DYNAMIC PUSH AND PULL STRATEGY BASED ON DATA INTERPOLATION FOR WEB SENSOR NETWORKS

by

ROHITA MOHAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2007

ACKNOWLEDGEMENTS

It is a pleasure to thank the many people who made this thesis possible. It is difficult to overstate my gratitude to my thesis supervisor, Dr. Yonghe Liu. With his enthusiasm, his inspiration, and his great efforts to explain things clearly and simply, he helped to make thesis fun for me. Throughout my masters' thesis period, he provided encouragement, sound advice, good teaching, and lots of good ideas. I would have been lost without him.

I am indebted to my student colleagues, Jing, Sunil, Samik Ghosh for providing a stimulating and fun environment and also for being really helpful through out my thesis. I am especially grateful to my friend, Praveen Kumar for his support regardless of the matter. I am grateful to my graduate advisor in the department of computer science, Mr. Mike O' Dell for assisting me in many different ways.

I wish to thank my sister and brother-in-law, Kavita and Prashant for believing in me. Lastly, and most importantly, I wish to thank my parents, Shanti and Ganapathi Mohan. They bore me, raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

July 18, 2007

ABSTRACT

DYNAMIC PUSH AND PULL STRATEGY BASED

ON DATA INTERPOLATION FOR

WEB SENSOR NETWORKS

Publication No. _____

Rohita Mohan, M.S

The University of Texas at Arlington, 2007

Supervising Professor: Dr. Yonghe Liu

Sensor Web, a new trend has been developed recently to effectively network large scale deployed heterogeneous sensors with Internet to make sensor data discoverable, accessible and controllable via World Wide Web. Since sensor nodes are highly resource constrained, continuous backhauling of a large amount of data is not feasible. In this paper, we propose a dynamic push/pull service architecture where sensory quality is dynamically monitored and controlled. The architecture is implemented in a web proxy residing at the gateway of the network. As data is being pushed continuously from the network for effective sketching of the sensing field, interpolation techniques are employed to further render the details. Due to varying user

requirements and dynamics of the sensing field, pushed data only may lead to low quality of the interpolated data. In this case, the system will dynamically pull data from the sensing field based on the requirements of the interpolation algorithms for desired confidence. Also, continuous demands of data pulling will eventually be converted to pushing services in order to reduce system overhead. The overall result is an architecture where push and pull services are dynamically monitored in order to reduce the overall system energy while satisfying user's demand for sensory data quality.

Simulation of a web proxy has been conducted to continuously regulate the push and pull rates of temperature data based on Spatial Point to Point Interpolation algorithm, which has been proved appropriate for GIS applications. From our simulation results, the optimal degree of push and pull rates is determined to maintain high quality of sensor data delivered to the users while reducing overall system energy but it can also be seen that with an increase in the accuracy of the interpolated data, there is a tradeoff in the energy consumption level of sensor nodes.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Table Page

CHAPTER 1

INTRODUCTION

1.1 World Wide Sensor Web

1.1.1 Background

Recent advances in micro-sensor technology have led to the quick development and large-scale deployment of low cost and low power sensing devices with computational and wireless sensing capabilities. This large number of sensors can be networked in many applications that require unattended operations, hence producing a wireless sensor network (WSN) that can be used in both civilian and military applications, such as intrusion detection, detecting the presence of hazardous material, weather monitoring etc. Because of their reliability, accuracy, flexibility, cost-effectiveness and ease of deployment, they are envisioned to be large scale and ubiquitous according to their recent development, and have been deployed in many diverse fields such as wildlife habitat monitoring [1], traffic monitoring [2] and lighting control [3]. Current works consider WSN as being designed for specific application, with data communication protocols strongly coupled to the applications. In fact, application is crucial to the network requirements, organization and routing behavior. But for such applications, there must be an effective way for user to gain access to the data produced by the wireless sensor networks.

1.1.2 Sensor Web

Sensor Web [4], a new technology has been developed recently to effectively network large scale deployed heterogeneous sensors with Internet to make sensor data discoverable, accessible and controllable via World Wide Web. By connecting the WSNs to Internet, remote access to the WSN can be achieved. To achieve this, many architectural solutions have been proposed in papers such as [5-7] for integrating WSNs and Internet where application specific requirements are separated from the data dissemination functions.

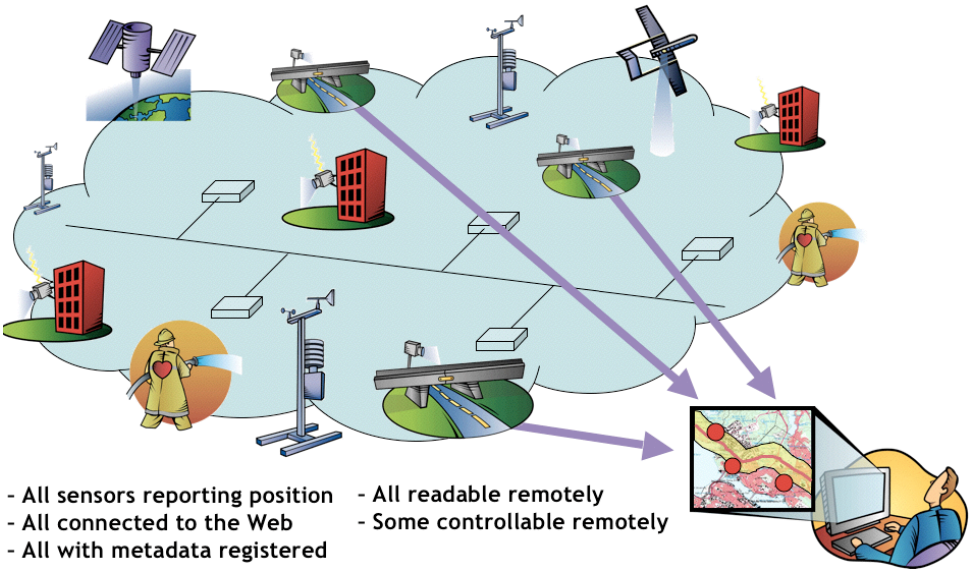

Figure 1.1 Sensor Web Concept.

1.1.3 Previous Work

An essential aspect of most applications is the ability to support remote interactions with WSNs. By the integration, users around the world can access the sensor data easy by Internet. The real-time application between Internet and WSN is

feasible, and it makes sensing data always online realization. Rich sensor data sources from different WSNs can be processed together, which will broaden the WSNs' applications. With Internet's advantages, some bottlenecks of WSNs will also vanish gradually. For many applications, however, it makes a lot of sense to integrate WSNs with Internet.

WSNs often are intended to run specialized communication protocols, and we cannot give an IP address to every sensor node for their large numbers, thereby making it impossible to directly connect WSNs with Internet by TCP/IP [8, 9]. To integrate WSNs with Internet, proxy is a possible architecture indicated in recent papers such as [1, 10]. As it can be seen in Fig. 1.2, a special proxy server is deployed between WSN and Internet. The proxy operates in two ways: relay and front-end. In the first case, the proxy simply relays the data coming from WSN to users in the Internet. In the second case, where the proxy acts as a front-end for the WSN, the proxy pro-actively collects data from the sensors and stores the information in a database. The users can query the proxy through SQL-queries or web based interfaces.

