributes sensed by each port of the simulated mote. A *Pcsensorboard* object representing a mica2 mote was used in my simulations.

SensorAttribute class objects, currently provided by TOSSIM, contain the values for the sensors. Sensor values are typed as *int* only as the sensor devices currently simulated by TOSSIM return only *int* values. When a sensor attribute is changed through region object, the environment scenario simulation, specifically the *EnvRegion* object, will automatically update the sensor port on any simulated mote currently positioned in that region.

EnviornmentAttribute class objects, provided with the environment scenario simulation, contain the values specific to an environment region. The data types for these attributes are more inclusive than for *SensorAttributes.* These attributes can be any class Object type. This gives more flexibility in defining, retaining and retrieving information needed to maintain and evolve the environment scenario.

EnvironmentBehavior and *EnvironmentEvent,* class objects work together to implement the environment changes and evolution of the scenario. *EnvironmentEvent,* allow cells to communicate with neighboring cells, initiating action of the neighboring cell when required. Transition rules are implemented in the behavior objects. Behavior objects manage the sensor and environment attributes. These are then stored in the associated region *EnvRegion* object. . Behavior objects can address other regions (*EnvRegion*) to determine their state. To propagate changes through the environment model, behavior objects direct events to its own region, other regions or to all regions. These event objects include a user defined subtype field that can be managed by the cell behavior object. Different scenarios can define an event subtype to manage transitions specific needs of that simulation application. Event data consists of the identifier of the targeted region and any relevant information to be transmitted. This data can be typed as a byte array, or a Vector or ArrayList object.

The following section illustrates how an environment simulation scenario is built.

5.3.1 Modeling Spreading Fire Scenario

The following describes how I use my Environment Scenario Architecture to create a model of Fire Spreading throughout a park terrain. I explain the CA model, the classes needed to implement the scenario and state changes these classes effect in the environment model.

Description of problem and query: A park terrain consists of a grassy plan, a slight incline with shrubbery and a hill with trees. There are also two (2) small lakes. Sensors are distributed through the park. A fire is ignited in a park. The application wants to monitor and report the changing perimeter of the burning area.

Purpose of the test: This will test the ability of a sensor network to find the changing perimeter of a fire. However, it is included to demonstrate the flexibility of the environment scenario architecture. This scenario demonstrates regions affecting both near and distant neighbors. Also, it includes an external environment condition pervasive over the entire scenario. In the case of this model, the wind direction affects the spread of the fire.

Cellular Automata Evaluation Model: Encinas, et al [152] proposed a model for the prediction of forest fires spreading based on the use of hexagonal CA. It incorporates weather (wind) and land topology conditions. The state of each region is defined by its fractional burned area:

EQUATION 11: Definition of state for fire spread model

$$state_{ab}^{(t)} = \frac{burned\ out\ area\ \ of\ \ region(a,b) at\ time\ t}{total\ area\ of\ region\ (a,b)}$$

If $\text{state}_{ab}^{(t)} = 0$, then the region (a,b) is unburned at time t; if $0 < \text{state}_{ab}^{(t)} < 1$, then the region (a,b) is partially burned out at time t; finally if $\text{state}_{ab}^{(t)} = 1$, the state is completely burned out at time t. Encinas' work maps states of regions onto the set StateValues = {0,0.1,…,0.9,1}. Any state value greater than 1 is taken to be equal to 1. The dynamic of this model supposes that the state of region (a,b) at time t+1, linearly depends on the states of its near and distant neighboring regions at time t, specifically:

EQUATION 12: Transition rule for fire spread model

$$state_{ab}^{(t+1)} = f\left(state_{ab}^{(t)} + \sum_{\alpha\beta} \mu_{\alpha\beta}^{(a,b)} state_{\alpha\beta}^{t} \right)$$

where $\mu_{\alpha\beta}^{(a,b)}$ are parameters involving physical magnitudes of the regions and the discretization function f maps the result onto the state value set: $f:[0,1] \rightarrow \text{StateValue}$. In this model, each region (a,b) represents a small hexagonal region of a forest and is endowed with three environmental attributes: the rate of fire spread for the region, $R_{a,b}$, the wind speed over the region, $W_{a,b}$, and the height of the region, $H_{a,b}$. Consequently, the parameter $\mu_{\alpha\beta}^{(a,b)}$ is evaluated as $\mu_{\alpha\beta}^{(a,b)} = w_{\alpha\beta}^{(a,b)} * h_{\alpha\beta}^{(a,b)} * r_{\alpha\beta}^{(a,b)}$, where $w_{\alpha\beta}^{(a,b)}$ stands for the wind influence of neighboring region (α,β) on region (a,b); $h_{\alpha\beta}^{(a,b)}$ stands for the influence of the height difference between region(α,β) and region (a,b) (fire travels faster uphill); and $r_{\alpha\beta}^{(a,b)}$ stands for the influence of the different rates of fire spread between the regions (α,β) and (a,b) .

Overview of Environment Scenario: Each region is instantiated with a height, rate of spread value. Wind is defined as a homogeneous linear front from north to south. Only wind influence from northern neighbors contributes to the state of the region. The forest is not flat and the rate spread is not homogeneous. The height influence function and rate spread used are those suggested by [152]. To initiate the test, an environment event is sent to selected regions to ignite a fire. The region's state is changed from unburned (state=0) to partially burned (state >0.0) . The region informs each of its near and distant neighbors that a burn has started. Those neighbors interrogate the states of their near and distant neighbors to calculate their own state. Each region where a burn or a burn watch has started, will then periodically recheck its state by examining its

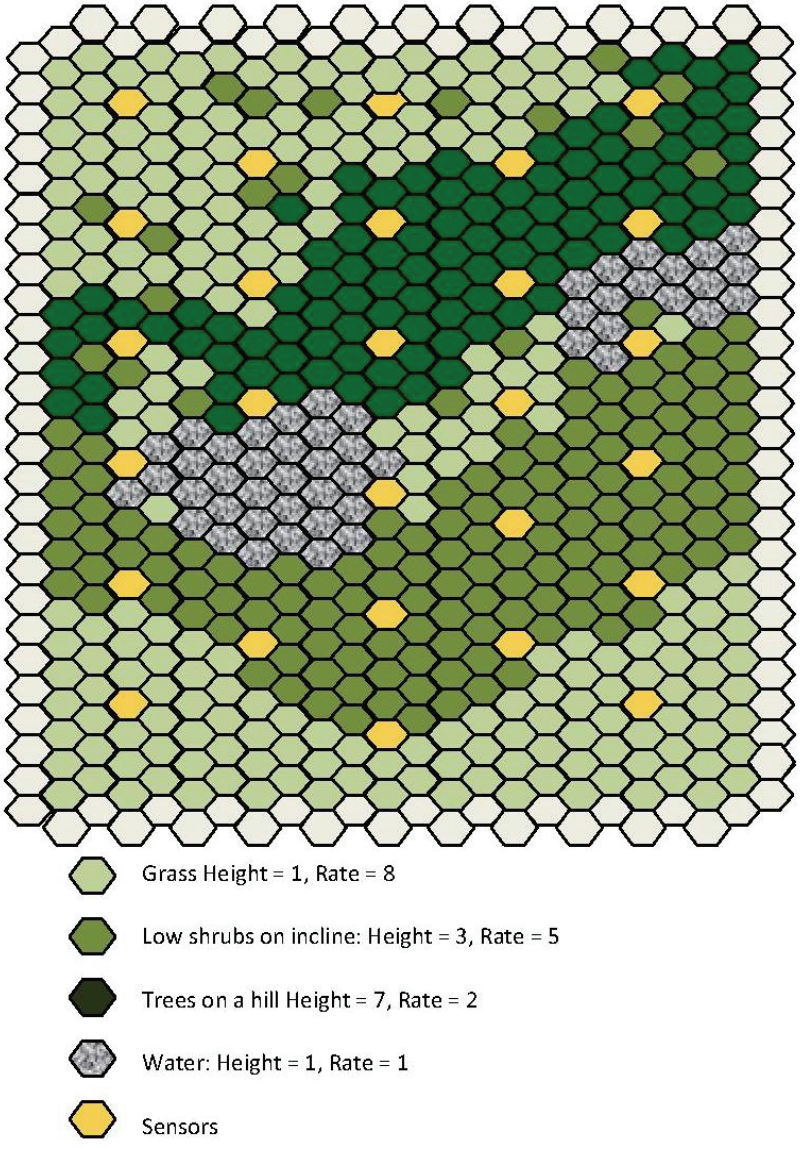


FIGURE 26: Fire Spread CA Model

own state and interrogating neighbors. When any region determines a state value ≥0.2, that region is set to 'ignited' status, informs its neighbors that a burn has started. The scenario repeats until all regions are burned completely or until the scenario is manually ended.

Environment Scenario Design: The sensor node used is a mica2 mote. The *PCSensorBoard* class is defined with the sensor ports available on a mica2 mote:

TABLE 2: Sensor ports on Mica2 mote

Port	Sensor Type
1	light
2	temperature
3	sound
4	acceleration on the X axis
5	acceleration on the Y axis
6	magnetic field X component
7	voltage
8	magnetic field Y component

The mote does not have a port for reading "fraction of area currently burned out". So, I choose port 2, temperature sensor, to represent the sensor that will read the value. When the region in which the mote resides changes state, it signals a change to the value read by port 2.

The *SensorAttribute* class identifies the temperatue sensor or 'temp'. The environment attributes, associated with the environment region, are used to manage the region state. For this scenario, *EnvironmentAttributes* classes are required for the height, rate of burn .and wind states. Each region manages its own instantiated *SensorAttribute* and *EnvironmentAttribute* objects for these classes.

Four *EnvironmentBehavior* classes are used, one for each region type: grass,

low shrubs, trees and water. These create the *SensorAttribute* needed and also create and inialize the associated *EnvironmentAttributes* (height, rate of burn .and wind.) The *EnvironmentBehavior* class defines the transitional state change alogithm (Equation 12). The sensor nodes deployed in the scenario are registered to the region in which they reside. If a sensor node moves, it is unregistered from the region it leaves and registered in the region to which it is moved. If a region contains a sensor node, it updates the appropriate ADC port of that node when the state of the *SensorAttribute* changes. Each region manages its own instantiated *EnvironmenBehavior* object.

EnvironmentEvents are used to communicate between regions and initiate state evaluation or state changes. This scenario defines three different *EnviornmentEvents* to 1) ignite or continue burn state; 2) notify neighbors of a change of state conditions (signalled only if the region is in burn state); 3) schedule an recalculation of the state of the current region. This recalcultion of state includes using state condition of this state and state conditions of neighboring states. To get state conditions of neighboring states a region makes a direct request to the region object

The state diagram (Figure 27) shows the behavior of each region. If a region receives an 'Ignite' event that region changes its state to 'Burning'. It notifies its neighbers by a 'Notification of Neighbor Burn' event. Also signals a "Schedule Recalculation of State' event to itself to adjust is state based on change increasing its burn state and incorporating the current state of its neighbors at future timestep. Once the region is completely burned, it quiesces to a 'Burned Out' state and no

longer listens for other events.

If a region receives a “Notification of Neighbor Burn” event, it changes state to ‘Burn Watch’. The state “Schedule Recalculation of State’ event to itself to adjust is state based on the states of itself and its neighbors at future timestep.The change in state of its burning neighbors will affect the state of this region. Once this region reaches a state that defines ‘burning’, it changes its state to ‘Burning’.

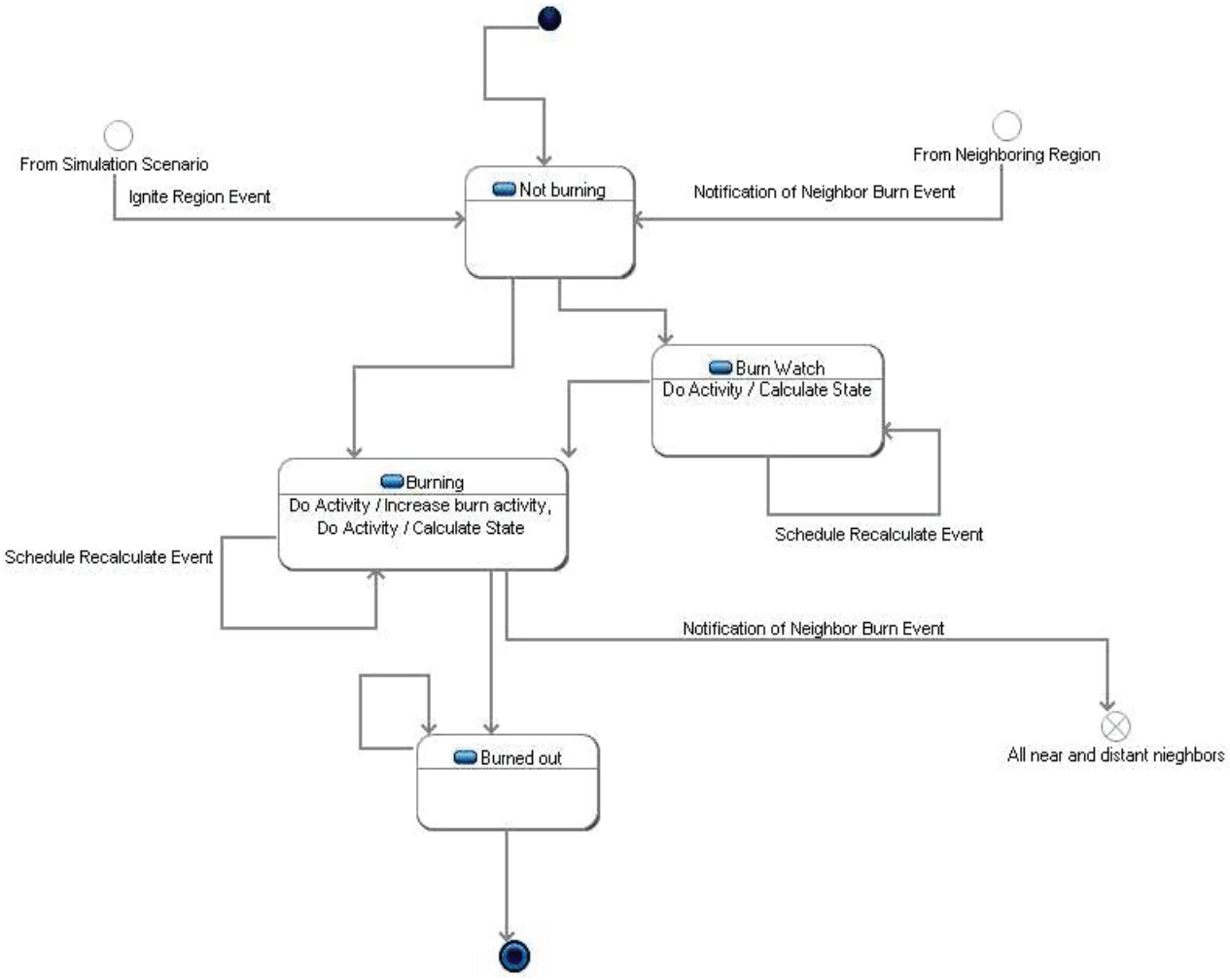


FIGURE 27: State Diagram for Fire Model Scenario

I have used the Fire Spread Model to illustrate the process of creating an Environment Scenario. This model maps the state of each region onto the set [0,1] representing the amount of the area that is burned. It does not provide measurable environment attributes. Therefore, it is not used as an evaluation test for fuzzy queries. In the next section, I present two environment scenarios that will be used to evaluation to evaluate fuzzy query processing: the Dam Break Scenario and the Gas Particle Flow Scenario,

5.4 Environment Scenario Simulations Used In Evaluation Study

I conducted three sets of evaluation studies with two primary objectives: 1) to demonstrate proof of concept for using fuzzy queries over interval functions and threshold functions in WSNs; and 2) to compare the efficiency and accuracy between fuzzy queries and classic (non-fuzzy) queries over WSNs. The environment scenario for the first two tests uses the model: 'Break in a dam wall.' The environment scenario for the third test uses the model: 'Flow of gas particles through a space." The details for the models are described in the next two subsections.

5.4.1 Scenario 1: Break In Dam Wall

Description of application problem: The city has recently rebuilt part of the dam structure across the local river. Civil engineers have determined the strength of the old part of the dam, the new part and the seam between the two. They have placed a sensor network throughout the structure. The network is to query the measured pressure and report back if the structure is 'close to' rupture. In the river bed, there are 3 beams that keep banging against the dam wall in a way that increases in strength with each pulse. Eventually, they rupture the wall. After which the banging continues and the rupture space increases. The WSN will monitor the stress level and report when the stress level is dangerously close to rupture, by sending a message with stress information.