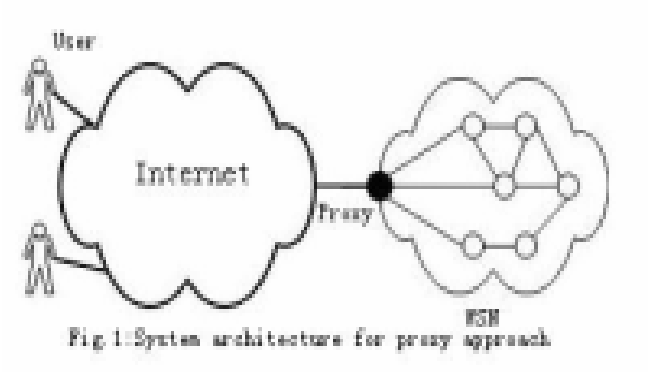

Figure 1.2 System architecture for proxy approach.

Another architecture that has been proposed in [10], is the Service-Oriented model to integrate Wireless Sensor Networks with the Internet. This design is similar to proxy architecture except that it separates the complicated management such as authenticating users, part aggregating sensor data, managing and analyzing queries from the gateway or the proxy. This way, the load on the proxy is minimized by delegating functionalities to different components or rather to different service agents. It proposes an integrated solution with application agent (AA), resource manager (RM), register agent (RA) and multi-gateway. It also provides a Multi-gateway solution to solve the bottleneck in a single gateway or proxy in the integrated networks. A new and special component RM (Resource Manager) designed plays a role of authenticating users, processing queries, aggregating and storing related data. At the same time, to provide user a quick response and decrease the load in gateway, a database with historical and recent records in RM is set up. The various agents have been defined below.

The Application Agent (AA) acts as a service requestor. It can help users to register some services to RA, and can find RM for each user's request according to the information provide by RA. After registering, user can get the service description by a lot of attribute-value pairs.

Register Agent (RA) is a service registry. It is responsible for the service register. It can broadcast their service to AA, collect different services description and get the related authentication information from RM.

Resource Manager (RM) is a crucial component in this architecture. It plays a role of authenticating a user, processing the query received from the user and

responding to the user according to its query analysis processor and databases, and also triggering a real-time query to the gateway if the user's queries need.

Here WSNs have been classified into two types on the basis of their mode of operation or functionality and the type of target application: proactive and reactive networks. In proactive networks, the nodes periodically switch on their sensors and transmitters, sense the environment, and transmit the data of interest. In reactive networks, the nodes react immediately to sudden and drastic changes in the value of a sensed attribute.

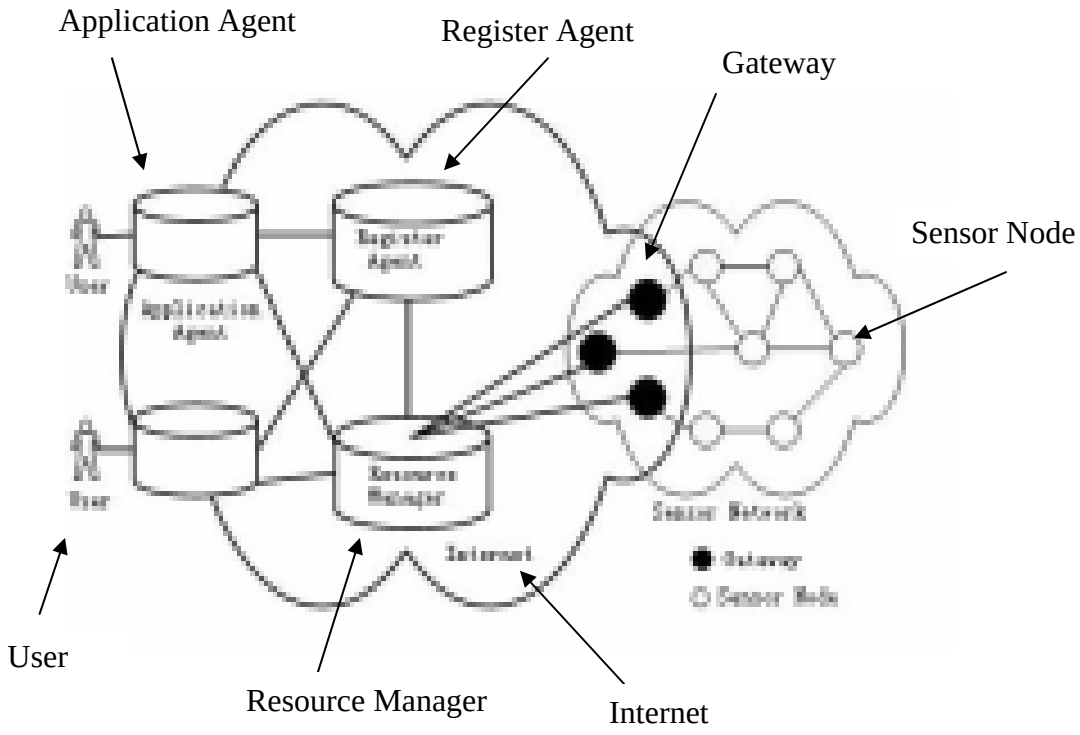

Figure 1.3 System architecture for service-oriented model.

1.1.4 Drawbacks

A major drawback of these two architectures is that, during the data transfer process between WSN and the proxy; there is a continuous backhauling of large amount of data in sensors. Since sensors are highly resource constrained, continuous transfer of data is not always feasible. After a certain amount of time of transfer, the nodes are sure to die out one by one ultimately resulting in the degradation of the service provided to the clients of the network [11-14]. The greater the amount of time a given node of the network is awake, the shorter the life span of that node. This problem has been addressed in many of the recent papers such as [15, 16]. The life span of the network must, therefore, be maximized but the quality of the sensory data received from the network must also not be compromised. Thus, there needs to be a dynamic solution that solves this problem by minimizing the overall energy of sensor network while still maintaining high quality of data delivered to the users.

1.2 Dynamic Push and Pull Strategy

In this paper, we propose a dynamic push/pull service architecture where sensory quality is dynamically monitored and controlled. The architecture is implemented in a web proxy residing at the gateway of the network. The proxy can be tied up with many other components for delegating different services such as authentication, registering services etc. Since it is deployed between WSN and the Internet, it reduces the computational overhead imposed on the sensors by performing all heavy computational operations that would be required to process the raw data from the sensors to answer users' queries. For this purpose, it is equipped with a stable

storage media (database) to save the sensor data and a powerful processing capability to aggregate sensor data, manage and analyze queries.

In our dynamic system, during the initial stage, as data from a defined number of sensors is being pushed continuously from the network for effective sketching of the sensing field, interpolation techniques are employed to further guess the data of the rest of the sensors. Due to varying user requirements and dynamics of the sensing field, pushed data only may lead to low quality of the interpolated data. In this case, the system will dynamically pull data from the sensing field based on the requirements of interpolation algorithms for desired confidence. Also, continuous demands of data pulling will eventually be converted to pushing services in order to reduce system overhead. Thus, the system adjusts the push rate of the network accordingly to ultimately converge into an optimal number of sensors that would be required to remain active to achieve desired accuracy of sensory data. The overall result is an architecture where push and pull services are dynamically monitored in order to reduce the overall system energy while satisfying user's demand for sensory data quality.

Simulation of a web proxy has been conducted to continuously regulate the push pull rate of temperature data based on Shepard's [17] interpolation method, which has been proved appropriate for GIS applications. From our simulation results, the optimal degree of push and pull rates is determined to maintain high quality of sensor data delivered to the users while reducing overall system energy but it can also be seen that with an increase in the accuracy of the interpolated data, there is a tradeoff in the energy consumption level of sensor nodes. Also, it can be observed that the time taken by our

dynamic system to converge into a minimal push rate value for achieving desired data accuracy depends upon the number of rounds of adjustments made to the push rate of the network to achieve desired accuracy of sensory data.

CHAPTER 2

THEORY & CONCEPTS

2.1 Sensor Web Architecture

Our architecture consists of three main components that include WSN, Proxy and Internet. The users that are connected to the Internet act as clients to the web proxy server and pose appropriate queries for retrieval of required data from the network. Proxy that resides at the gateway of WSN has a database at its back end to store sensor data from the network which describes the push mechanism of sensor network. It basically acts as both a client and a server. It acts as a server to the users by accepting their queries, parsing them, and giving appropriate results back to the clients. It acts as a client by retrieving available raw data from the database according to the queries, and in case sufficient data is not available, it pulls required data from the sensors, processes them and sends them back to the users. WSN comprises of sensors that monitor their environment on a periodic basis and continuously push data to the database of the web proxy for effective sketching of the sensing field. Our components have been discussed in detail below.

2.1.1. User Queries

Users residing at the Internet, pose different kinds of queries to the proxy server. Initially, the users are informed about the types of services available from the sensors. These services may include temperature, humidity, pressure, light etc. related data that can be monitored by the sensors and made available to the users. Based on the services

available, the user may decide to register to one or more services available and accordingly may require information to be displayed to them on a daily basis. For example, if a user decides to seek temperature data from a particular region monitored by sensors, his query may look like “Select average (temperature) from all sensors (covering that region) where time = current timestamp and epoch = 1 hour”[18]. In this case, the user would like to receive the average of temperatures of all sensors controlling a particular region every 1 hour starting from the current time given. Another query could look like “Select average (temperature) from a particular sensor for all timestamps on a particular day”. Similar queries can be framed for other sensing parameters such as humidity, pressure etc. In our experiments, temperature has been used as our primary sensing field that would be required by the users.

2.1.2. WSN

Sensors in WSN are continuously monitoring their environment and sending raw data to the proxy that gets stored in the database at regular intervals of time. Since sensors have low power and limited resources and also are largely deployed in unattended areas, it may not be possible for them to expend energy and power continuously at all times. By reducing the number of sensors that remain active at one point of time, the overall system energy can be conserved to a greater extent. Thus, there needs to be a way by which the number of sensors that remain active can be reduced while still maintaining high sensory data quality to satisfy user’s demand.

In this paper, a dynamic push and pull strategy has been proposed for determining the optimal number of sensors that would be required to remain active for

obtaining sensory data quality of a desired confidence. During the process of calculating the optimal push rate, the sensor nodes play a major role of sending data to the proxy according to a defined push rate (number of active sensors) pro-actively and also reactively by sending data to the proxy when the proxy requests for data from specific sensors.

2.1.3. Web Proxy

Web Proxy plays a major role in our architecture as it is responsible for parsing the user queries, retrieving appropriate data from the network, processing the raw data and sending the results back to the users. There are two major operations involved in retrieval of raw data from the network. They are known as push and pull processes [10].

2.1.3.1 Push Process

Push process is described as one in which the sensors send their raw data to the proxy proactively, that eventually gets stored in the database, without actually being triggered by the proxy. This means that data is periodically being sent by the sensors at a defined rate and stored in the database for answering possible queries that may be posed by the users. This method actually saves time for the proxy as data is automatically made available to the proxy for further processing instead of actually sending a request to the sensors for the appropriate data and the nodes sending data back to the proxy. The latter process almost takes double the time and energy as the former push process as there is a roundtrip of request and response being involved in the pulling process.

Since sensor nodes are highly resource constrained in terms of power and energy, continuous transfer of data to the proxy at all times is not always feasible. After a certain amount of time of transfer, the nodes are sure to die out one by one ultimately resulting in the degradation of the service provided to the clients of the network. The greater the amount of time a given node of the network is awake, the shorter the life span of that node. The life span of the network must, therefore, be maximized but the quality of the sensory data received from the network must also not be compromised.

Our implementation of sensor web architecture mainly focuses on reducing overall system energy while still maintaining high sensor data quality to satisfy user's demand. In our implementation, the sensor nodes send data to our proxy database at a minimal defined rate in terms of number of sensors sending data at a particular time. This rate defines the minimal number of sensors that need to remain active so as to ensure high quality sensory data. This way, the overall system energy is reduced as only a certain number of sensors remain active at one point of time, thereby increasing the longevity of the network. Also, the number of active sensors is carefully chosen such that the quality of sensory data remains close to the actual values with a desired confidence.

2.1.3.2 Pull Process

During this process, the proxy triggers the sensor nodes for required data at a particular timestamp and the nodes send the raw data to the proxy for immediate processing. This can also be termed as a reactive process as the sensors immediately react to the request sent by the proxy, by sending data. This action consumes more

energy than the pull process as there is a two-way communication between the proxy and the sensors for both the request and response. Thus, the sensors undergo almost double the communication overhead that would have been imposed with push process.

In our implementation, we try to dynamically calculate the number of sensors that need to remain active so as to achieve data quality with a desired confidence. This cannot be achieved without the pull process coming into the context. After the system sets the push rate or the number of active sensors to a certain minimal value, there needs to be a method by which the values of the rest of the sensors can be calculated so as to acquire the sensory values of all the sensors. This method is known as interpolation [19]. With the help of interpolation algorithms, the values of the unknown data can be calculated using the values of the known data. Due to varying user requirements and dynamics of the sensing field, the interpolated values may lead to low quality of data. This is the place where pull process plays a vital part. For improving the quality of sensory data, the proxy will start to pull data from the network based on the requirements of interpolation algorithm for desired confidence. Since this process inquires a lot of system overhead, the number of pulls can eventually be converted to pushing services so as to reduce the overall system overhead. Thus, with the help of this dynamic push and pull strategy and network learning of interpolation model, we can arrive at an optimal value for the push rate or number of active sensors that would ensure high sensory data quality to satisfy users' demands.

Figure 2.1 Dynamic Push and Pull System.

2.2 Interpolation

In our architecture, sensor data is not obtained from all of the sensors that comprise the network for obvious reasons such as power management and minimizing the overall system energy to increase the longevity of the network. The implication of this is that we may need a mechanism to guess values of the data of those sensors whose data was not obtained during the push process, should these values be required by the client. Interpolation is an instrument to achieving this. It has been used as far back as 300 B.C. to interpolate the locations of celestial bodies and is presently used in image and signal processing [20].

2.2.1. Definition

Interpolation is a procedure for estimating unknown values using existing known values at sampled points. Many interpolations exist for different domains [17, 20]. These can be of varying complexity and accuracy. There are two broad categories

of interpolation, namely spatial and temporal [21]. Spatial interpolation [22] allows an estimation of data to be made, let's say temperature for example, at a particular point given that surrounding temperatures are known. This would allow a client of the WSN to request the temperature at any point from where sensory data was not collected. Temporal interpolation in contrast would be able to approximate the temperature of a given point at a particular instance in time, even if the sensor was not sensing at that exact moment, as long as at least one previous and one subsequent sensor values were available.

Spatial Interpolation is more suitable for GIS applications [23] as estimation of unknown values will be based on the locations of the surrounding points with respect to the unknown points. In our architecture, we are well aware of the locations of the sensor nodes in the network and considering sensory data to be of type temperature, pressure, humidity etc. it can be deduced that the values of the unknown data in specific locations would definitely be influenced by the known data values in neighboring locations. Thus, we have considered spatial interpolation as our primary interpolation technique in our implementation.