Cellular Automata Evaluation Model: The OFC or slider-block model is a CA designed to simulate stress loading and rupture cycles [102], as well as earthquakes. In the slider-block model, changes in environment conditions in one region cause changes in neighboring regions. Each region is initialized with a threshold value for the amount of stress it can tolerate before it ruptures. In a selected area(s) of the grid, the stress on is uniformly increased. When the stress load in a region exceeds its threshold strength, the region ruptures, propagating a proportion of its stress onto its nearest neighbors. If the

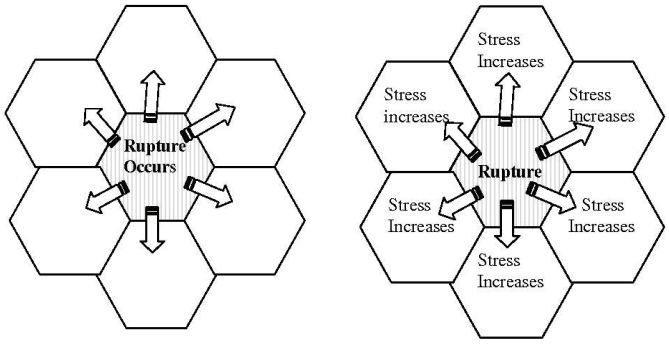


FIGURE 28: Stress propagation following

stress load on these regions now exceeds their threshold strength, they will also rupture. This process continues until no regions are above their threshold strength. All ruptured regions have uniform stress at the start of the next loading cycle.

My implementation, based on [88], employs a behavior object that generates pseudo-random region strength at instantiation ensuring the regions will rupture at various times. Stress is incremented by a constant size at a constant rate. An environment event is issued to select regions of the grid to signal an increase in stress. If a region ruptures, that region issues another *environment event*, propagating its stress overflow to its neighbor. That region incorporates the stress overflow, as well as any constant stress increment it receives, until it ruptures and repeats the stress overflow propagation.

An environment event is then sent to repeat this scenario until a termination environment event, signaled at the start of the scenario is processed.

5.4.2 Scenario 2: Flow of Gas Particles Through a Space

Description of the application problem: Gas is flowing through an enclosed space. It enters through a vent in the left wall and exits on through a vent in the right wall. The application desires the space to have a controlled level of saturation of this gas, however flow must be constant. The vents on either side may be opened or closed. The experiment starts with the exit vent closed. When the saturation level reaches a certain point, the vent in the right wall must open and when the saturation level is below a certain point, the vent in the right wall must close. Inversely, the vent on the left wall must open and close to control the increase the level of gas particles. To insure constant flow, one vent must be open at all times.

Purpose of the test: This scenario compares the effectiveness of fuzzy versus classic queries with respect to the environment conditions. The queries must manage the general saturation level of the particles in the region while accommodating for the expected microclimates around the vents. The saturation of the area close to the exit vent will tend to be less than that in the greater space when the exit is open. Similarly the saturation of the area close to the entrance vent will tend to be greater when the entrance vent is open. Queries will be sent to the WSN to request repeated monitoring of the saturation level in the area. Using data returned from the nodes, the gateway node will signal the environment to open or close the vents

Cellular Automata Evaluation Model: The Frisch, Hasslacher and Pomeau (FHP) model [100] is used to simulate diffusion of gas particles. The FHP

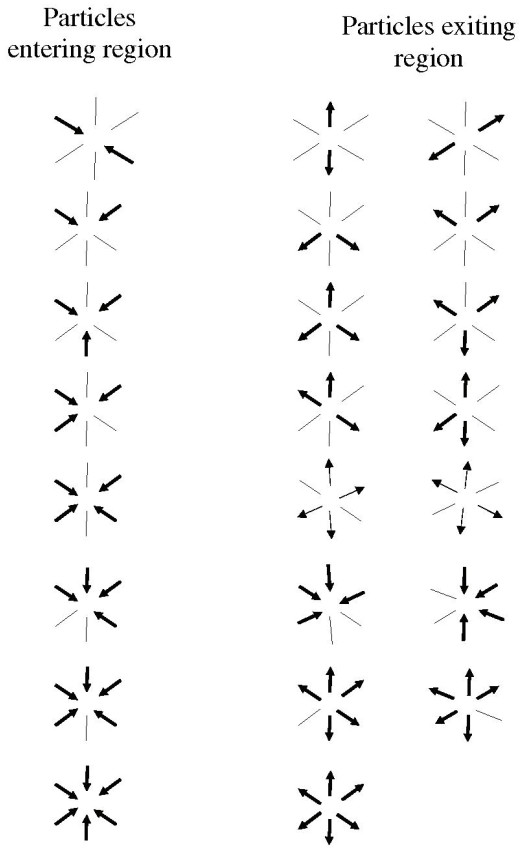


FIGURE 29: Collision resolution for saturation scenario

model is an abstraction, at a microscopic scale, of a fluid. It is expected to contain all the salient features of a real fluid. The microdynamics of the FHP model have been shown to include the conservation of particles and momentum [10] as required by a representation of fluid dynamics. In this scenario, the presence or. Absence of particles within a region is represented in terms of Boolean variables. This model describes the motion of particles traveling in a discrete space and colliding with each other. The FHP particles move along one of the six directions of the grid, such that in one time step, each particle travels one grid space and reaches a nearest neighbor region. In the absence of any collision, the particle moves in a direction of a straight line. Interactions take place among particles entering the same region at the same time and result in the particle changing direction along the grid. For instance, when

exactly two particles enter the same region in opposite directions, both particles are deflected 60 degrees so that the output of the collision is still a zero momentum configurations. When three particles enter the same region with an angle of 120 degrees between them, they bounce back to the sending region.

CHAPTER 6: EVALUATION OF FUZZY QUERY PROCESSING

I conducted three sets of evaluation studies with two primary objectives: 1) to demonstrate proof of concept for using fuzzy queries over interval functions and threshold functions in WSNs; and 2) to compare the efficiency and accuracy between fuzzy queries and classic (non-fuzzy) queries over WSNs.

In this chapter, I present the results of the tests run to evaluation fuzzy query processing. Using the scenarios discussed in Chapter, I repeat the description of the problem addressed by each test. Section 6.1 explains the environmental setup for the study. In section 6.2, I explain the measurement criteria used in evaluation. Section 6.3 presents the findings comparing a fuzzy query to a classic query in a constant homogeneous environment. This verifies the efficiency and accuracy of a fuzzy query with respect its classic query counterpart. It shows that given the same environment and equivalent selection criteria, the fuzzy query is as accurate and efficient as the classic query. Section 6.4 presents finding comparing a fuzzy query to a classic query in a constant, heterogeneous environment. This test shows that in a heterogeneous environment, the fuzzy query can perform more efficiently at the cost of some accuracy. Both these tests use the scenario ‘Break in a dam wall’. Section 6.5 presents finding comparing fuzzy and classic queries in a dynamic environment. Using the environment scenario “Flow of gas particles,” it compares the class query and the fuzzy query to conditions of the environment.

6.1 Experimental Setup Design

The evaluation studies were run on the TOSSIM simulator with the assumptions that all nodes are stationary and the sensor network is not partitioned; all sensor nodes are homogeneous in their capabilities; and all communications are symmetric. Simulated sensor nodes were deployed in differing configurations. Additional details are as follows.

WSN Mote Assumptions: The sensor nodes are assumed to be resource constrained devices, running on batteries. The simulated sensors perform actual data acquisition. When attributes, representing environmental values, change in the running scenario, those changes are reflected in the respective analog-to-digital conversion (ADC) port of the mote object. When the application or middleware running on the simulated mote queries environment data sensed through that port, the value assigned to the ADC port is returned.

WSN Mote Deployment: Random Deployment is calculated by predetermined seeds to provide repeatable pseudo-randomness of mote placement. Uniform deployment is determined by pre-set positions of the nodes on the environmental grid.

WSN Network Model: Existing networking, routing and data forwarding techniques from SwissQM will be used. The SwissQM sensor network is built using a gateway node and one or more sensor nodes. The gateway node is assumed to have sufficient computing power and no energy or memory restrictions. The gateway acts as the interface to the system. The sensor nodes are assumed to be resource constrained devices, running on batteries. The sensors perform the actual data

acquisition.

The sensor nodes are organized into a tree that routes data towards the root, where the gateway node is located (Figure 1). This strategy facilitates in-network aggregation and reduces the amount of routing data a node has to keep. Every node has a link to a parent closer to the gateway. This study assumes a single tree although SwissQM is built to support multiple independent trees and individual node addressing. The SwissQM routing tree is formed using the Mint routing protocol [97, 122] and is used to send result data back to the gateway. Every node in the tree maintains a link to a parent node closer to the root. Thus, a result message is sent to the parent node, which will then forward the message to the next node closer to the root. The protocol establishes a link quality estimator that is based on a window mean with exponential weighted moving average (WMEWMA) of the success rate. The success rate for a link is estimated from the number of packets successfully received over that link divided by the number of expected packets for a given period of time. The number of packets is obtained by routing messages that contain routing table entries the nodes periodically exchange (in SwissQM, every four seconds). These tables are then used by the nodes to select their parent nodes.

"Perfect" Communication is assumed. That is, there are no dropped messages or broken links. No communication parameters are varied in any of the experiments.

Environmental Test Scenarios: The environment scenario for the first two sets of studies models a break in a dam wall, and the environment scenario for the third set of studies models the spread of gas particles. The details for the models are

described in the following subsections.

6.2 Evaluation Measurement Used

The goal of the evaluation of the fuzzy query processing approach is to validate that it is both accurate and efficient. Most energy is consumed by the transmission of messages as data is returned to the gateway node. The ability to construct queries using fuzzy terms will result in a different message delivery count needed to fulfill the intent of the request. Therefore, to measure efficiency, I will count the number of node transmissions or the number of messages generated by the query. To evaluate accuracy, I will use the measures of recall and precision, commonly used measures of data retrieval accuracy for database management systems.

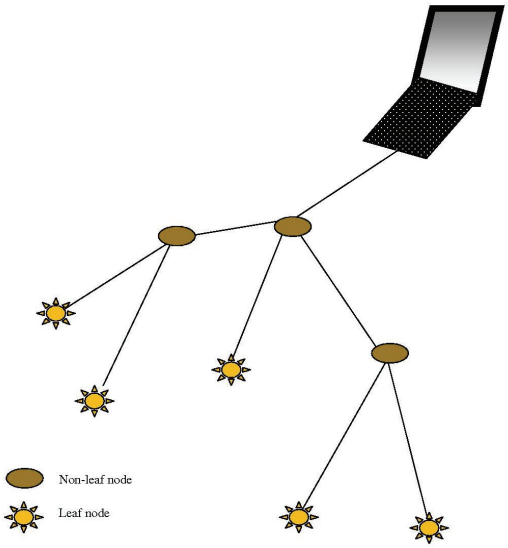


FIGURE 30: WSN network - leaf and non leaf nodes

Message count is the number of messages received at the gateway. Message count may understate the gain in efficiency, particularly in cases where aggregation techniques are used. Aggregation is when an intermediary or non-leaf node collects the data from its child nodes and combines that data with its own, if required, before resending the result message. The data from the 3 nodes is sent with only one message. If only 1 child participates in the query response, the message count remains at 1. Energy at the nonparticipating node is not spent. That is not reflected in message count. Node transmission count is the count of the number nodes participating in sending the result message. In the case where

both child nodes and the parent participate in the response, the node transmission count is 3. In the case where only one child node participates in the response, the node transmission count is 2, reflecting the efficiency gain.

Precision represents the exactness of the data returned. It measures the fraction of the results returned that are relevant (Figure 31). The formal definition for precision is shown in Equation 13.

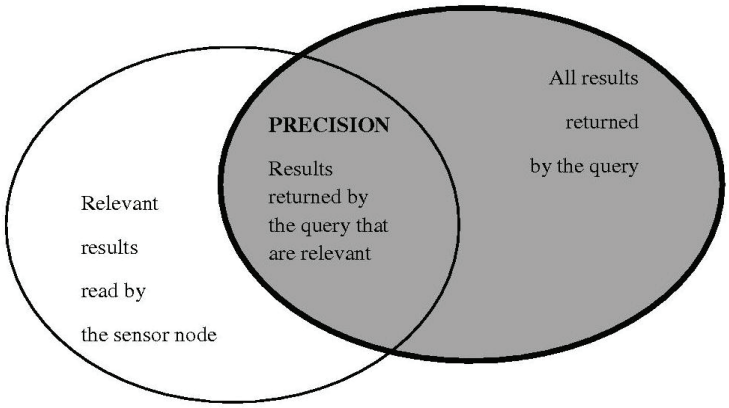


FIGURE 31: Illustrative definition of the accuracy measure: Precision

EQUATION 13: Precision definition

$$\boldsymbol{precision} = \frac{\{\boldsymbol{relevant\ results}\ \} \cap \{\boldsymbol{all\ results\ returned}\ \}}{\{\boldsymbol{all\ results\ returned}\}}$$

Recall represents the completeness of the data returned. It measures the fraction of the relevant sensor readings that are successfully retrieved (Figure 32). The formal definition for recall is shown in Equation 14.

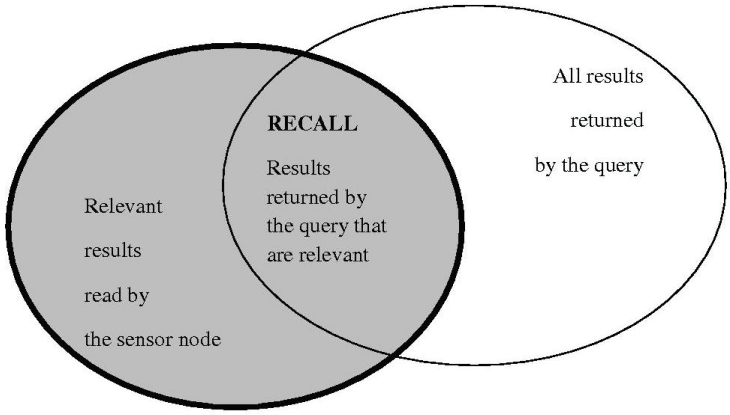


FIGURE 32: Illustrative definition of the accuracy measure: Recall

EQUATION 14: Recall Definition

$$\boldsymbol{recall} = \frac{\{\boldsymbol{relevant\ results}\} \cap \{\boldsymbol{all\ results\ returned}\}}{\{\boldsymbol{relevant\ results}\}}$$

To measure accuracy, I will use a measure of precision and recall. The evaluation criteria for recall and precision will be defined specific to each test.

.

6.3 Test 1: Constant Homogenous Environment

Environment Scenario Used: Break in a dam wall

Fuzzy Membership Function Used: S-function fuzzy membership function will be used.

Description of application problem: The city has recently rebuilt part of the dam structure across the local river. Civil engineers have determined the strength of the old part of the dam, the new part and the seam between the two. They have placed a sensor network throughout the structure. The network is to query the measured pressure and report back if the structure is 'close to' rupture. In the river bed, there are 3 beams that keep banging against the dam wall in a way that increases in strength with each pulse. Eventually, they rupture the wall. After which the banging continues and the rupture space increases. The WSN will monitor the stress level and report when the stress level is dangerously close to rupture, by sending a message with stress information.

Purpose of the test: This test examines how closely a fuzzy query performs to its classic counterpart.

Description of the test: The scenario provides a constant, homogeneous environment. Every region in the 'wall' has the same strength, initial stress level and threshold of stress level at which the region breaks. Each region is initiated with a stress value of 50,000 psi (pound-force per square inch) , and will rupture when experiencing 75,000 psi. There are 3 regions designated as stress regions representing where the 'beams' hit the wall. In these regions the stress is increased every 5 secs by 1000 psi, plus a random noise factor ranging from 10-100 psi. When

the region ruptures, overflow stress is divided and propogated to its neighbors. Once the region is identified as completely ruptured, all the stress it receives is divided and propagated to its neighbors. Both the classic query and the fuzzy query sample the pressure information every 3 seconds.

The fuzzy query uses a membership function that measures a value as it approaches a threshold. In particular it uses the S-membership function. When the region reaches 70% possibility of ruturing,i.e, a fuzzy value of 0.7, a message is sent to the gateway node. Sensors are provided the values which map the S-function parameters to the strength of the wall. The classic queries and fuzzy queries are constructed to be identical. That is, the selection criterion in the classic query (65,200 psi) was selected to exact the same minimum selection criterion of the fuzzy query. Since the requirement is for each sensor node to return notification if a danger level of stress is experienced, in-network aggregation, i.e. averaging, is not appropriate. By comparing these queries, efficiency and accuracy of the fuzzy query with respect to an equivalent classic query is examined.