Our implementation can also be extended in future to incorporate temporal interpolation method to interpolate unknown data of one particular sensor at a specific timestamp using data from that sensor known at different timestamps.

2.2.2. Spatial Point Interpolation

Spatial interpolation procedures can be classified into many categories such as point, areal, global, local etc [24]. Point interpolation best suits our application as the

values of other points at predetermined locations can be determined, given a number of points whose values and locations are known. They are used for data which can be collected at point locations such as weather station readings, spot heights, porosity measurements etc. There are a variety of algorithms that describe spatial point interpolation [25, 26]. Few of them are Inverse Distance Weighting (Shepard's method), Kriging, and Splines etc. A glimpse of their comparison has been given in following section.

2.2.2.1 Comparison of Spatial Interpolation methods

This section describes few of the spatial interpolation methods, compares them [27] and concludes stating the better interpolation method that can be used for our implementation.

Inverse distance weighted averaging (IDWA) is a deterministic estimation method where values at unsampled points are determined by a linear combination of values at known sampled points. Distance-based weighting methods have been used to interpolate climatic data. IDWA makes the assumption that values closer to the unsampled location are more representative of the value to be estimated than samples further away. Weights change according to the linear distance of the samples from the unsampled point. The spatial arrangement of the samples does not affect the weights. IDWA has seen extensive implementation in the mining industry due to its ease of use. IDWA has also been shown to work well with noisy data. The choice of power parameter in IDWA can significantly affect the interpolation results. As the power parameter increases, IDWA approaches the nearest neighbor interpolation method

where the interpolated value simply takes on the value of the closest sample point. Optimal inverse distance weighting is a form of IDWA where the power parameter is chosen on the basis of minimum mean absolute error.

Splining [28] is a deterministic technique to represent two dimensional curves on three dimensional surfaces. Splining may be thought of as the mathematical equivalent of fitting a long flexible ruler to a series of data points. Like its physical counterpart, the mathematical spline function is constrained at defined points. Splines assume smoothness of variation. Splines have the advantage of creating curves and contour lines which are visually appealing. Some of splining's disadvantages are that no estimates of error are given and that splining may mask uncertainty present in the data. Splines are typically used for creating contour lines from dense regularly-spaced data. Splining may, however, be used for interpolation of irregularly-spaced data.

Kriging [28] is a stochastic technique similar to inverse distance weighted averaging in that it uses a linear combination of weights at known points to estimate the value at an unknown point. Kriging is named after D.L. Krige, who used kriging's underlying theory to estimate ore content. The general formula of kriging however, was developed by Matheron (1969). Kriging uses a semivariogram, a measure of spatial correlation between two points, so the weights change according to the spatial arrangement of the samples. Unlike other estimation procedures investigated, kriging provides a measure of the error or uncertainty of the estimated surface. In addition, kriging will not produce edge-effects resulting from trying to force a polynomial to fit the data as with TSA.

Out of these methods, Shepard's method or IDWA is quite simple, straightforward, easy to use and computationally less expensive. It gives good results on an average and is also an exacting function that honors the sample data points while interpolating data. Thus, we have employed IDW method for interpolating data at unknown data points.

2.2.2.2 Shepard's method

The simplest form of inverse distance weighted interpolation is sometimes called Shepard's method [17, 29]. This method is based on the assumption that the interpolating surface should be influenced most by the nearby points and less by the more distance points. The interpolating surface is a weighted average of the scatter points (known data points) and the weight assigned to each scatter point diminishes as the distance from the interpolation point to the scatter point increases. The equation used is as follows:

$$F(x,y) = \sum_{i=1}^{n} w_i f_i$$

Where n is the number of scatter points in the set within a defined search radius, f_i are the prescribed function values at the scattered points (e.g. the data set values), and w_i are the weight functions assigned to each scatter point. The classical form of weight function is:

$$w_i = \frac{h_i^{-p}}{\sum_{i=1}^{n} h_i^{-p}}$$

Where p is an arbitrary positive real number called the power parameter (typically, p=2) and h_i is the distance from the scatter point to the interpolation point or

$$h_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$$

Where (x,y) are the coordinates of the interpolation point and (x_i,y_i) are the coordinates of each scatter point. The weight function varies from a value of unity at the scatter point to a value approaching zero as the distance from the scatter point increases. The weight functions are normalized so that the weights sum to unity.

The effect of the weight function is that the surface interpolates each scatter point and is influenced most strongly between scatter points by the points closest to the point being interpolated.

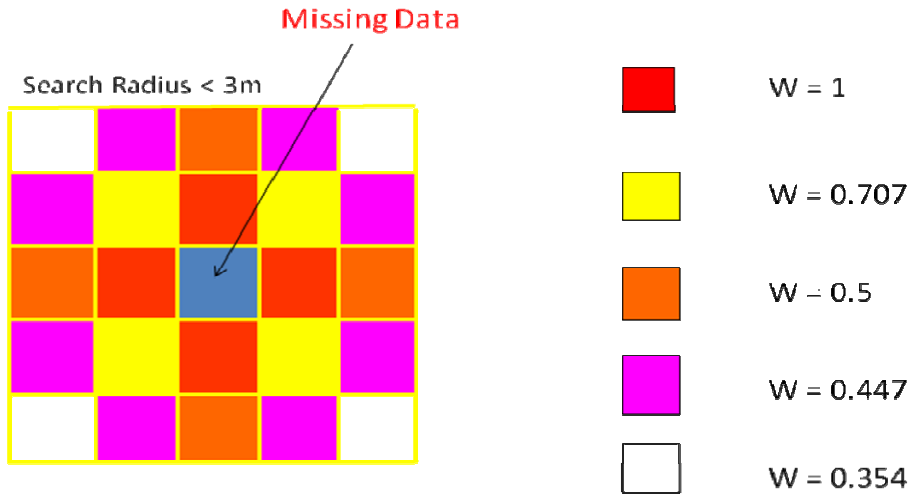

Figure 2.2 The above figure shows the weight values for Shepard's method by taking inverse of the scatter point's distance from the missing data raised to power of 1.

There are two parameters here that can be tuned to get the desired type of interpolated data. They are power p and the number of scatter points, n. As the power p increases from 2, 3 to 4, the interpolated data takes the value of its neighboring scatter point. The number of scatter points can be varied according to the desired distance range from the interpolating point to the scatter points. In our implementation of Shepard's method, a power parameter p was set to 2 and n was set to include all scatter points as it gave us desired results for all of our interpolated points.

2.2.2.3 Leave-One-Out Cross Validation

In our method of implementing dynamic push and pull strategy, we receive data from only a limited number of sensors as defined by our push rate and hence, are not aware of the true temperature surface, when considering temperature as our sensing field. Thus, we need to have a technique by which the accuracy of our interpolation can be measured with just known values of scatter points. One of the methods that can be used to evaluate the accuracy of the interpolation is the leave-one-out cross validation technique [19, 30].

In essence, this technique can be used when there is a predictive function that is to be evaluated according to some data set. An element is temporarily removed from the set. The rest of the set is then used by the function to compute the predicted value for that element and then this is compared to the actual value giving the error for that particular element. This is repeated for all elements in the set and gives us the cross validation errors for each of the elements in the set.