The following classic and fuzzy queries are used. The following queries are sent through the sensor network to monitor the wall structure:

Classic Query: *SELECT nodeID, pressureValue*
FROM sensors
WHERE pressureValue> 65,200 psi
EPOCH 3 secs

Fuzzy Query: *SELECT nodeID, fuzzyPressureValue*
FROM sensors
WHERE fuzzyPressureValue>possibility-of-failure-value-70%
EPOCH 3 secs

The classic query retrieves all node identifiers and their pressure value when the pressure value sensed is greater than 65,200 psi. The fuzzy query retrieves the node identifier and pressure values from any sensor where the pressure value results in a fuzzy value greater than 0.7 when it is processed by the fuzzy membership function residing on that node.

Mote Deployment: Motes are placed in the environment in a variety of topological deployments of 6 and 12 motes. There are six different deployment variations. In five sets of deployments, motes are placed using a random distribution, each using a different seed for randomization. This pseudo-random generation allows the test to be repeatable for both the classic and fuzzy query conditions. The randomness of note placement may result in 2 motes residing in the same region. One deployment is organized in a uniform distribution.

Results: This following charts show the results for Test 1. Figure 33 and Figure 34 showing message count are used to evaluate the efficiency of the query. Figures 42-45 are used to evaluate the accuracy of query.

EFFICIENCY MEASURMENTS

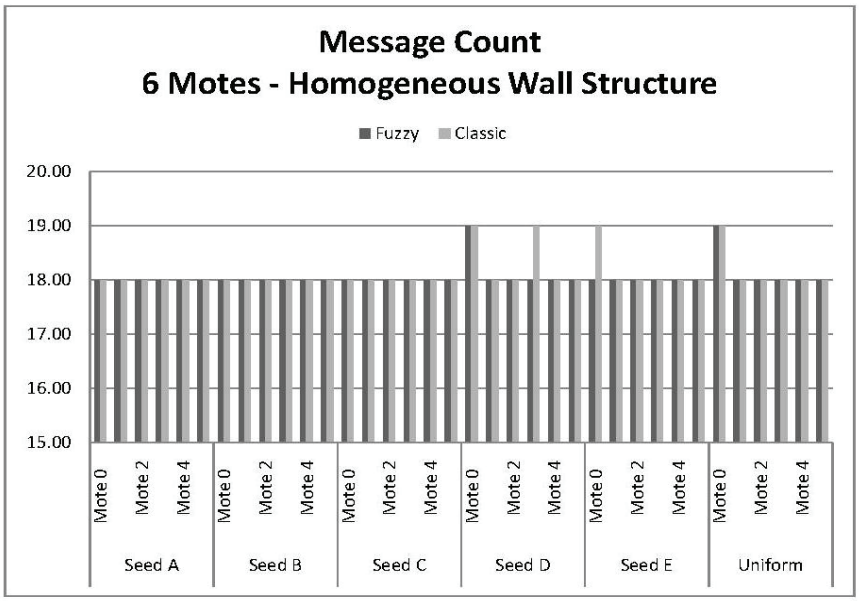


FIGURE 33: Message count for 6 motes in a homogenous structure

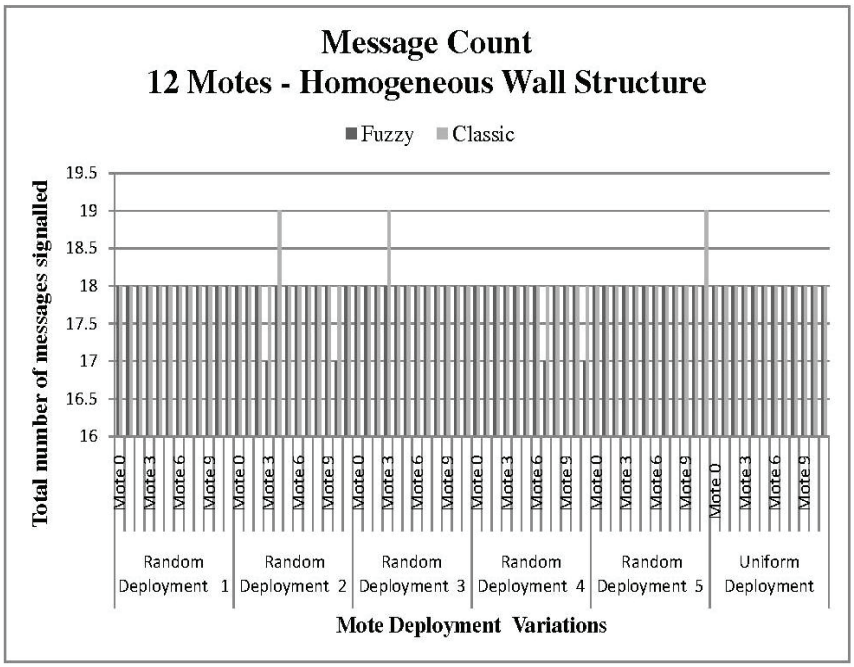


FIGURE 34: Message count for 12 motes in a homogenous structure

ACCURACY MEASURMENTS

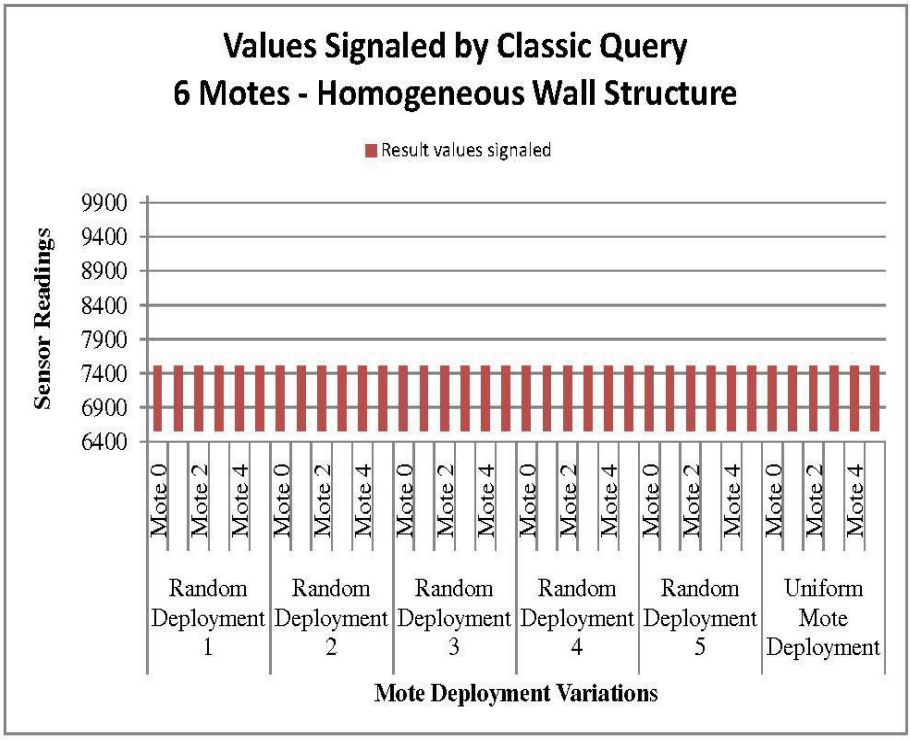


FIGURE 33: Result values for classic query over 6 motes in a homogenous structure

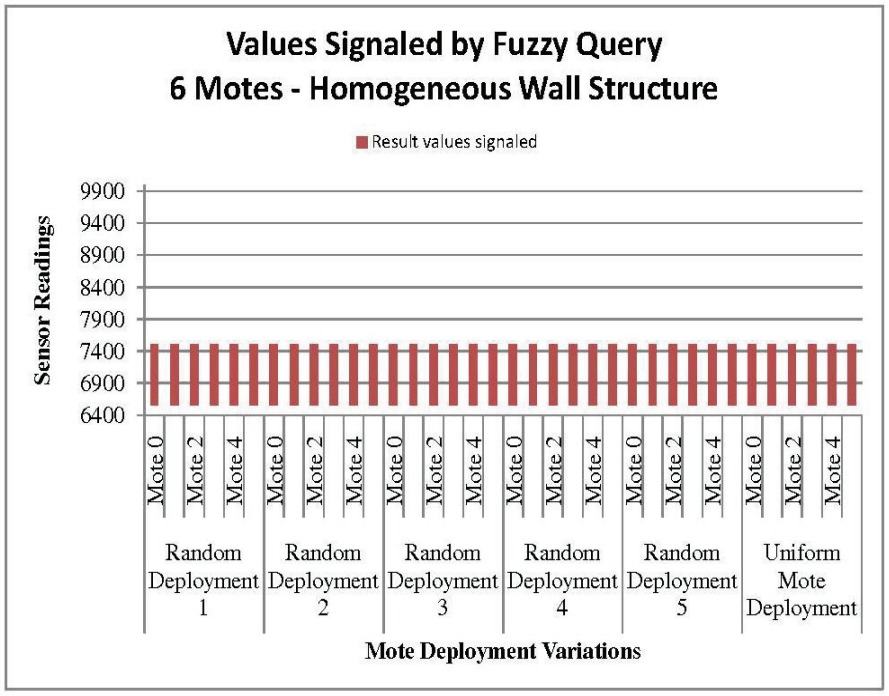


FIGURE 34: Result values for fuzzy query over 6 motes in a homogenous structure

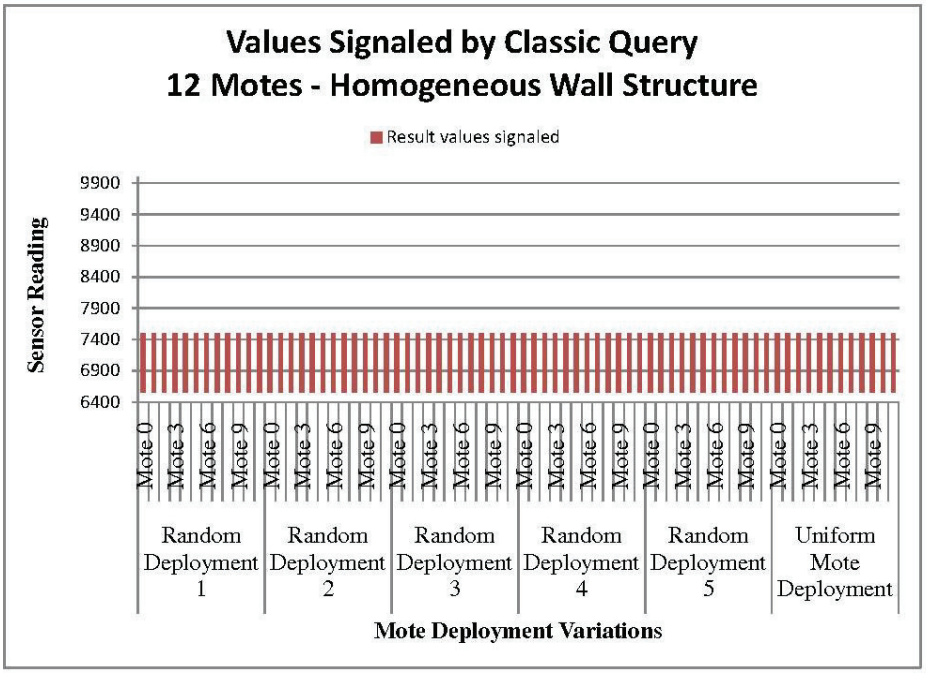


FIGURE 36: Result values for fuzzy query over 12 motes in a homogenous structure

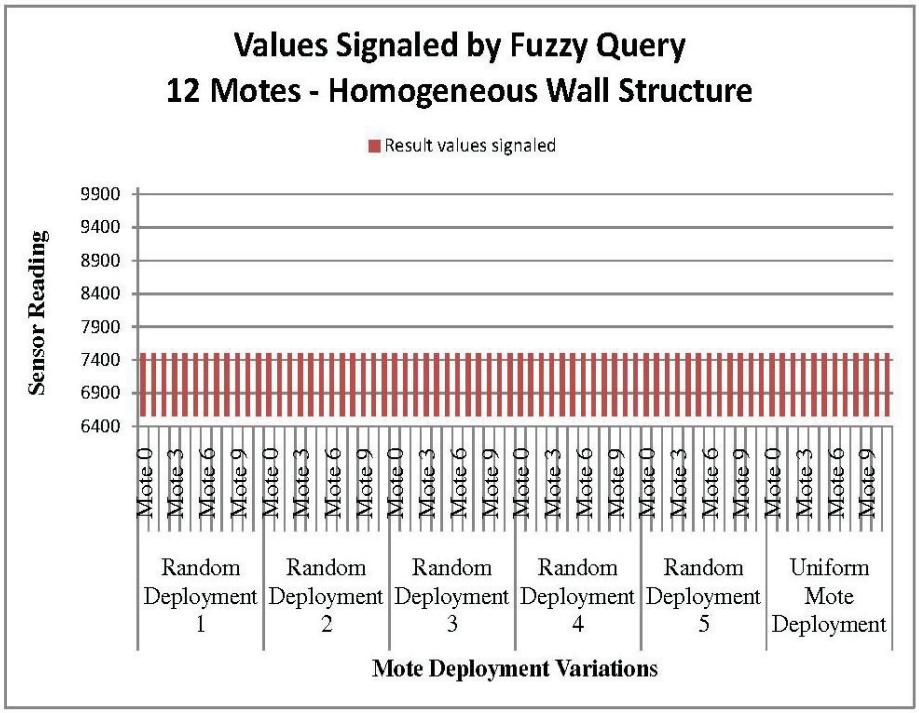


FIGURE 35: Result values for classic query over 12 motes in a homogenous structure

Discussion of results: To evaluate efficiency, I use the number of messages from the first report of a node sensing a value at the danger level, until the region ruptures. To evaluate recall accuracy, (recall represents the fraction of relevant objects that are retrieved in the answer relative to the total number of true relevant objects), I will use the sensor value triggering the query result as well as the sampling epoch at which it was returned. If the fuzzy query returns results later than a danger level is signaled by the classic query, its recall is less accurate. To evaluate precision accuracy, (precision represents the fraction of retrieve objects that are relevant) I compare the first result reporting the anomaly from both the classic and the fuzzy query. I will use the sensor value triggering the query result as well as the sampling period at which it was returned as criteria. If the fuzzy query returns results before a danger level exists, it is less precise.

The efficiency of the queries is shown in Figure 39 and Figure 40. Figure 39 shows the message count for both fuzzy and classic queries in a 6 mote network. Figure 40, shows the message count for both fuzzy and classic queries in a 12 mote network. The average message count for classic queries, in both test suites, is 18 messages per mote. The average message count for fuzzy queries is also 18 messages per mote. Therefore, in equivalent tests, the efficiency of the fuzzy query is equal to that of the classic query. To compare the results returned from the fuzzy query, I determine the number of responses generated by counting the result messages for each region. In this test, 1872 results were returned by the classic query; 1872 results were returned by the fuzzy query.

The values returned by both the classic and fuzzy query start at 65,500psi.

Readings end when the region ruptured: 75,000 psi The set of values returned from the fuzzy query [65500,…75000] are a subset of the set of values returned by the classic query (relevent results). The intersection of the relevant (classic query) results and the results retrieved by the fuzzy query is exactly the results of the fuzzy query.

Precision measures the fraction of the results returned that are relevant. To calculate precision, I use the precision formula defined in Equation 13 and repeated here for the reader:

$$precision = \frac{\{\, relevant\ results\ \} \cap \{all\ results\ returned\ \}}{\{all\ results\ returned\}}$$

Applying the results from the tests suites, the precision of the fuzzy is determined as:

EQUATION 15: Precision of fuzzy query results in Test 2

$$precision = \frac{1872}{1872} = 1$$

Therefore, the precision of the fuzzy query is equivalent to that of the classic query. All the results returned from the fuzzy query are relevant.

Recall measures the fraction of the relevant sensor readings that are successfully retrieved. To caluclate precision, I use the recall formula defined in Equation 14 and repeated here for the reader:

$$recall\ = \frac{\{relevant\ results\ \} \cap \{all\ results\ returned\}}{\{relevant\ results\}}$$

Remembering that the union of the set of relevant results and the set of all results returned is, in fact, the set of values from the classic query I apply my findings to

the recall formula:

EQUATION 16: Recall of fuzzy query results in test 2

$$recall = \frac{1872}{1872} = 1$$

Therefore, the fuzzy query only has the same recall as compared to the classic query.