When applying this technique in our logic, we have data from a defined number of active sensors at a particular instant of time and this data is considered to be a known data set. The data of one of the sensors is removed temporarily and the data of the rest of the sensors is then used by Shepard's method of interpolation to calculate the interpolated value for that sensor which is then compared with the actual value giving the error for that particular sensor known as cross validation error. This error is then checked for desired accuracy or confidence by comparing with the permissible error calculated for desired accuracy. This process is repeated for the rest of the sensors in the know data set and the cross validation errors thus calculated are all compared with their permissible errors for desired accuracy. This way, the accuracy of the interpolated data can be measured and checked for desired confidence.

2.3 Implementation

This section of the report describes the algorithm, methods and data structures that have been employed to implement the dynamic push and pull strategy based on Shepard's Interpolation method for getting the optimal push rate or the minimal number of sensors that need to remain active to achieve sensory data quality of desired accuracy.

2.3.1 Algorithm

The implementation of the dynamic push and pull logic mainly resides in the web proxy component of our architecture. Since proxy is equipped with sufficient power and resources to carry out expensive computations and processing when compared to the sensor nodes in WSN, the processing of raw data is concentrated in this

component. So sensor nodes only transmit raw data at regular intervals to the proxy while the proxy is responsible for changing the raw data into useful data in response to user queries.

As mentioned earlier, our logic mainly focuses on delivering high quality sensor data of all sensors to the users, with only a minimal number of sensors being active, thus reducing overall system energy. A diagrammatic representation of our algorithm is given below.

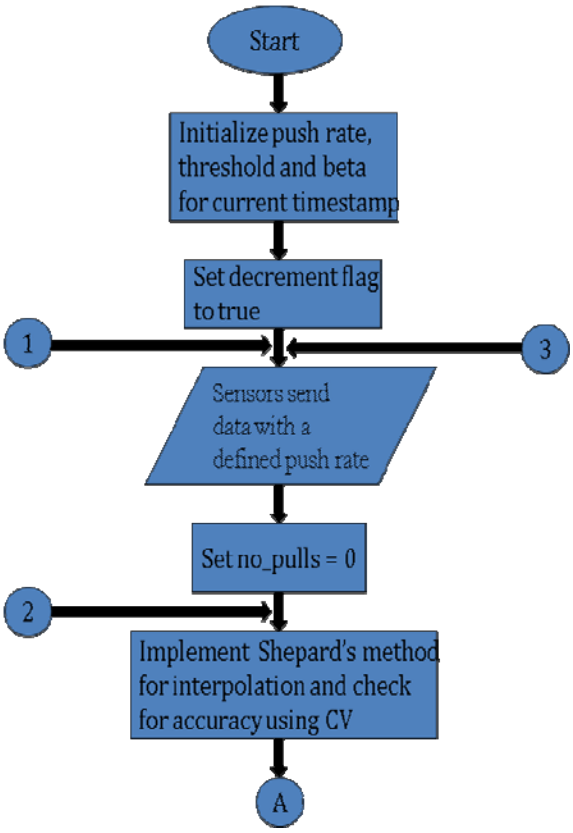

Figure 2.3 First half of the flowchart that represents the algorithm for dynamic push and pull system.

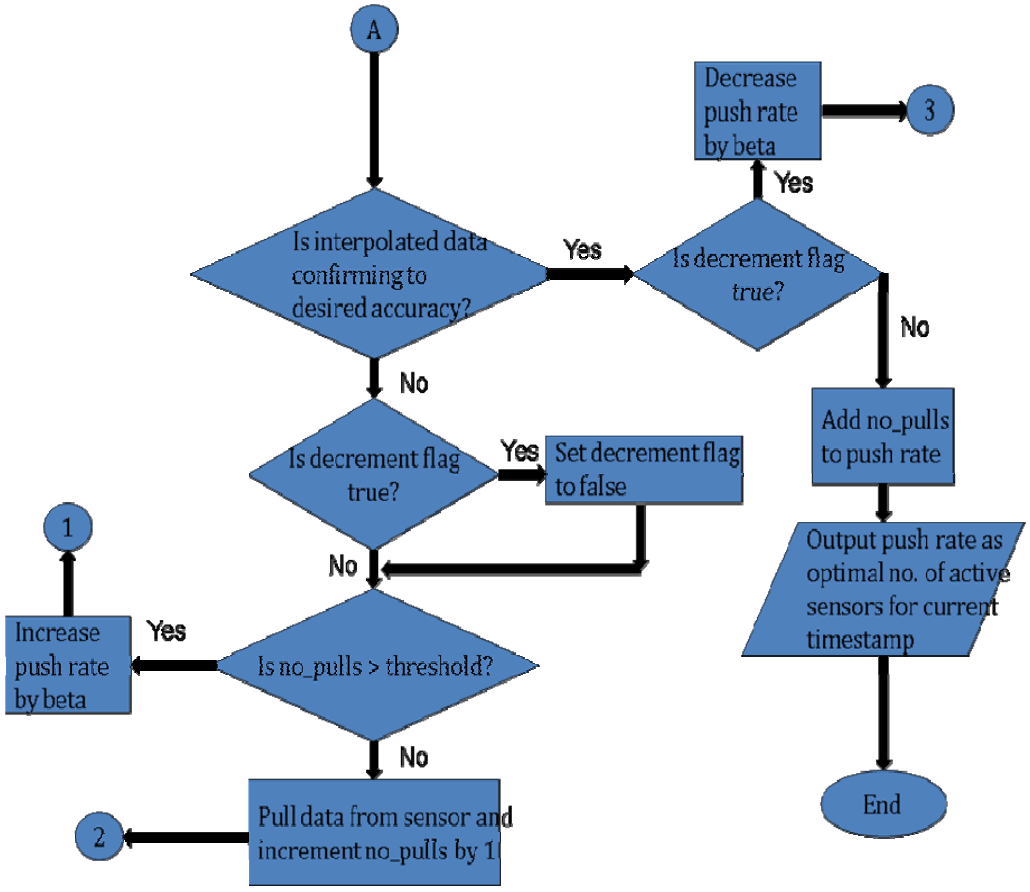

Figure 2.4 Second half of the flowchart that represents the algorithm for dynamic push and pull system.

The algorithm pertaining to the above flowchart is explained below.

1. To start with, a set of parameters that take part in our implementation are initialized to given values for current timestamp. These include push rate, threshold and beta. Push rate signifies the number of sensors that would initially remain active to send data to the proxy for further processing. Threshold indicates the maximum number of pulls that can be made by the proxy for

improving the accuracy of the interpolated data before increasing the push rate of the sensors and beta represents the value that would be added to the push rate once the number of pulls made by the proxy exceeds the threshold value. The threshold and beta values can be tuned appropriately to suit different requirements of applications.

2. A decrement flag is set initially so as to arrive at a minimal push rate value in the beginning by decrementing the given push rate. Sometimes the given initial push rate may be higher than the optimal push rate value of sensors and thus the value would be required to be reduced so that the program can output the minimal push rate that achieves the desired data accuracy first. If the given push rate immediately achieves data of desired accuracy, then the push rates lower than the given value may also have a chance of giving accurate data as they have not been tested for, earlier and hence the push rate is decremented till it reaches the nearest value that does not produce accurate data. Having this minimal push rate with us, we can then begin to run our algorithm by incrementing the push rate and pulling sensor data until we achieve the minimal optimal push rate for achieving data with desired confidence. Setting the decrement flag allows us to keep track of the point up to which the push rate would be required to be decremented initially just to make sure that we arrive at a push rate lesser than the minimal push rate value for achieving data of desired accuracy.
3. Next, the sensors whose number is defined by the push rate parameter begin to send data to the proxy pro-actively and the proxy stores the data in the database.