This demonstrates that a fuzzy query can be defined without loss of efficiency and without concern of bad or inaccurate results. In a static homogeneous environment, a fuzzy query is as precise as a comparable classic query (100% precise) with the same recall (100% recall). Building on this, the next test compares a fuzzy and classic query in varied environment, that is, a macro-environment (the dam wall) with three separate mini-environments (three sections with differing strength values).

6.4. Test 2: Constant Heterogeneous Environment

Environment Scenario Used: Break in a dam wall

Fuzzy Membership Function Used: S-function fuzzy membership function

Description of application problem: The city has recently rebuilt part of the dam structure across the local river. Civil engineers have determined the strength of the old part of the dam, the new part and the seam between the two. They have placed a sensor network throughout the structure. The network is to query the measured pressure and report back if the structure is 'close to' rupture. In the river bed, there are 3 beams that keep banging against the dam wall in a way that increases in strength with each pulse. Eventually, they rupture the wall. After which the banging continues and the rupture space increases. The WSN will monitor the stress level and report when the stress level is dangerously close to rupture, by sending a message with stress information.

Purpose of the test: This tests compares fuzzy query using a single fuzzy membership function, to a classic query in a constant, heterogenous environment

Description of the test: The 'wall' is divided into three sections

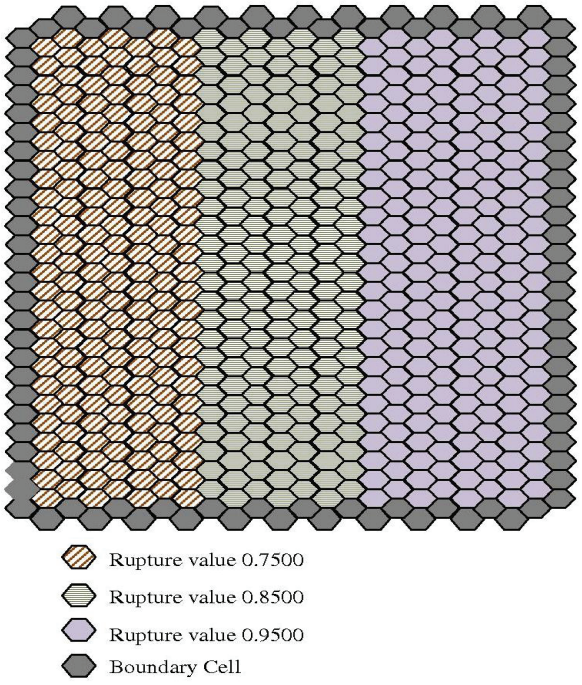


FIGURE 37 : Heterogeneous Environment

(Figure 46). All regions in the wall are initiated with a stress value of 50,000 psi. Each section has a different rupture stress value. Each region in weakest region will rupture when it experiences 75,000 psi. Each region in the mid-strength region will rupture when it experiences 85,000 psi. Each region in the strongest section will rupture when it experiences 95,000 psi. Again, there are 3 regions designated as stress regions representing where the 'beams' hit the wall. In these regions the stress is increased every 5 secs by 1000 psi, plus a random noise factor ranging from 10-100 psi. When the region ruptures, overflow stress is divided and propagated to its neighbors. Once the region is identified as completely ruptured, all the stress it receives is divided and propagated to its neighbors. Both the classic query and the fuzzy query sample the pressure information every 3 seconds.

As in Test 1, the classic query uses 62,500 psi as the selection criteria.. The fuzzy query measures the sensor value as it approaches the stress threshold usiing the membership S-function. The S-function parameters are provided to each sensor mote, mapping the strength of the region in which they are deployed. When the region reaches the fuzzy value 0.7, has a 70% possibility of ruturing, a message is sent to the gateway node.

The following classic and fuzzy queries are used. The following queries are sent through the sensor network to monitor the wall structure:

Classic Query: *SELECT nodeID, pressureValue*
FROM sensors
WHERE pressureValue> 65,200 psi
EPOCH 3 secs

Fuzzy Query: *SELECT nodeID, fuzzyPressureValue*
FROM sensors

WHERE fuzzyPressureValue>possibility-of-failure-value-70%

EPOCH 3 secs

The classic query retrieves all node identifiers and their pressure value when the pressure value sensed is greater than 65,200 psi. The fuzzy query retrieves the node identifier and pressure values from any sensor where the pressure value results in a fuzzy value greater than 0.7 when it is processed by the fuzzy membership function residing on that node.

Mote Deployment: Motes are placed in the environment in a variety of topological deployments of 6 motes (sparse deployment), 12 motes, 24 motes, and 50 motes (dense deployment). There are six different deployment variations. In five sets of deployments, motes are placed using a random distribution, each using a different seed for randomization. This pseudo-random generation allows the test to be repeatable for both the classic and fuzzy query conditions. The randomness of note placement may result in 2 motes residing in the same region. One deployment is organized in a uniform distribution.

Results: This following charts show the results for Test 2. Figure 46 and Figure 48 showing message count are used to evaluate the efficiency of the query. Figures 48-51 are used to evaluate the accuracy of query.

EFFICIENCY MEASURMENTS

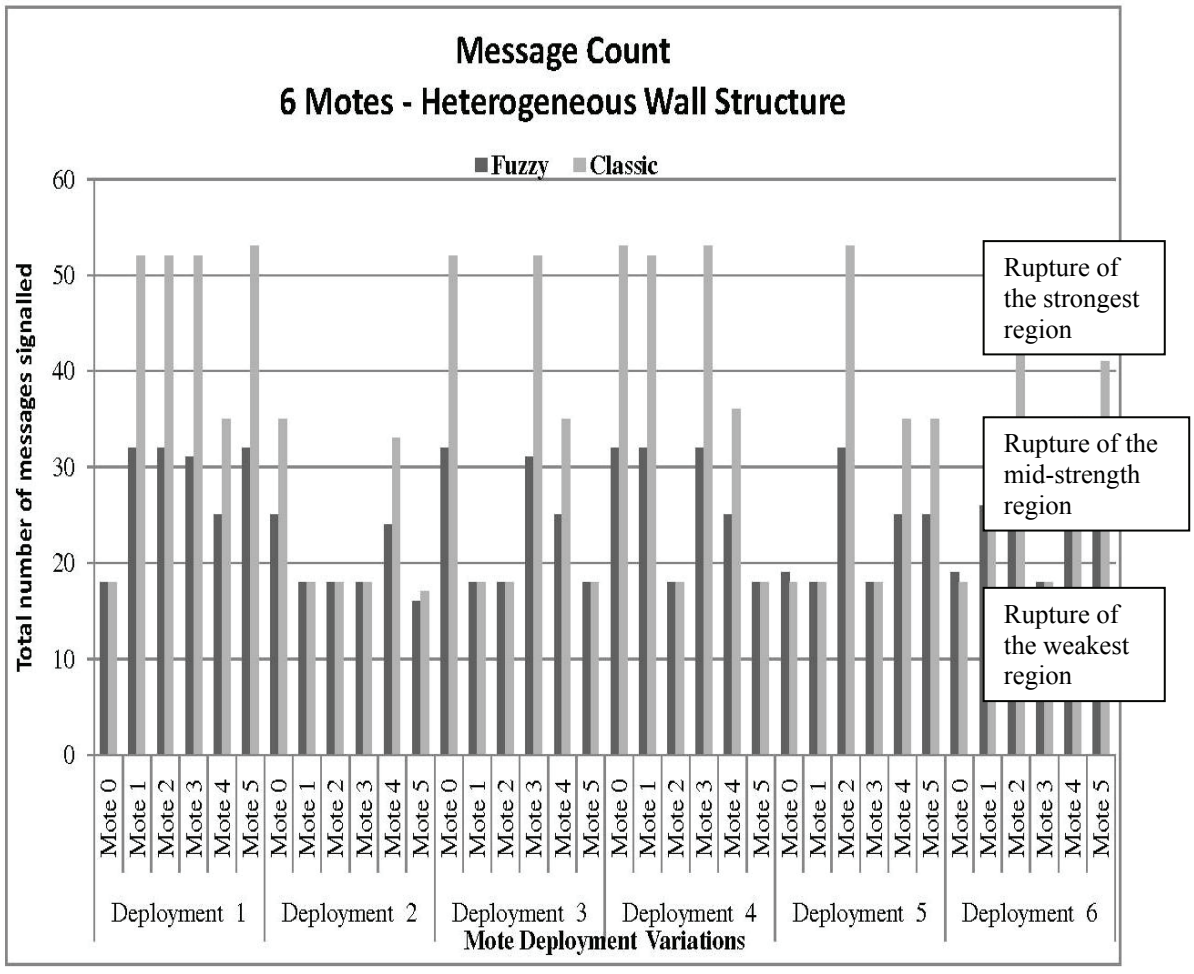


FIGURE 38: Message count for 6 motes in a heterogeneous structure

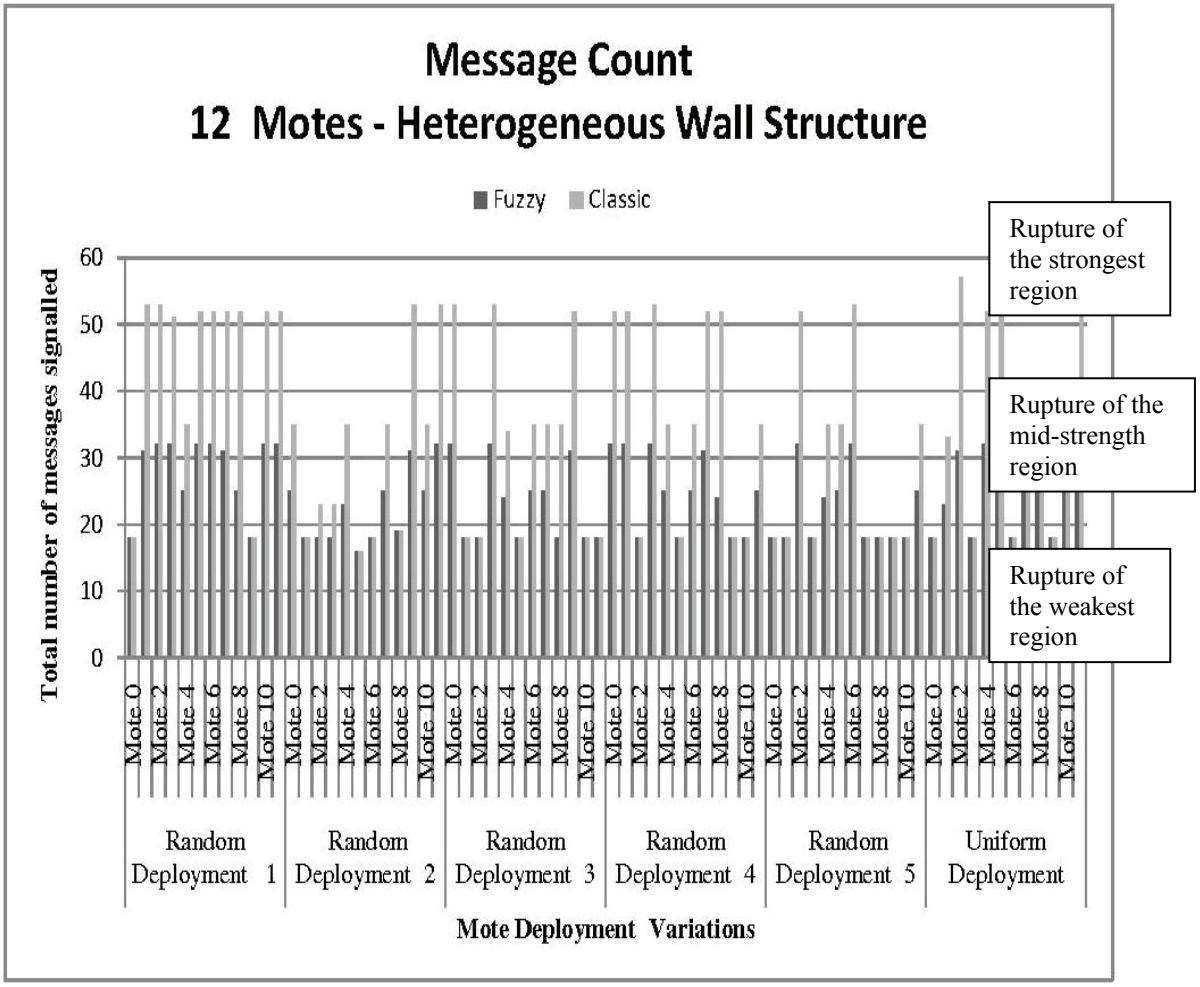


FIGURE 39: Message count for 12 motes in a heterogeneous structure

ACCURACY MEASURMENTS

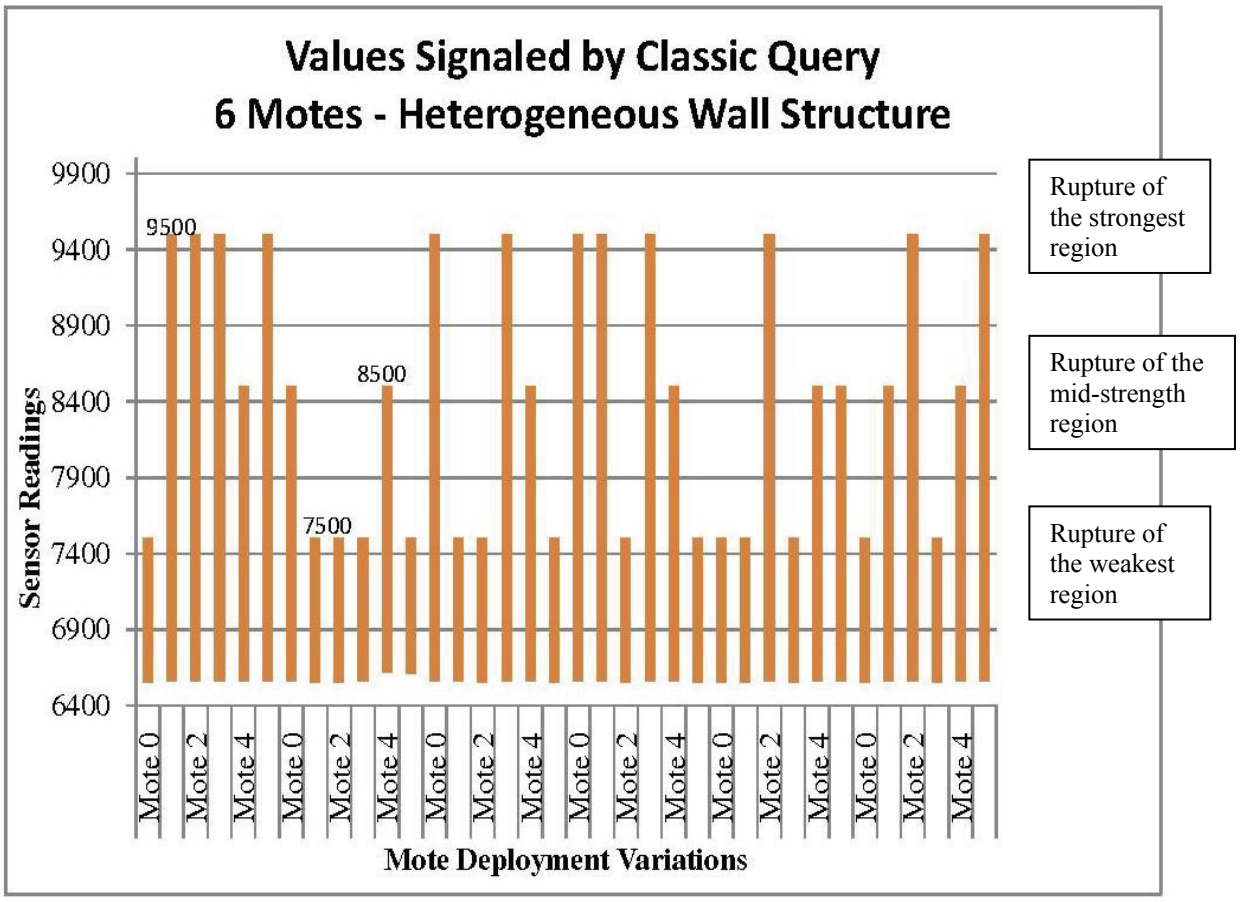


FIGURE 40: Results for classic query over 6 motes in a heterogeneous structure

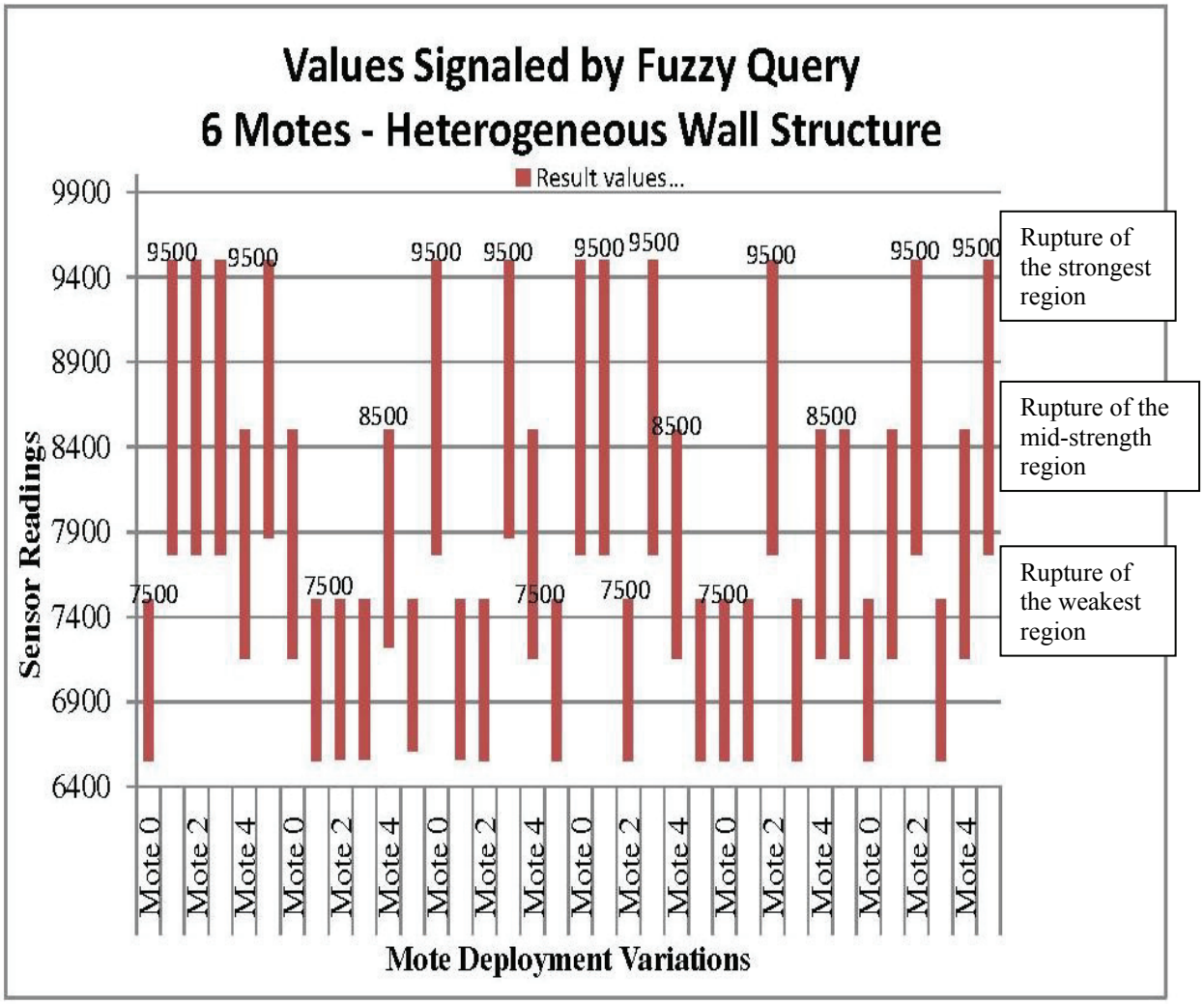


FIGURE 41: Results for fuzzy query over 6 motes in a heterogeneous structure

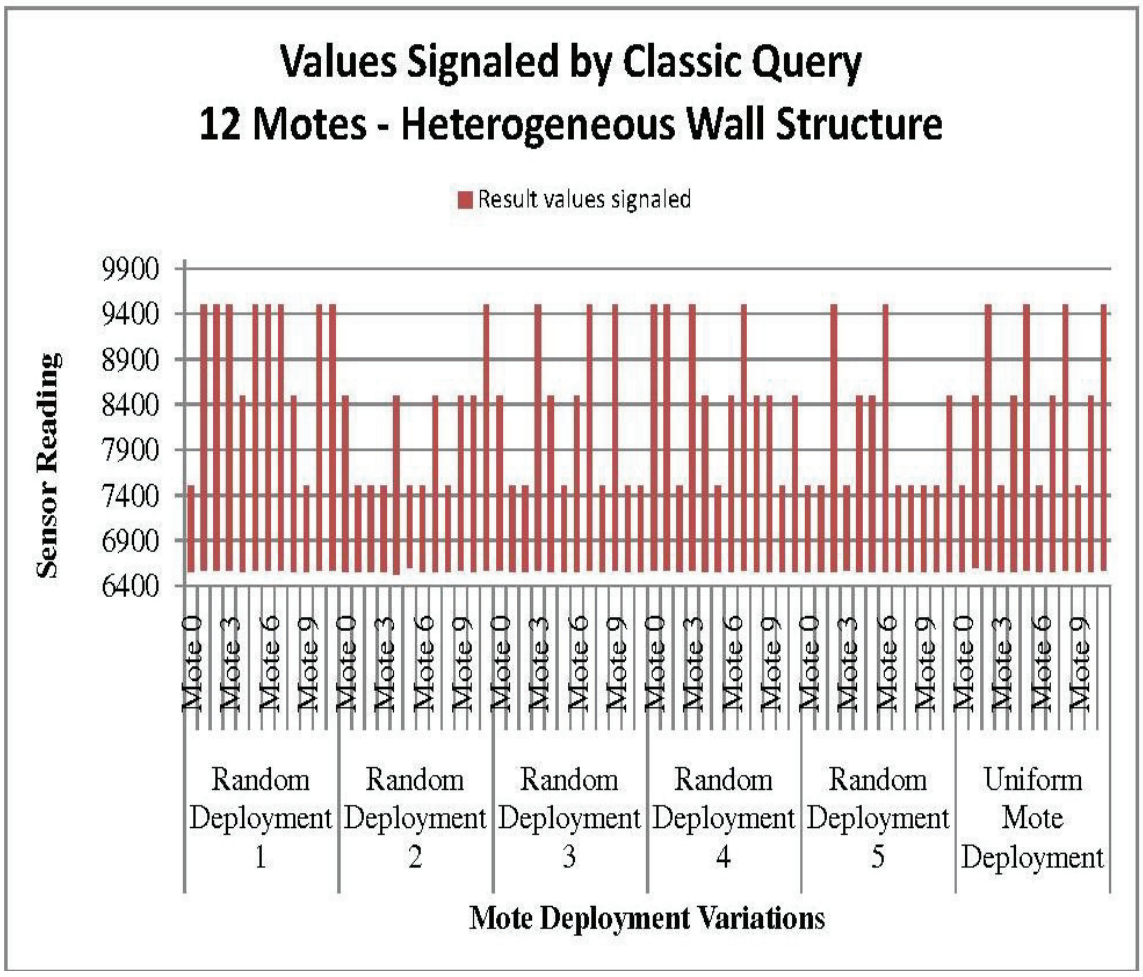


FIGURE 42: Results for classic query over 12 motes in a heterogeneous structure

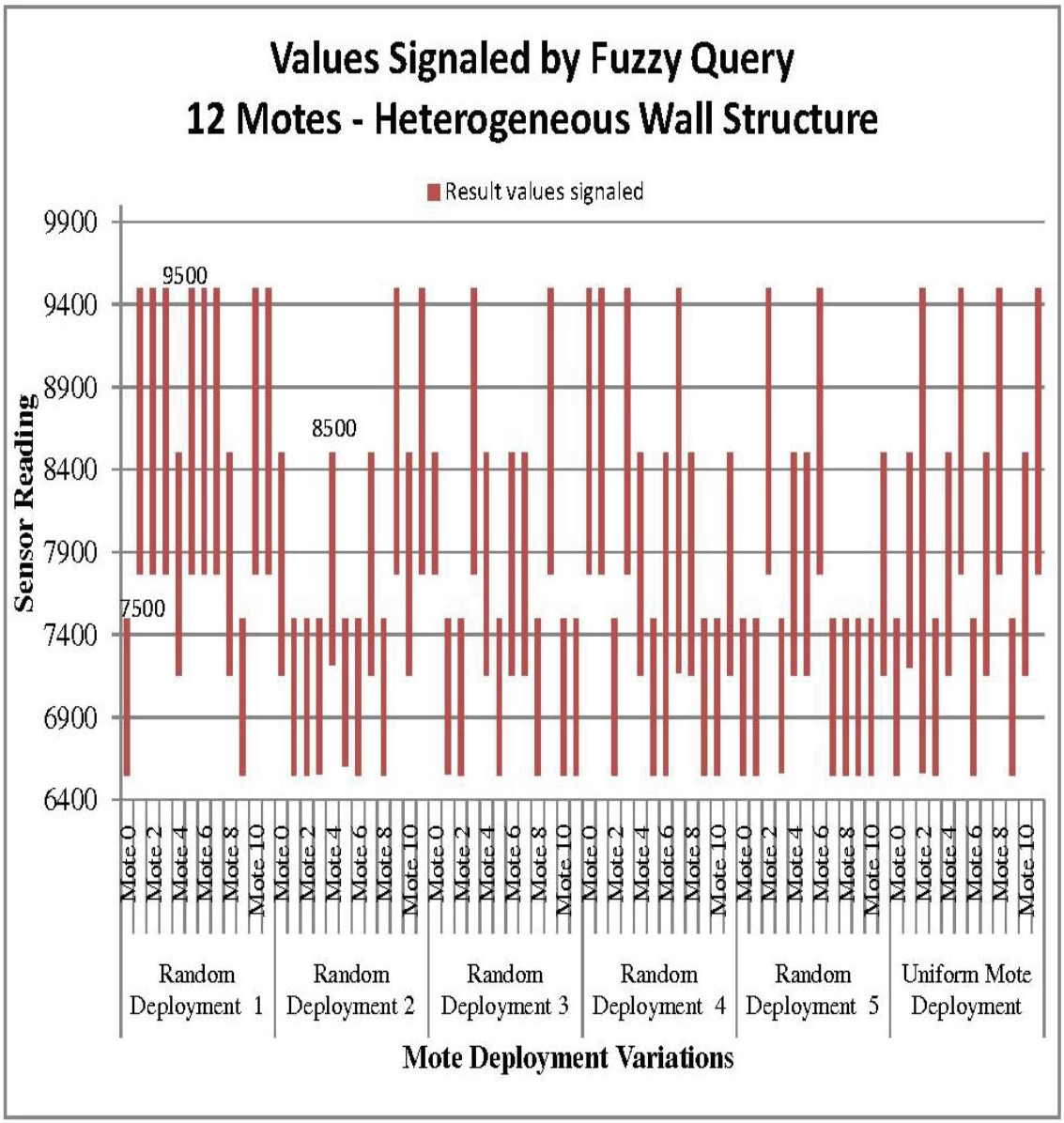


FIGURE 43: Results for fuzzy query over 12 motes in a heterogeneous structure

Discussion of results: The efficiency of the queries is shown in Figure 46, which shows the message count for both fuzzy and classic queries in a 6 mote network, and in Figure 47, which shows the message count for both fuzzy and classic queries in a 12 mote network. Similar results were obtained for networks of 24 and 50 motes. The average number of message signalled over all networks are listed for each query type is given in Table 3. These are then shown in the graph (Figure 46).

TABLE 3: Average message count for Test 2

	Classic Query	**Fuzzy Query**
Strongest Region	51	32
Mid-strength Region	36	25
Weakest Region	18	18
Average over all Regions	33	24

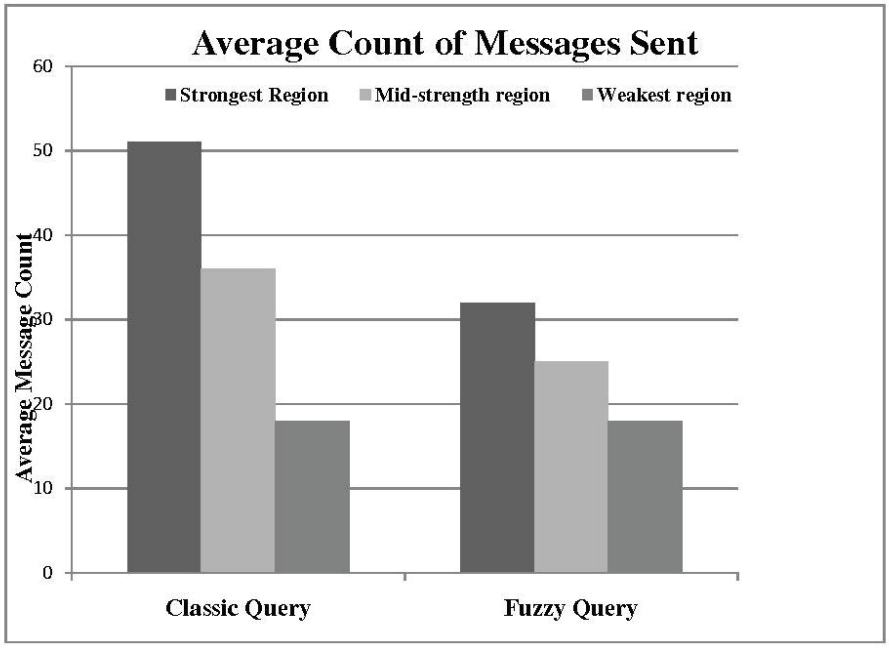


FIGURE 44: Average message count for Test 2

It can be seen that the fuzzy query return fewer messages as compared to the classic query. In particular, the fuzzy query retuned 40% fewer message in the strongest region and 32% fewer in the mid-strength section. To the classic query. While in the weakest section (which replicates the conditions in Variation 1), the message count is the same for both queries. Overall the fuzzy query returned 27% fewer message than the classic query.

To examine the accuracy of the fuzzy query with respect to the classic query, I define the results returned by the classic query to be the complete set of relevant data. The retrieved results from the fuzzy query are compared to those relevant results to ascertain its relative accuracy.

The values returned by the classic query start at 65,000psi. Readings end when the region ruptured: 75,000 psi for the weakest region, 85,000 psi for the mid-strength region, and 95,000 psi for the strongest region. The set of values returned from the fuzzy query [6550,…9500] are a subset of the set of values returned by the classic query (relevent results). The intersection of the relevant (classic query) results and the results retrieved by the fuzzy query is exactly the results of the fuzzy query.

TABLE 4: Relevant response count classic queries for Test 2

Classic Query	**Minimum value returned**	**Maximum value returned**	**Number of messages returning these results**
6 Motes			
Strongest Region	6550	9500	617
Mid-Strength Region	6550	8500	315
Weakest Region	6550	7500	269
12 Motes			
Strongest Region	6550	9500	2439
Mid-Strength Region	6550	8500	732
Weakest Region	6550	7500	1186
24 Motes			
Strongest Region	6550	9500	2705
Mid-Strength Region	6550	8500	1273
Weakest Region	6550	7500	1000
50 Motes			
Strongest Region	6550	9500	4098
Mid-Strength Region	6550	8500	2758
Weakest Region	6550	7500	1640
Total number of relevant results sent			**17,846**

TABLE 5: Response count from fuzzy query for Test 2

Fuzzy Query	**Minimum Value Returned**	**Maximum Value Returnd**	**Number of messages Returning these results**
6 Motes			
Strongest Region	7764	9500	270
Mid-Strength Region	7157	8500	224
Weakest Region	6550	7500	381
12 Motes			
Strongest Region	7764	9500	730
Mid-Strength Region	7157	8500	493
Weakest Region	6550	7500	521
24 Motes			
Strongest Region	7764	9500	1550
Mid-Strength Region	7157	8500	903
Weakest Region	6550	7500	1065
50 Motes			
Strongest Region	7764	9500	2428
Mid-Strength Region	7157	8500	2013
Weakest Region	6550	7500	1740
Total number of results sent returned by the fuzzy query			**12,318**

To compare the results returned from the fuzzy query, I determine the number of responses generated by counting the result messages for each region (Table 5).

Precision measures the fraction of the results returned that are relevant. To calculate precision, I use the precision formula defined in Equation 13 and repeated here for the reader:

$$precision = \frac{\{\, relevant\ results\ \} \cap \{all\ results\ returned\ \}}{\{all\ results\ returned\}}$$

Applying the results from the tests suites, the precision of the fuzzy is determined as:

EQUATION 17: Precision of fuzzy query results in Test 2

$$precision = \frac{12{,}318}{12{,}318} = 1$$

Therefore, the precision of the fuzzy query is equivalent to that of the classic query. All the results returned from the fuzzy query are relevant.