4. We also maintain a counter, for the number of pulls that would be made by the proxy so as to tune the interpolated data for desired accuracy, for comparing it with the threshold value and making appropriate decisions. This parameter is called no_pulls and is initialized to zero.
5. The Shepard's method of interpolation is then implemented to estimate the values of unknown data points of sensors at predetermined locations using known data set.
6. The accuracy of the interpolated data is then calculated using the Leave-One-Out Cross Validation technique [19] and the errors thus obtained for each of the known sensor data are then compared with the permissible errors calculated for desired accuracy.
7. If the cross validation error of even one of the sensors falls above the permissible error for desired accuracy, then the known sensor values are not enough to interpolate data to a desired accuracy and hence would require more sensors to send data to achieve the desired accuracy.
8. So as to improve the accuracy of interpolation, the proxy will begin to pull data from the sensors. After every pull, the no_pulls counter is incremented by one and the process gets repeated from step 4.
9. The above process is continued as long as no_pulls counter remains less than the threshold value. Once the threshold value is exceeded, the push rate or the number of active sensors is increased by beta and the process gets repeated from step 2. The increase in push rate is carried out so as to minimize the overall

system overhead due to large number of pulls as pull process takes almost twice the energy compared to push process.

10. If the cross validation errors of all the available sensors' data satisfy the permissible errors for desired accuracy, this implies that the known set of values are just enough to interpolate data that confirms to the desired accuracy and the sum of push rate and no_pulls value is output to signify the number of sensors that would be needed to remain active to achieve the desired confidence for the current timestamp. This algorithm can be repeated for subsequent timestamps if desired.

Thus, following the above algorithm would give us the minimal push rate or the minimum number of sensors that would need to remain active for achieving desired sensory data quality during a particular timestamp or dataset recording by the sensors.

2.3.2 Program

The dynamic push and pull strategy was implemented in Java and MySQL database was used as a back end storage medium for storing sensor data from the sensors.

Our program used Intel Berkeley Research Lab temperature data [31] from 54 sensors that were monitored for a period of one month. A sample of the data was stored in our database for providing us with streaming sensor data. Temperature data was our main field of interest. Our sensor data mainly comprised of sensor id, location in terms of x axis and y axis in meters, temperature and timestamp. Since we were dealing with

spatial characteristics to estimate data, locations of sensors were a required field for all of the sensors.

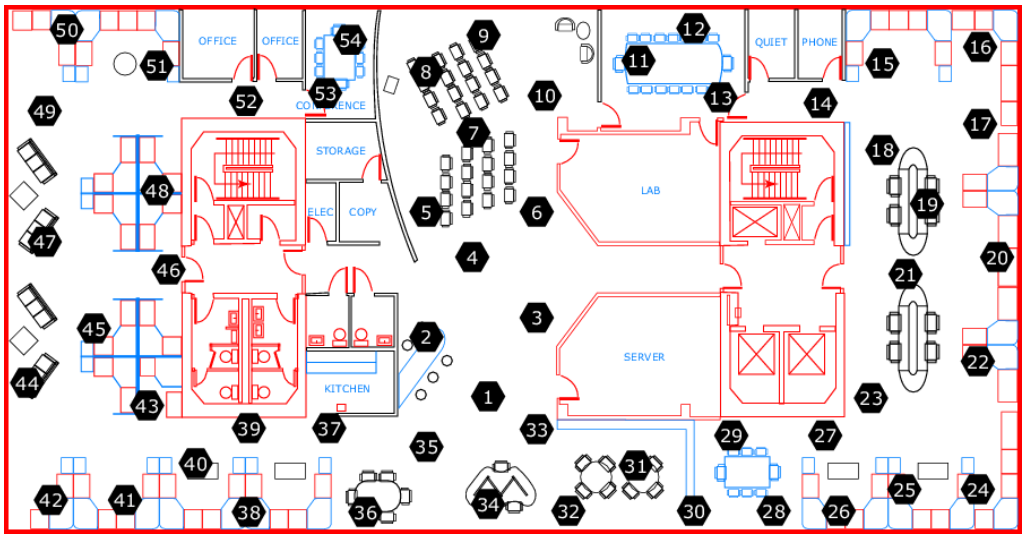

Figure 2.5 The above figure shows the arrangement of sensors in Intel lab.

Table 2.1 The locations of first five sensors are shown in the table.

Sensor ID	X axis (in meters)	Y axis (in meters)
1	21.5	23
2	24.5	20
3	19.5	19
4	22.5	15
5	24.5	12

Thus, based on these locations, the weight values are calculated in Shepard's method of interpolation for determining the interpolated value at unsampled points.

Several functions were used to perform different operations for our project. An init function was used to initialize the various parameters that were required during the

program execution. These include push rate, threshold, beta, and system buffers. They were globally declared for use by appropriate functions.

Another function called readFromSensor function was defined for implementing data push with a defined push rate from the sensors. Every time this function was called, a random generator function was initialized and a set of random numbers equivalent to the push rate, were generated so as to select the active sensors randomly. After this, a connection to the database was established and data from these active sensors were retrieved from the sensor table for a particular timestamp. For further processing of this sensor data, the data retrieved from these sensors were stored in system buffer, a double dimensional array. Using system buffers saved us a lot of time as processing speed was greatly increased by retrieving data from buffers.

A function called interpolation function was implemented to carry out Shepard's method of interpolation over sensor data stored in the buffer, and to check the interpolated data for desired accuracy using leave-one-out cross validation method. If the data did not satisfy the desired confidence, a call to the pull function was made to pull data from the sensors. A counter for the number of pulls was also maintained to track the pull degree and was periodically checked against the threshold limit. If the pull degree exceeded the threshold limit, the push rate parameter was incremented by beta value and a call to the readFromSensor function was made passing the push rate parameter to push the sensor data from active sensors defined by the new push rate.

Last but not the least; the pull function was implemented in our program for pulling data from a sensor by establishing a connection with the database and retrieving data for a particular sensor for the current timestamp.

CHAPTER 3

RESULTS & ANALYSIS

3.1 Experiments

The experiments were carried out with sensor data from 52 sensors that was available for different timestamps and were repeated multiple times. The locations of the sensors were used as a basic reference for determining the distances between the interpolating points and scatter points in Shepard's method of interpolation. The power parameter, p was set to 2 in Shepard's method.

For our experiments, a threshold value of 5 and beta value of 10% was considered for pull and push process respectively. These values can vary depending upon the requirements of the application.

3.2 Discussion of Results

3.2.1. Optimal push rates for different timestamps

The simulation was run for 5 different timestamps or different recordings of sensor data with an initial push rate of 10 active sensors and the optimal number of active sensors for achieving data accuracy of 95% was obtained for each timestamp of sensor data. This experiment was repeated multiple times and the graph below shows an average value of push rate plotted against their respective timestamps. The standard deviations for the values obtained have been indicated by vertical lines over their respective markers.

From the graph below, it can be seen that for the first timestamp, 46 out of 52 sensors were required to remain active for achieving sensory data quality of 95% confidence where as for the second timestamp; only 43 sensors were needed to remain active for the same level of accuracy of sensory data. The optimal push rates for the rest of the timestamps came up to 48, 49 and 50 respectively.