Recall measures the fraction of the relevant sensor readings that are successfully retrieved. To caluclate precision, I use the recall formula defined in Equation 16 and repeated here for the reader:

$$recall\ = \frac{\{relevant\ results\ \} \cap \{all\ results\ returned\}}{\{relevant\ results\}}$$

Remembering that the union of the set of relevant results and the set of all resutls returned is, in fact, the set of values from the classic query I apply my findings to the recall formula:

EQUATION 18: Recall of fuzzy query results in Test 2

$$recall = \frac{12{,}318}{17{,}848} = .69$$

Therefore, the fuzzy query only has a 69% recall as compared to the classic query.

This variation shows that fuzzy queries offer improved efficiency, but sacrifice some accuracy over the classic query. However, it can be argued that the fuzzy query, by allowing the user to more exactly pose the question, the fuzzy query results more closely match the user's intent. If the user wants to be notified only when the sensor mote detects a stress level indicating a 70% possibility of rupture in a region where the rupture threshold is determined to be 85,000 psi, then the results returned by the classic query lower than 71,570 psi are extraneous.

6.5. Test 3: Dynamic Environment

Environment Scenario Used: Break in a dam wall

Fuzzy Membership Function Used: PI-function fuzzy membership function.

Description of the application problem: Gas is flowing through an enclosed space. It enters through a vent in the left wall and exits on through a vent in the right wall. The application desires the space to have a controlled level of saturation of this gas, however flow must be constant. The vents on either side may be opened or closed. The experiment starts with the exit vent closed. When the saturation level reaches a certain point, the vent in the right wall must open and when the saturation level is below a certain point, the vent in the right wall must close. Inversely, the vent on the left wall must open and close to control the increase the level of gas particles. To insure constant flow, one vent must be open at all times.

Purpose of the test: This test compares the effectiveness of fuzzy versus classic queries with respect to environment conditions.

Description of the test: The test implementation based the behavior defined in [101]. In this scenario, all regions receive the same behavior function. A row of regions is set as a 'wall' by indicating these are boundary regions. Three adjacent regions of the wall are selected as a 'hole' by defining their behavior as non-boundary regions. Each non-boundary region will be initiated with a pseudo-random particle count. Each particle is given a directional vector value, selected (pseudo)randomly, from 45°, 90, ° 135°, 225°, 270°, and 315° degrees.

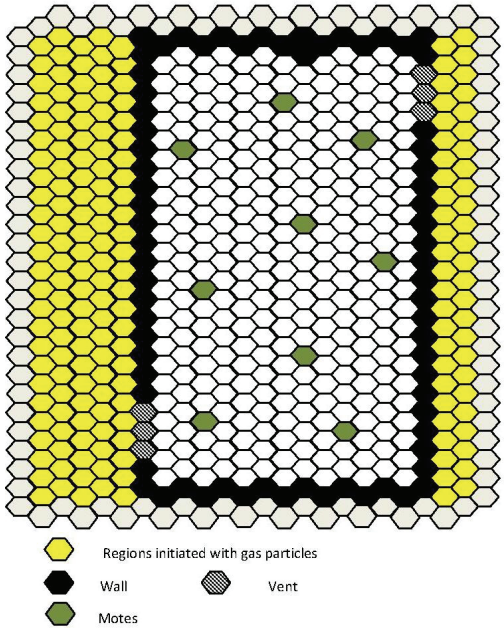


FIGURE 45: Grid for saturation scenario

The regions in the environment, both in the chamber and outside the chamber are initiated with 16, to 18 particles, each given a random direction (Figure 53).. The parameters for both queries are required to maintain saturation level between 12 and 14 average particles. For the classic query, if the average reading is lower than 12 or greater than 14, then the direction of the flow is reversed by either opening the entrance vent and closing the exit vent or closing the entrance vent and opening the exit. For the fuzzy query if 7 sensors or more report a fuzzy value less than 90, which is defined as *acceptableSaturation* then the vents must be adjusted. Since the fuzzy value gives no indication of which direction to adjust the vents, the fuzzy query also returns the average of these sensors. If the average reading is lower than 12 or greater than 14, then the direction of the flow is reversed by either opening the entrance vent and closing the exit vent or closing the entrance vent and opening the exit. The fuzzy membership function used is the PI function. Collisions of the particles in the region are resolved and directional vectors are reset. The collision rules for this test are illustrated in Figure 29. As a result of collisions, particles are sent to a target region by signaling environment events to that region. Any particle hitting a boundary region will be directed in is opposite direction and will remain in

the sending region (a bounce back). Any particle coming into the hole region will be directed to the an adjacent (non-wall) region beyond the wall. The direction will be randomly selected from 45°, sending the particle to the NE neighboring region, 315°, sending the particle to the SE near neighboring region, and 0°, sending the particle to the East distant neighbor. This causes the particle in the hole to move only in a NE, E, or SE direction. In order to maintain continuing movement, the environment grid will be built with cyclic boundaries. However the monitored space will be enclosed boundary regions.

In this test, the simulated WSN must communicate with the environment scenario. The gateway node will signal an environment event to trigger the environment scenario application to open or close the vents. The gateway node receives the query results from each sensor. Keeping a log of the most current result from each sensor, the gateway node determines whether a vent should be opened or closed. The vents are open or closed by signaling an environment event. That event is handled by the environment model. The enivornment model ensures that one and only one vent is always open. To better evaluate the accuracy of the results, the environment model will periodically report the average region saturation of the entire area.

Queries will be sent to the WSN to request repeated monitoring of the saturation level. Using data returned from the nodes, the gateway node will signal the environment to open or close the vents. They, of course, will know only the particle saturation level in the region in which they reside as the gas particles flow around the chamber. The classic query will collect the average value of all readings

from the sensors deployed. The fuzzy query will select readings only from sensors which read a saturation level greater or less than the desired level. Based on the readings, the WSN signals to the environment whether to open the vent allowing particles into the chamber while closing the vent that allows particles out, or whether to open the vent allowing particles out of the chamber while closing the vent that allows particles to escape. I compare the overall average of the particles in the chamber with desired saturation level. I also run a control test, in which no sensor nodes are used. In this case, the vents are controlled by the total number of particles in the region.

The following queries are not standard SQL, but are used to illustrate how each query functions at both the node and at the gateway. In WSN, the averaging aggregation technique results in a sum of values and a count of participating nodes sent to the gateway. The gateway node, then calculates the average.

Classic Query:

```
SELECT nodeID, AVG(particleConcentration ) AS saturation
FROM sensors
EPOCH 3 secs
CASE saturation
        WHEN saturation < 12 THEN
                signal_open_flow_IN_vent
    WHEN saturation > 14 THEN
                signal_open_flow_OUT_vent
END
```

Fuzzy Query

```
SELECT nodeID, AVG(particleConcentration ) AS saturation
 FROM sensors
WHERE fuzzyGasConcentration >acceptableSaturation
EPOCH 3 secs
CASE saturation
        WHEN (saturation < 12)THEN
                signal_open_flow_IN_vent
      WHEN  (saturation > 14) THEN
                signal_open_flow_OUT_vent
END
```

Mote Deployment: Ten motes will be deployed in the area into which the particles will spread. The deployments will consist of one uniform distribution and three distributions randomly positioned. The random spread was configured using five different seeds for a random generator. This pseudo-random generation allows the scenarios to be repeatable for both the classic and fuzzy query conditions.

Results: This following charts show the results for Test 3. Figure 48 shows node transmission count are used to evaluate the efficiency of the query. The count started at the epoch that resulted in the first change of vent open/close status for each test. Figures 49, showing the average particle count for each region in the chamber is used to evaluate the accuracy of query.

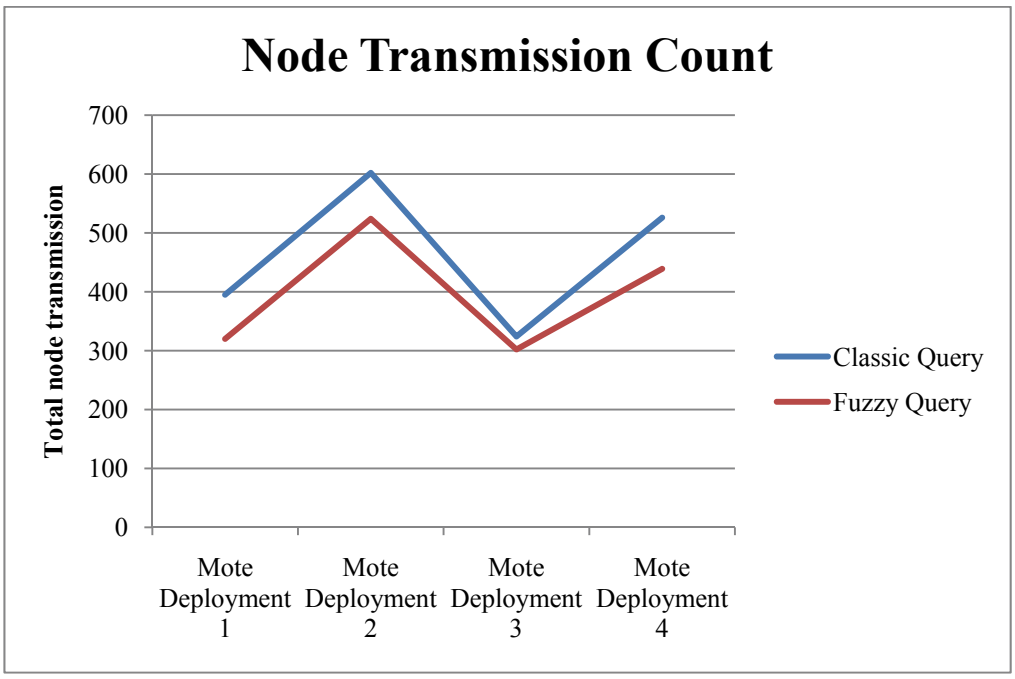


FIGURE 46: Test 3 – Node transmission Count

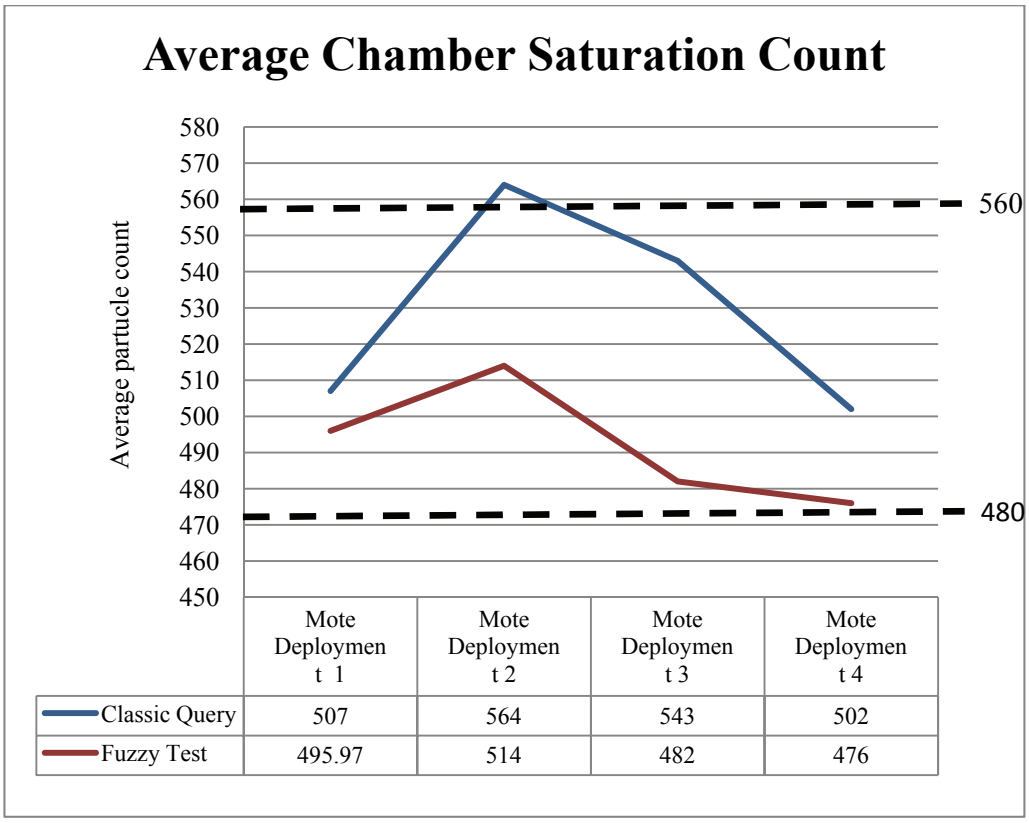


FIGURE 47: Test 3 – Average particle count in chamber

Discussion of results: In this test, efficiency is measured by counting node transmissions. The rsult message sent includes the node count. A comparision is made between the fuzzy and classic queries. Because the queries start before all the particles are distributed, it is difficult to derive constant starting epoch. At the start of the test, both vents are closed and the chamber is above saturation level. The comparision begins, for each query, at the epoch which first initates a change in the vent open/close status. The comparision ends for each test after an equal number of epochs executed. Since the fuzzy query returns reports only the nodes below the *acceptableSaturation* value (fuzzy value 0.90), it is expected that some leaf nodes may not transmit their results, if their reading falls within a fuzzy value of [0.9, 1.0]. Intermediary nodes, are required to transmit child data, regardless of their own reading. The classic query must report readings from all sensor nodes to determine the average. Aggregation is used to limit the number of node transmission for each query.

To examine efficiency, I look the number of node transmission counts sent during the query. Since the test begins with both gates closed with the chamber at a higher than allowed saturation level. I began the count at the epoch when the query first opens one of the gates. They each opened the exit gate first. The node transmission count was taken for equal number of return results. The first return result for each query was that which first opened a vent. Figure 54 shows the node transmission counts for the tests. Since the fuzzy query does not require transmission from every sensor node, instead using the fuzzy value to determine if it needs to participate, and the classic query requires the value from each sensor node

to calculate the average, the fuzzy query shows a slight improvement in energy savings through fewer node transmissions.

To examine the accuracy, I refer to the problem description: The application desires the chamber to have a controlled level of saturation of this gas, between 480 and 560 particles. The environment scenario reports the total saturation level of the chamber every three seconds. That is used to determine the saturation level for each test. The average chamber saturation levels for the tests are shown in Figure 49. Both fuzzy and classic queries kept the saturation levels in the range required. The average saturation level for classic queries is a bit higher than that for fuzzy queries. Classic queries included all outlier values in its calculation. I had observed some regions having as many as 150+ particles in one epoch and some having as few as 5 in one epoch. These values would skew the average. Fuzzy queries counted the outliers bit were not influenced by their extreme values.

These tests suggest that in a dynamic environment fuzzy queries can offer improved efficiency, while not significantly, affecting accuracy.

CHAPTER 7: DISCUSSION

In Test 1, the environment is constant and the strength value of the wall is constant through the structure. The classic query tests for a dangerous pressure value, determined by a domain expert, that indicates a possible rupture area. If the current reading is greater than the dangerous pressure value, a result message is sent though the network reporting possible rupture. The fuzzy query tests for the same possible rupture area. However, it used a fuzzy membership function, provided by the domain expert, to determine if the result should be reported. The results of Test 1 show that the classic query and fuzzy query performed equivalently. I had expected these results since I had built the queries to be functionally identical and the environment was constant. The test demonstrates that a fuzzy query can be defined to perform without loss of efficiencyand without concern of bad or inaccurate results. In a static homogeneous environment, a fuzzy query is as precise as a comparable classic query (100% precise) with the same recall (100% recall). This test validates that using a fuzzy query in a WSN is reasonable and that we can expet its results to be relevant.

Building on Test 1, I investigated how a varied environment affects the comparison of the two query types. In Test 2, the environment varies in it is divided into 3 areas each with different strengths. Both the classic query and the fuzzy query remain the same. The classic query uses the same dangerous pressure value used in

Test 1. Since it is important that any danger is reported, the classic query uses the pressure value that indicates a dangerous level in the weakest region. The fuzzy query uses the same membership function, used in Test 1, to determine if results should be reported. For the fuzzy query test, the parameters used in the fuzzy membership function are sent to the sensor node once the nodes are positioned. These parameter values reflect the strength of the area in which the node is deployed. The results of this test show that fuzzy queries offer improved efficiency, but sacrifice some accuracy over the classic query. The increase in efficiency is achieved because the fuzzy query can use the parameters of the fuzzy membership functions as apriori knowledge about its region to limit the number of results it is required to report. The trade-off of accuracy is expected because the fuzzy queries, themselves, are imprecise. The selection criteria used, the fuzzy value returned from the fuzzy membership value is less precise than the predetermined stress value used in the classic query. It can be argued that the fuzzy query, by allowing the user to more semantically pose the question, provides results that closely match the user's intent. The classic query asks "Report when the stress value read, is greater than this specific value". The fuzzy query asks "Report when the stress value read is close to rupture" where the fuzzy value selection criteria defines what 'close to" means. Without argument, Test 3 demonstrates that a fuzzy query can be defined to perform more efficiently with a trade-off in accuracy, in a constant heterogeneous environment.

Finally, I investigated how a fuzzy query compares to a classic query in a dynamic, constantly changing environment. Test 3 has a constanly moving flow of

gas particles throuh a chamber. The WSN measures the concentration level at each node and works with the environment scenario to manage the entrance and exit vents to keep the concentration of the gas particles at a certain level. Throughout the test the entrance and exit vents can not be closed at the same time. Since fuzzy queries report only values that are ‘out of bounds’ and classic queries report reading to calculate average, it is expected that fuzzy queries will be more efficient. This was shown in test results. The test also showed that fuzzy queries did not provide significantly less accurate results.

CHAPTER 8: CONCLUSIONS AND FUTURE WORK

The utility of a sensor network is primarily in its ability to gather and communicate collected data. However, it is well know that communicating data over the wireless medium, even at short ranges, consumes far more energy than processing that data on the same sensor node. Minimizing the amount of data sent through the network, yet still providing the gateway node the essential information, can significantly prolong the lifetime of a working sensor node. Research is being done in networking and topology management to address the communication efficiency. In-network data aggregation techniques have been suggested to combine increase efficiency by decreasing the number of message required to answer a particular query question. Aggregation techniques are used to combine data from incoming routes to minimize the number of messages sent enroute. Regression techniques are used to extrapolate data from previous stored readings on the gateway in order to limit the requests of data from the network. However, more attention is needed on using in-network WSN data management techniques energy efficiency.

In my data management survey, I showed that each classification presented different approaches to data management in sensor networks with respect to the information queried by the application solutions. These solutions require the context of the information queried to be expressed using exact parameters.

However, in sensor networks, which collect real world data, the data is often imprecise. It is hard to exact a clear cut-off value to gather all the relevant data when querying data from a sensor network.

Fuzzy database research is an area that can be beneficial in wireless sensor networks, specifically if used to model and reason about sensor data. In many respects, fuzzy representation of sensor data depicts the physical world more realistically than crisp numbers, taking into account all phenomena in the physical universe have a degree of inherent uncertainty. To implement in-network processing of fuzzy queries, I selected membership functions that were appropriate for WSN. These provide useful functions for WSN, monitoring for a threshold conditions and for conditions around an interval, and these functions should small concise storage requirements. I built these membership functions, adding them to the SwissQM query manager. I added a new instruction 'get_fuzzyvalue' that indexes into a list of membership functions implemented. I added persistent storage in SwissQM. To send new persistent storage to the sensor node, I added a new message type and a new code section to SwissQM bytecode and the assembler that intercepts the message and updates the persistent storage on the sensor node. In Chapter 6, I presented findings in comparing fuzzy queries with classic queries and showed that fuzzy queries can provide an increase in efficiency, trading of a level of accuracy.

In order to evaluate the fuzzy query technique, I created a tool that simulated a real world environment and also separated the changes in that environment from the network functions of the WSN. I designed a dynamic environment simulation architecture allows investigators to create their own scenario and use it to evaluate

WSN. This provides repeatable tests to evaluate independent and the creation of benchmarks to compare different solutions. The architecture is meant to be used as a design for implementation in various simulators. I implemented this architecture over TOSSIM simulator for TinyOS. I created three environment scenarios to demonstrate the use and flexibility of the architecture. To ensure authenticity, each scenario represents a CA model described by current research in earth science or physics. Of the scenarios I implemented: one was used to show the steps to build a scenario and to show the versatility of the architecture; the remaining two were used in the evaluation of fuzzy queries. The scenarios allowed me to compare fuzzy and classic queries in a repeatable environment.

To evaluate fuzzy queries in WSN, I conducted simulation with different scenarios and varying mote deployment configurations. The simulation results for Test 1, by comparing a fuzzy query with an equivalent classic query in a homogenous environment, proved that fuzzy queries return results that are correct. It also showed that in a constant, homogeneous environment, fuzzy queries offer no benefit over classic queries. The simulation results for Test 2 proved that in a constant heterogeneous environment, fuzzy queries can improve efficiency of a WSN. By using fuzzy query membership functions, general knowledge about the environment can be pushed onto the sensor node. This knowledge can be used, in-network, by the sensor node to determine whether or not it will participate in the query. Nonparticipation saves energy for the node. Test 2 also proved that the improved efficiency comes at a cost of accuracy of results. However, since fuzzy queries allow the user to pose a query semantically, allowing that the specifics of the

environment are known by the sensor node, the inaccuracy may be acceptable for the intent of the query. The simulations in Test 3 suggest that in a dynamic environment, improved efficiency with only a slight loss of accuracy by fuzzy queries; supporting previous tests. However, due to the nature of the variability of dynamic environments, more research is needed to understand the exact conditions which will benefit from fuzzy queries. The dynamic environment scenarios proved that the environment simulation architecture provides a reliable way to repeat tests. This allowed for comparisons between fuzzy and classic queries to be evaluated over the same conditions.

As for future work, the research presented in this dissertation can be extended in many directions. First scenario testing can continue to evaluate the environment conditions in which fuzzy query processing adds benefit. Second complex algorithms can be devised to better take advantage of fuzzy query processing in WSN. Third, if more than one sensor attribute can be defined as fuzzy, complex algorithms can be invented that use the union of intersection of those fuzzy attributes or their fuzzy values. Finally, cluster algorithms or correlation of sensor nodes using fuzzy values can be developed that more efficiently cooperate with neighboring sensors.

REFERENCES

[1] K. Henricksen and R. Robinson, “A survey of middleware for sensor networks: state-of-the-art and future directions,” in *Proceedings of the International Workshop on Middleware for Sensor Networks* (Melbourne, Australia, November 28 - 28, 2006). MidSens '06, vol. 218, pp. 60-65ACM, New York, NY, 2006.

[2] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker, "The Sensor Network as a Database,” Technical Report 0-771, Computer Science Department, University of Southern California, September 2000.

[3] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In Proceedings of the 2nd International Conference on Mobile Data Management, Hong Kong, January 2001. http://citeseer.csail.mit.edu/article/bonnetOltowards.html

[4] A,Faradjian,J.Gehrke,P.Bonnet, " GADT: A probability space ADT for Representing and querying the physical world,”In International Conference on Data Engineering, 2002.

[5] A.Mamdani,J.Estathiou,D.Pang "Inference under uncertainty.,”Expertsystems 85, Proc.5th Conf. British Comp Soc.,Specialist Group on Expert Systems,Dec17-19, 1985, University of Warwik, Great Britain, 181-190.

[6] R.Cheng,S.Prabhakar ,"Managing Uncertainty in Sensor Databases,” SIGMOD Record, Vol.32, No.4, December 2003

[7] Y.Yao,J.Gehrke,"Query Processing for Sensor Networks" ,Proceedings of the w2003 CIDR Conference.

[8] Y.Yao,J.Gerhke,"The Cougar Approach to In-Network Query Processing in Sensor Networks,” In SIGMOD Record, September 2002.

[9] S.R.Madden, M.M.Franklin, J.M.Hellerstein, W. Hong , "The Design of an Acquisitional Query Processor for Sensor Networks,” In sigmod'03, San Diego, CA, 2003.

[10] J.Galindo, A. Urrutia, M.Piattini, Fuzzy Databases Modeling, Design and Implementation, Idea Group Publishing, 2006.

[11] S. Madden, M. Franklin, J. Hellerstein and W. Wong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," in ACM Transactions on Database Systems, vol. 30, no.1, March 2005, 122-173.

[12] H.Hiedemann,F.Silva,C.Intanagonwiwat,R.Govindan,D.Estrin,D.Ganesan,"Building efficient wireless sensor networks with low-level naming," in Proceedings of the 18th ACM Symposium on Operation Systems Principles [21], pp.59-159, Oct. 2001.

[13] J.Elson, D.Estrin, "Sensor Networks: A Bridge to the Physical World". In *Wireless Sensor Networks,* edited by C.Raghavendra,K.Sivalingam, T.Znati , pp. 3-20, Kluwer Academic Publishers 2004.

[14] . Arampatzis, J.Lygeros and S. Manesis, "A Survey of Applications of wireless sensors and Wireless Sensor Networks," in *Proceedings of the 13thMediterranean Conf. on Control and Automation*, Cypus, Greece, 2005, pp.719-724.

[15] K.Pottie, "Wireless Integrated Network Sensors," in *Communications for the ACM* 43(5); 41 May 2000.

[16] T. Lindholm and F. Yellin. The Java Virtual MachineSpecification. Addison-Wesley Professional, secondedition, 1998.

[17] D. Estrin, R. Govindan, J. Heideman, S, Kumar, "Next Century challenges: scalable coordination in sensor networks" *in Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, Seattle, WA, USA, pp. 263-270, 1999.

[18] C.Chong,S.Kumar,"Sensor Networks: Evolution, Opportunities and Challenges," In *Proceedings of the IEEE*, Vol.91,No.8,Aug 2003, pp.1247-1256.

[19] The Jython Project, http://www.jython.org

[20] C.Intanagonwiwat,R.Govindan,DEstrin,"Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks"; in *Proceedings of the Sixth Annual International Conference on Mobile Computing and networks(MobiCom 2000)* August 2000.

[21] Sykes, R.I., Henn, D.S., Parker, S.F., and Gabruk R.S., "SCIPUFF – A generalized hazard dispersion model " in Ith Joint Conference on Applications of Air Pollution Meteorology with A&WMA, 1996.

[22] L. Hernandez Encinas, S. Hoya White, A Martin del Rey, G. Rodriques Sanchez, "Modelling forest fire spread using hexagonal cellular automata," in Proceeding of Applied Mathematical Modeling, Volum 31(6) pp. 1213-1277, 2007.

[23] Barranco,C.,Campana,J.,Medina,J. 2005.Towards a XML Fuzzy Structured Query Language. Proceedings of the Joint 4th Conference of the European Society for Fuzzy Logic and Technology, Barcelona, Spain, September 7-9, 2005.

[24] A.Mainwaring, J.Polastre, R.Szewczyk, D.Culler and J.Anderson. "Wireless sensor networks for habitat monitoring". Technical Report IRB-TR-2-006.1251

[25] I.F. Akyildiz,W.Su,Y.Sankarasubramaniam,E.Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks* 38,1 Elsevier Press, pp. 393-422, December 2001.

[26] Walus, K., Schulhof, G., Jullien,G.A., Zhang, R., Wang, W. , "Circuit design based on majority gates for applications with quantum-dot cellular automata" in Proceedings of the 38th Asilomar Conference on Signals, Systems & Computers, Pacific Grove CA, Nov. 2004.

[27] D.E.Culler,W.Hong,"Wireless Sensor Networks,"Cominunication *of the ACM* Vol.47,No.8, June 2004.

[28] J.Hill,M.Horton,R.Kling,L.Krishnamurthy, "The Platforms Enabling Wireless Sensor Networks," in *Communications of the ACM,* Vol. 47, No. 8, pp. 41-46, June 2004.

[29] M.Vieira,D.daSilva,Jr.,C.Coelho,Jr.,J.daMata,"Survey on Wireless Sensor Network Devices,"2003 IEEE.

[30] P. Juang, H. Okiu, Y.Wang, Y., et al. " Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," *In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X*), Oct. 2002

[31] S.Madden,,Franklin,,J.Hellerstein,W.Hong,"TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," *5th Annual Symposium on Operation System Design and Implementation (OSDI),* December 2002.

[32] T. Liu, C. Sadler, P. Zhang, and M. Martonosi, "Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet," Proceedings *of the 2nd International Conference on Mobile Systems, Applications and Services (MobiSYS'04),* June 2004.

[33] L. Iftode, C. Borcea, and P. Kang, " Cooperative Computing in Sensor Networks," in Computer Journal, vol. 29, no.4, 2004, pp.295-494.

[34] C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor Information Networking Architecture and Applications," in *IEEE Personal Communications*, vol. 8, no. 4, August 2001, pp. 52-59.

[35] C. Fok, G. Roman and C. Lu, " Rapid Development and flexible deployment of adaptive wireless sensor network applications," *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'05*), 2005.

[36] C. Fok, G. Roman, G., and C. Lu, "Mobile agent middleware for sensor networks: an application case study*," Proceedings of the 4th international Symposium on information Processing in Sensor Networks*, Los Angeles, California, April 24 - 27, 2005.

[37] S.Ganeriwal, C.C.Han, and M.B.Srivastava. "Going beyond nodal aggregates: Spatial average of a continuous physical process on sensor networks," In *NESL Technical Report,* August 2004.

[38] C.Guestrin, B. Bodic, R.Thibaux, M.Paskin, S. Madden, "Distributed Regression: an Efficient Fraemwork for Modeling Sensor Network Data". In IPSN'04 , ACM, Berkley, California, USA, 2004.

[39] A.Deshpande,C.Guestrin,S.Madden, J.Hellerstein,W.Hong,"Model-Driven Data acquisition in Sensor Networks". In Proceedings of the 30th VLDB Conference, Toronto Canada, 2004.1401

[40] J. Gehrke, and S. Madden, "Query Processing in Sensor Networks," in IEEE Pervasive Computing, vol. 03, no. 1, pp. 46-55, Jan-Mar, 2004.

[41] S.Ahn, and D.Kim, "Proactive Context-Aware Sensor Networks," in Proc. 3rd European Workshop of European Wireless Sensor Networks (EWSN 2006), Zurich Switzerland, 2006, pp.38-53.

[42] K.Whitehouse, F. Zhao, and J. Liu, "Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data," in Proc. 3rd European Workshop of European Wireless Sensor Networks (EWSN 2006), Zurich Switzerland, 2006, pp.38-53.

[43] T.Abdelzaher, B. Blum B, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru and A. Wood, "Envirotrack: Towards an environmental computing paradigm for distributed sensor networks, " The 24th International Conference on Distributed Computing Systems. Tokyo, Japan, pp.582-589, March 23-26, 2004.

[44] A. Fuggetta, G. Picco, and G. Vigna, "Understanding Code Mobility," in IEEE Transaction of Software Engineering,. 24, 5 (May. 1998), pp. 342-361.

[45] P.Levis, N.Lee, A.Woo, M.Welsh and D.Culler. TOSSIM: Accurate and scalable simulations of entire tinyOS applcitions. In SenSys, November 2003.
http://www.ks.uiuc.edu/Development/MDTools/mdx/tutorial/tinysim.html.

[46] C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor Information Networking Architecture and Applications," in IEEE Personal Communications, vol. 8, no. 4, August 2001, pp. 52-59.

[47] J.Hill, R.Szeqczyk, A. Woo, S.Hollar,D.Culler,K.Pister, "System Architecture Directions for Networked Sensors," in Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, 2000.

[48] D.Gay,P.Levis,R.vonBehren,M.Welsh,E.Brewer,D.Culler,"The nesC Language: A Holistic Approach to Networked Embedded Systems," In Proceedings of Programming Language Design and Implementation (PLDI) 2003, June 2003.NesC v1.1 Language Reference - http://nescc.sourceforge.net/papers/nesc-ref.pdf

[49] J.Galindo,J.M.Medina, O.Pons, J.C.Cubero, "A Server for Fuzzy SQL Queries". In Journal: Lecture Notes in Computer Science, Vol. 1495, 1998, pp.164-174.

[50] J.Blumenthal, M.Handy,F.Golatowski,M.Haase,D.Timmermann, "Wireless Sensor Networks –New Challenges in Software Engineering," Proceedings of 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Lissabon, Portugal, September 2003D.D.Chanberlin,et al, "SEQUEL 2: A Unified Approach to Data Definition, Manipulation and Control". IBM Journal of Research and Development, Vol. 20, Num. 6, November 1976, pp560- 575.

[51] Wolfram, S. Cellular Automata and complexity, Addison Wesly, Reading , Massachusets 1994

[52] Mathworks contributors, "Fuzzy Logic Toolbox.," *The Mathworks*, http://www.mathworks.com/access/helpdesk/help/toolbox/fuzzy/index.html?/access/helpdesk/help/toolbox/fuzzy/fp72.html&http://www.mathworks.com/ (accessed May 1, 2009).