However, it can also be observed that quite a large number of sensors were required to remain active in the network to achieve just 95% of data accuracy. This is because the algorithm concludes upon a minimal push rate or rather stabilizes upon a minimal push rate value only after the data of all the sensors satisfy the 95% confidence level and not just few of the sensors. Since our dataset had different values for different sensors and the deviation of the interpolated data from the actual data for all sensors was not consistent or rather very different, a large number of sensors were needed for the dynamic system to converge into an optimal value to achieve the accuracy of 95% for each of the sensors. The optimal push rate can greatly vary depending upon the nature of the dataset coming from all the sensors comprising the network and also upon how strictly the requirement that the interpolation must be correct to a specified accuracy for every sensor, is followed by our dynamic system.

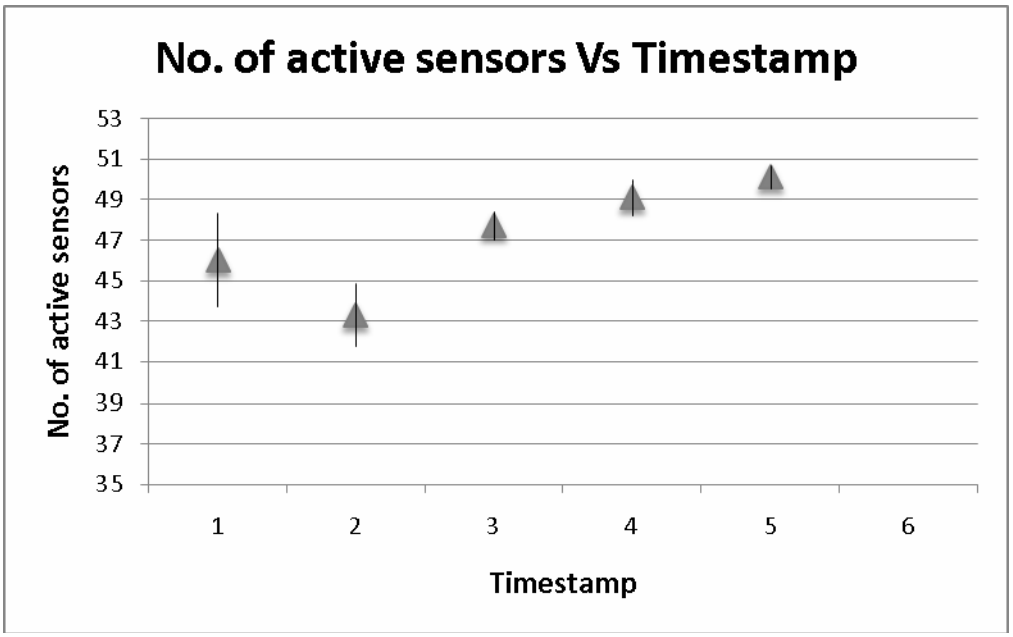

Figure 3.1 Each marker in the graph represents the optimal number of active sensors obtained for achieving data quality of 95% confidence level during each of the five different dataset recordings.

3.2.2. Accuracy of Interpolation

The graph below shows the comparison of the actual temperature data with the interpolated data of all sensors during a particular timestamp after an initial push rate of 10. Since only 10 out of 52 available sensors were used initially to send data to the proxy for effective sketching of the sensing field, the data for the rest of the sensors was obtained by interpolating with only 10 existing sensor data. Thus, it can be seen that the interpolated values of the rest 42 sensors have a pronounced deviation from their actual temperature values and hence do not confirm to a 95% confidence level or desired accuracy.

It can also be observed that the deviations of interpolated data from actual temperature data for all the sensors are not uniform. Interpolated data of most of the sensors show a pronounced deviation from actual data where as data from few of the sensors show almost negligible deviation from the actual data. Thus, our dynamic system would be required to adjust the push rate as well as perform subsequent pulls from appropriate sensors so as to improve the data accuracy of those sensors that show pronounced deviation in their interpolated data. This may inquire a large number of sensors to remain active as interpolated data must be correct to specified accuracy for every sensor and not just few of the sensors.

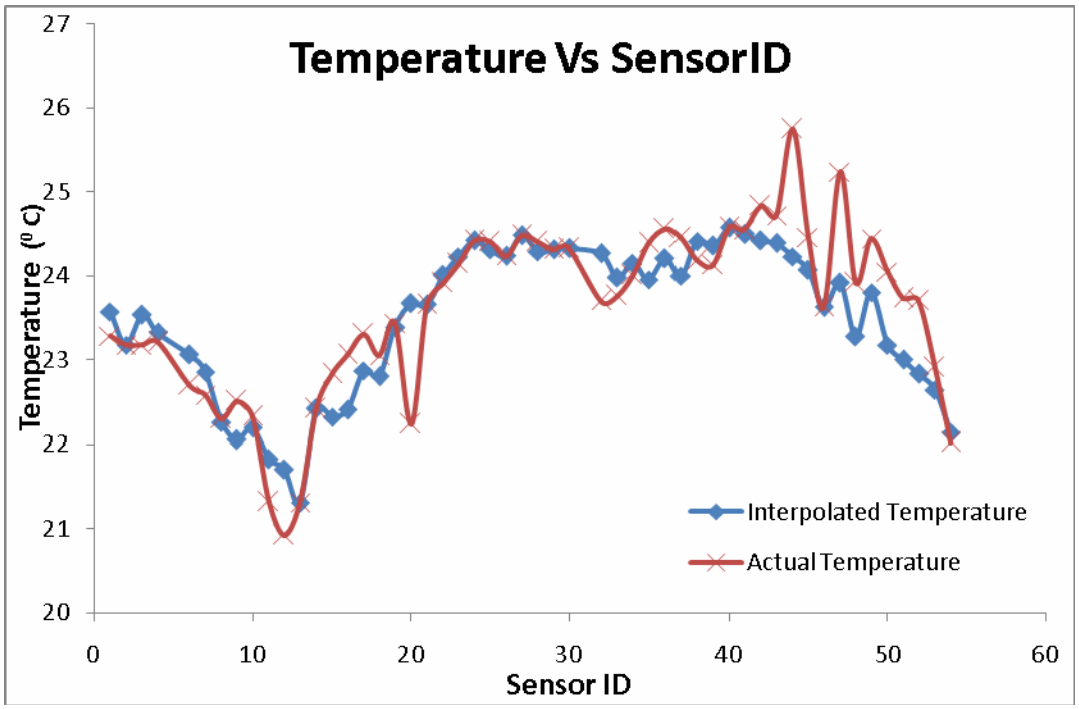

Figure 3.2 This graph shows the comparison of interpolated temperature against actual temperature data for all sensors after a push of 10 sensors data.

The other graph given below shows the comparison of the actual temperature data with the interpolated data that was obtained after having made subsequent pulls

and increase in the push rates to make the data confirm to 95% confidence level of accuracy. It can be observed that there is almost negligible deviation of interpolated data from the actual temperature sensor data, thus making the data quality high while reducing overall system energy. Thus, the interpolated data of sensors that showed a pronounced deviation in the initial stage of push rate, have been made accurate to 95% confidence after making appropriate adjustments to the push rate of the network.