[53] E. Souto, G. Guimaraes, G. Vasconcelos, M. Vieira, N. Rosa, and C. Ferraz, " A message-oriented middleware for sensor networks," *in Proc. 2nd Workshop on Middleware for Pervasive and Ad-hoc Computing*, Toronto Canada, 2004, pp. 127-134.

[54] M. Zoumboulakis, G. Roussos, and A. Poulovassilis, "Active Rules for Sensor Databases," in *Proc. of the 1st International Workshop on Data Management for Sensor Networks (DMSN 2004),* Toronto Canada, 2004,

pp. 98-103.

[55] N.W. Paton, and O. Díaz, "Active database systems," ACM Computer Survey 31, 1 (Mar. 1999), 63-103.

[56] S. Li, S. Son, and J. Stankovic, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks," In *Proc. Of the 2nd Intl Workshop Informatio Processing in Sensor Networks (IPSN 2003)* LNCS 2634, Springer 2003, pp 502-517.

[57] C.Shen,C.Srisathaornphat,C.Jaikaeo, "Sensor information networking architecture and applications," *IEEE Personal Communications*, August 2001, pp.52-59.

[58] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong," Approximate Data Collection in Sensor Networks using Probabilistic Models, *in Proc. of the 22nd Int'l Conf. on Data Engineering (CDE'06*), Vol.00, 2006, p. 48.

[59] P. Costa, L. Mottola and G. Picco, "Programming Wireless Sensor Networks with TeenyLIME Middleware," *in Proceedings of the 8th ACM/IFIPUSENIX International Middleware Conference (Middleware 2007*), Newport Beach CA, Nov 26-30, 2007.

[60] P. Costa, L. Mottola, A. Murphy, and G. Picco, "TeenyLIME: Transiently Shared Tuple Space Middleware for Wireless Sensor Networks," i*n Proceedings of the International Workshop on Middleware for Sensor Networks*, co-located with the 7th International Middleware Conference (Middleware), Melbourne, Australia, November, 2006.

[61] C. Curino, M. Giani, M. Giorgetta, A, Giusti, A. Murphy, and G. Picco, "TinyLIME: Bridging mobile and sensor networks through middleware," in *Proc. 3rd Int'l Conf. on Pervasive Computing and Communications (PerCom 2005)*, Phoenix, AZ, 2005, pp. 61-72.

[62] A. Deshpande,C. Guestrin,S. Madden, J. Hellerstein, and W.Hong, "Model-Driven Data acquisition in Sensor Networks," *in Proc. of the 30th VLDB Conf*., Toronto Canada, 2004.

[63] D. Tulone, and S.Madden, "PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks," in *Proc. 3rd European Workshop of European Wireless Sensor Networks (EWSN 2006),* Zurich Switzerland, 2006, pp.21-37.

[64] A. Soheili, V. Kalogeraki, and D. Gunopulos, "Spatial Queries in Sensor Networks," in *Proceedings of the 13th Annual ACM international Workshop on Geographic information Systems* (Bremen, Germany, November 04 - 05, 2005). GIS '05. ACM, New York, NY, 61-70.

[65] J.M. Medina, O.Pons, and A. Vila, "FIRST: A fuzzy interface for relational systems". In Proceedings of the Sixth International Fuzzy Sytems Assciation World Congress, pp. 402-412, Brazil, 1995

[66] R. Muller. G.Alonso, D. Kossman, "A Virtual Machine for Sensor Networks". In Proceedings of EuroSys 2007, Lisbon Portugal, March 21-23, 2007

[67] R. Muller. G.Alonso, D. Kossman, "SwissQM: Next Generation Data Processing for Sensor Networks". In Proceedings of 3rd Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10th 2007.

[68] M . Demmer, P. Levis, A. Joki , E. Brewer , and D. Culler, " Tython : A Dynamic Simulation Environment for Sensor Networks," USC Berkeley UCB/CSD-05-1372, 2005.

[69] E. Gamma , R. Helm ,R . Johnson , and J.M . Vissides, " Design Patterns: Elements of Reusable Object-oriented Software" , Addison-Wesley Professional Computing Series, Addison-Wesley 1995

[70] P.Bosc, O.Pivert,. SQLf: A Relational Database Language for Fuzzy Querying. In IEEE Transactions on Fuzzy Systems, Vol 3, No.1 Feb. 1995.

[71] P.Bosc, O.Pivert, Fuzzy querying in conventional databases. In Fuzzy Logic for the Management of Uncertainty, editors L.Zadeh, J.Kacprzyk, publisher John Wiley & Sons, Inc. 1992, pp 645-671.

[72] T.Ichikawa, M.Hirakawa, ARES: a relational database with the capacity of

performing flexible interpretation of queries. In IEEE Transactions on Software Engineering 12(5), pp. 624-634, 1986.

[73] A.Motro, 'A Trio of Database User Interfaces for Handling Vague Retrieval Requests, IEEE Data Engineering Bulletin, 12(2), 1989.

[74] A. Motro, 'VAGUE: A User Interface to Relational Data bases that Permits Vague Queries', ACM Transactions on Office Information Systems, Vol 6, No. 3, July 1988, Pages 187-214.

[75] D.Dubois,H.Prade. Using fuzzy sets in flexible querying: Why? And How?. In Proceedings of the 1996 Workshop of Flexible Query-Answering Systems (FQAS'96), pp.89-103, May 1996.

[76] K.Raman, "A Fuzzy Resolution of the Prisoner's Dilemma," Center For Study of Complex Systems Technical Report, CSCS-2002-001, University of Michigan Ann Arbor, Ann Arbor, MI 48109, U.S.A., 2002

[77] R.Godvindan, "Data Centric Routing and Storage in Sensor Networks". In Wireless Sensor Networks, edited by C.Raghavendra,K.Sivalingam, T.Znati , pp. 185-206, Kluwer Academic Publishers 2004.

[78] L.A.Zadah, Fuzzy Sets. In Information and Control, 8, pp.338-353, 1965.

[79] Zemankova-Leech, M. Kandel, A. Fuzzy relational databases: a key to expert systems. Koln, Germany: Verlag TUV Rheinland, 1984.

[80] Zemankova-Leech, M. Kandel, A. Implementing imprecision in information systems. In information Sciences, 37, 1985, pp. 107-141.

[81] Medina,J.M., Pons, 0., Vila, M.A., GEFRED: A generalized model of fuzzy relational databases. In Information Sciences, 76 (1/2), pp. 87-109, 1994.

[82] Bordogna,G, Lucarella, D., Pasi,G. A fuzzy object-oriented data model managing vague and uncertain information. In International Journal of Intelligent systems 14(7), 1999, pp. 623-651.

[83] Bordogna,G, Leporati, A. Lucarella,D, Pasi,G. The fuzzy object-oriented

database management system. In Recent issues on fuzzy databases, Bordgona,Pasi (Eds.) Physica-Verlag, 2000, pp.183-207.

[84] Umano,M. Fukami S. Fuzzy relational algebra for possibility distribution-fuzzy-relation model of fuzzy data. In *Journal of Intelligent Information Systems*, 3 1994, pp. 7-28.

[85] Jimenez, A. Posadas, .M. and Marfil J.M, "A probabilistic seismic hazard model based on cellular automata and information theory," in *Nonlinear Processes in Geophysics,* 12, pp 1-16, 2005

[86] D.L. Turcotte, "Scaling in geology: Landforms and earthquakes," *Proceedings of the National Academy of Science,* USA 1995 July 18; 92(15): 6697–6704.

[87] T. Liu, C. Sadler, P. Zhang, and M. Martonosi, "Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet," *Proceedings of the 2nd International Conference on Mobile Systems*, Applications and Services (MobiSYS'04), June 2004.

[88] Naylor, M. "OFC Slider Block Model," EPSRC Nania Cluster, 5/14/2008, www.ph.ed.ac.uk/nania/nania-projects-OFC.html

[89] M. Cosnar, J. Demongeot, and A/ Lebreton (eds). "Rhythms in Biology and Other Fields of Applications" in *Lecture Notes in Mathematics, Vol 49 Springer Verlog*, Berlin, Heidleberg, New York, 1983.

[90] Buckles, B.P. Petry, F.e. A fuzzy representation of data for relational databases. In *FuzzySet Systems*, 7, 1982, pp.213-226.

[91] Zadeh, L.A., Similarity relations and fuzzy orderings. In *Information Sciences 3*, 1971, pp.177- 200.

[92] Zadeh, L.A., Fuzzy sets as a basis for a theory of possibility. In *Fuzzy Sets and Systems* 1,1978, pp.3-28.

[93] Dubois,D. Prada, H. Possibility Theory, An approach to computerized processing for uncertainty. NewYork, Plenum Press, 1988.

[94] Prade, H. Testemale, C., Fuzzy relational databases: Representational issues and reduction using similarity measures. In J*ournal of American Society Information Sciences* 38(2), 1987, pp.118- 126.

[95] Prade, H. Testemale, C. Generalized database relational algebra for the treatment of incomplete/uncertain information and vague queries. In *Information Sciences 34*, 1984, pp.115- 143.

[96] Doman,M, Dahlberg, T., and Payton,J, "Simulation Environments Scenarios Using Cellular Automata for Wireless Sensor Network Analysis," In *Proceedings of the 42nd Annual Simulation Symposium (ANSS)*, San Diego, CA, 2009.

[97] Woo,A.Tong,T. and Culler.D, "Taming the underlying challenges of reliable multi-hop routing in sensor networks". In *SenSys2003* pages 14-27, 2003.

[98] Park, A. Perumalla, K., Protopopescu, V., DeNap, F, Gorman, Bryan, "On Evaluation Needs of Real-Life Sensor Network Deployments," in *Proc. of the Euorpean Modeling and Simulation Symposium,* 2006.

[99] Toffoli, T. Margolus, N., Cellular Automat Machines, MIT Press, 1987.

[100] Frisch, E., Hasslacher, B . and Pomeau,Y., Latttice-gas automata for the Navier-Stokes equation. *Physics Review Letters*. 56:1505, 1986

[101] Chopard, B. and Droz, M., Cellular Automata Modeling of Physical Systems, Godreche, C, Editor, Cambridge University Press, Cambridge, UK.1998.

[102] Olami,Z., Feder,H.J.S., Christensen,K., "Self-organized criticality in a continuous, nonconservative cellular automaton modeling earthquakes," *Physical Review Letters*, 1992, Vol: 68, Pages: 1244 – 1247.

[103] S. Kirkpatric, "Models of disordered- materials, in Ill Condensed Matter, Les Houches, *Echole d'Ete de Physicaque Theorique,* session xx1, Holland 1979.

[104] B. D. Malamud, D. L. Turcotte, "Cellular-Automata Models Applied to

Natural Hazards," *Computing in Science and Engineering*, Vol. 2, No. 3. (May 2000), pp. 42-51

[105] Cunha, R. O., Silva, A. P., Loureiro, A. A., and Ruiz, L. B. 2005. Simulating Large Wireless Sensor Networks Using Cellular Automata. *Proc. of 38th Annual Symposium on Simulation* (April 04 - 06, 2005). Annual Simulation Symposium. IEEE Computer Society, Washington, DC, 323-330. DOI= http://dx.doi.org/10.1109/ANSS.2005.40

[106] Karafyllidis, I., Thanailakis, A., "A model for predicting forest fire spreading sing cellular automata," in *Ecological Modeling*, vol 99. pp. 87-97, 1997.

[107] N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Wireless Communications*, pp. 6-28, December 2004.

[108] I.F.Akildiz, W. Su, Y.Sankarasubramaniam, and E. Cayirci,E, "A Survey on Sensor Networks," in *IEEE Communications Magazine*, vol. 40, no. 8, August 2002, pp. 102-114.

[109] T. Bokareva, N. Bulusu, and S. Jha,, "A performance comparison of data dissemination protocols for wireless sensor networks: *IEEE Global Telecommunications Conference Workshops,* 2004 IEEE, pp85-89

[110] H. Karl, and A. Willig, "A Short Survey on Wireless Sensor Networks," Telecommunication Networks Group (TKN) Technical Report Series, TKN-03-018, October 2003.

[111] http://www.cs.wustl.edu/mobilab/projects/agilla/

[112] A. Vargas., OMNet++ Web Page, http://www.omnetpp.org

[113] M. Loebbers, D. Willkomm, and A. Koepke, The Mobility Framework for OMNeT++ Web Page http://mobility-fw.sourceforge.net

[114] Ns-2: Network simulator-2, http://www.isi.edu/nsnam/ns/

[115] A. Kröller, D. Pfisterer, C. Buschmann, S.P. Fekete, and S. Fischer,Shawn: A new approach to simulating wireless sensor networks, *Proceedings of the Design, Analysis, and Simulation of Distributed Systems(*DASD ’05), San Diego, 2005.

[116] S. Hadim and N. Mohamed,”Middleware for Wireless Sensor Networks: A Survey,” in *Communication System Software and Middleware*, 2006. Comsware 2006. First International Conference, Jan.2006, pp. 1-7.

[117] K. Henricksen and R. Robinson, “A survey of middleware for sensor networks: state-of-the-art and future directions,” in *Proceedings of the International Workshop on Middleware for Sensor Networks* (Melbourne, Australia, November 28 - 28, 2006). MidSens '06, vol. 218, pp. 60-65ACM, New York, NY, 2006.

[118] W Masri, Z, Zoubir Middleware for Wireless Sensor Networks: A Comparative Analysis Network and Parallel Computing Workshops, 2007. NPC Workshops. *IFIP International Conference* 18-21 Sept. 2007 Page(s):349 - 356
Digital Object Identifier 10.1109/ICNPCW.2007.4351509

[119] M. Molla, and S. Ahamed, “A Survey of Middleware for Sensor Network and Challenges,” in Proc. 2006 i (ICPPW’06), Columbus OH, 2006, pp.223-228.

[120] A. Murphy, and W. Heinzelman,, “MiLAN: Middleware linking applications and networks,” University of Rochester, Technical Report TR-795, 2002.

[121] B.Karisnamachari, D. Estrin, S.Wicher, “The impact of Data Aggregation in Wireless Sensor Networks”, In *International Workshop of Distributed Event Based Systems(DEBS),*Vienna, Austria, July 22.

[122] P. De, A. Raniwala, S. Sharma, T. Chiueh, MiNT: A miniaturized network testbed for mobile wireless research, in: Proc. of the IEEE INFOCOM, 2005.

[123] F .E. Soulie, Y. Robert, and M. Tchuente, Eds. Automata networks in

computer science, Princeton University Press, Princeton, NJ, 1987.

[124] T.Toffoli and N, Margolus. "Cellular Automata Machines: A New Environment for Modeling" The MIT Press 1987.

[125] J.Agre,L.Clare, "An Integrated Architecture for Cooperative Sensing Networks", Computer, Volume 33, Issue 5, pp 106-108,May,2000

[126] S. Nath, P.Gibbbons, S.Seshan,Z.Anderson, Synopsis Diffusion for Robust Aggregation in Sensor Networks, In *Sensys04*, ACM, November 2004.

[127] C.Shrivastava,C.Buragohain,D.Agrawal,S.Suri," Medians and beyond: new aggregation techniques for sensor networks". In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems,* Baltimore, MD USA, 2004, pp. 239-249.

[128] K. .Kalpakis, V.Puttagunta, P.Namjaoshi, Accuracy vs. Lifetime: Linear Sketches for Approximate Aggregate Range Queries in Sensor Networks. In *ACM, DIALM-POMC '04*, Philadelphia, PA. ,October 2004.

[129] S..Madden,R.SZewczyk,M.Franklin,D.Culler, "Supporting Aggregate Queries over Ad-Hoc Wireless Sensor Networks", In *Workshop on Mobile Computing and Systems Applications,* 2002.

[130] J. Considine, F.Li, G.Kollios, J.Byers, "Approximate Aggregation Techniques for Sensor Databases". In *ICDE'04*, Boston, MA, 2004.

[131] S. Madden,,Franklin,,J.Hellerstein,W.Hong,"TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks", *In Proceeding of 5th Annual Symposium on Operation System Design and Implementation (OSDI),* December 2002.

[132] J.Elson, D.Estrin, "Sensor Networks: A Bridge to the Physical World". In Wireless Sensor Networks, edited by C.Raghavendra,K.Sivalingam, T.Znati , pp. 3-20, Kluwer Academic Publishers 2004.

[133] S. Hadim and N. Mohamed,"Middleware for Wireless Sensor Networks: A Survey", in *Communication System Software and Middleware, 2006.*

Comsware 2006. First International Conference, Jan.2006, pp. 1-7.