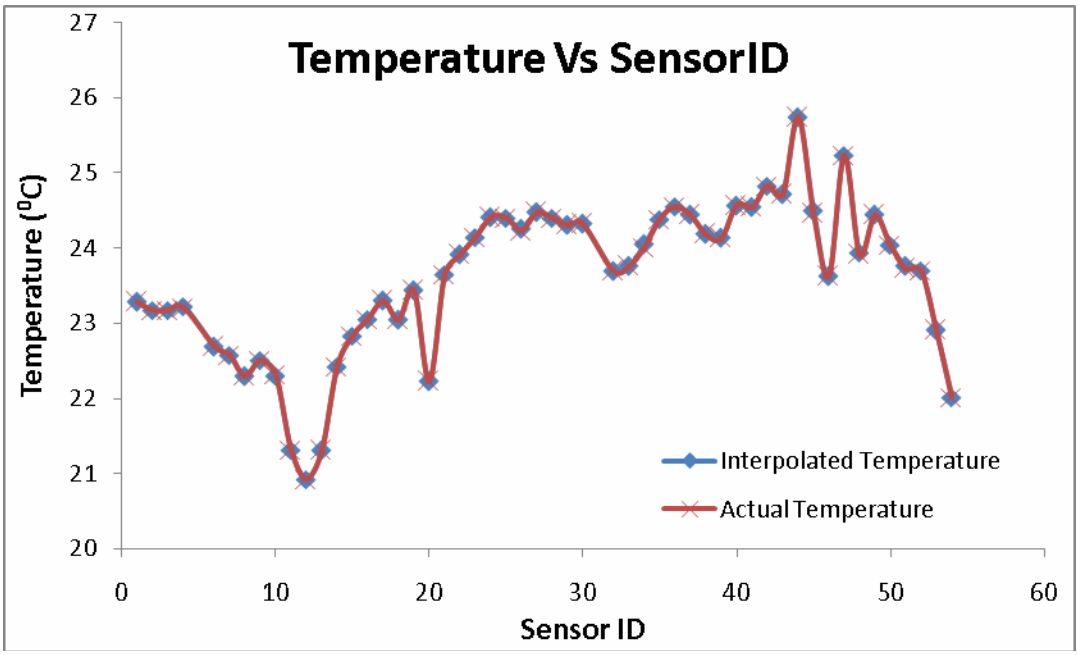

Figure 3.3 This graph shows the comparison of interpolated temperature with actual temperature data obtained for all sensors after subsequent pulls and increase in push rate that confirms to 95% accuracy.

3.2.3. Optimal push rates for varying levels of data accuracy

The experiments were conducted for different timestamps with different levels of accuracy or with different percentage of errors of interpolated data with respect to the actual temperature data. The push rates that were obtained for achieving the varying levels of accuracy for all timestamps gave us some interesting results. From the graph

given below, it can be observed that with an increase in the accuracy of the interpolated data or rather with a decrease in the percentage of error of interpolated data obtained with respect to the actual data, there was an increase in the number of active sensors or the energy level. This implies that the energy consumed by the network increases with increase in the accuracy of the interpolated data. For example, during timestamp 1, 50 out of 52 sensors were required to remain active for achieving a high data accuracy of 5% error from the actual temperature data where as only 45 sensors were required to be active for achieving interpolated data confirming to a lower accuracy of 16% error from the actual value. A similar behavior can be seen for all the other timestamps from the graph as the plot for each of the timestamps shows a decreasing behavior of energy level with a decrease in the accuracy of data.

It is quite surprising to see that quite a large number of sensors were required to remain active for achieving data accuracy of just 16% but then, when this value is compared to the minimal number of active that was required by the system to achieve 5% error or higher data accuracy; it is relatively a smaller value or rather requires relatively lesser number of sensors to remain active for achieving lower data accuracy. The number of active sensors obtained for achieving desired data accuracy can vary depending upon the nature of the sensory data.

Thus, it can be concluded that with an increase in the accuracy of interpolated data, there is a tradeoff in the energy consumption level of sensor nodes.

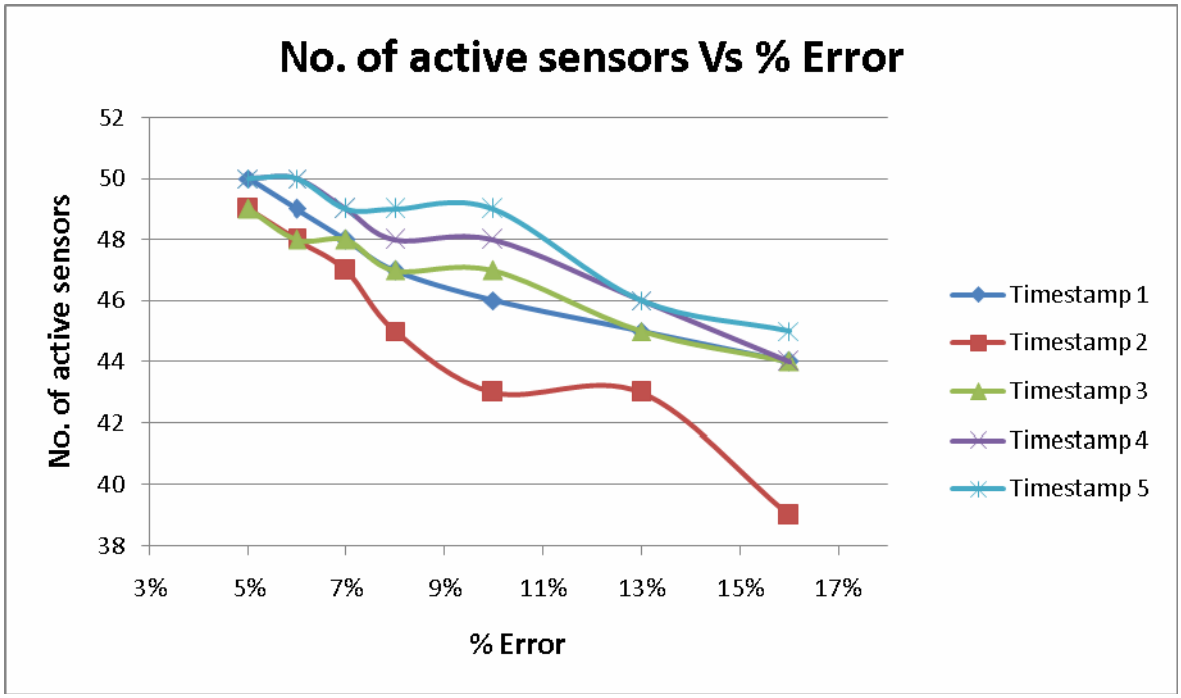

Figure 3.4 The graph above represents the energy level consumed for achieving sensor data of varying levels of accuracy for different dataset recordings.

3.2.4. Time taken for convergence to an optimal push rate

When the experiments were conducted for various timestamps to obtain an optimal number of active sensors or push rate for achieving data pertaining to a desired confidence level, the time taken by the dynamic system to converge to an optimal push rate was measured in terms of the number of rounds of adjustments that were made to the push rate starting from the initial push rate of the system, to finally arrive at an optimal value.

The graph below shows that the system made almost 7 rounds of adjustments in the push rate from an initial push rate value of 10, to arrive at a minimal push rate value i.e. 45 to attain the desired confidence of 95%. It can also be seen that, once the desired

accuracy is attained by a minimal push rate value, there are no more adjustments made to the push rate as increasing the push rate will anyways achieve the desired accuracy value of sensory data and hence, the system stabilizes by setting the optimal push rate value of the network to the minimal push rate value to achieve the desired data accuracy of sensory data. However, the system will start to adjust the push rates again once the future data set recordings start to disobey the desired accuracy of sensory data in which case, the algorithm runs again to calculate the minimal push rate for achieving desired data accuracy so as to satisfy users' demands.

Thus, it can be concluded that the time taken by the system to converge to a minimal push rate that is required to attain a desired accuracy, depends upon the threshold and beta value of the dynamic system, which can be tuned accordingly to suit different requirements of the applications.

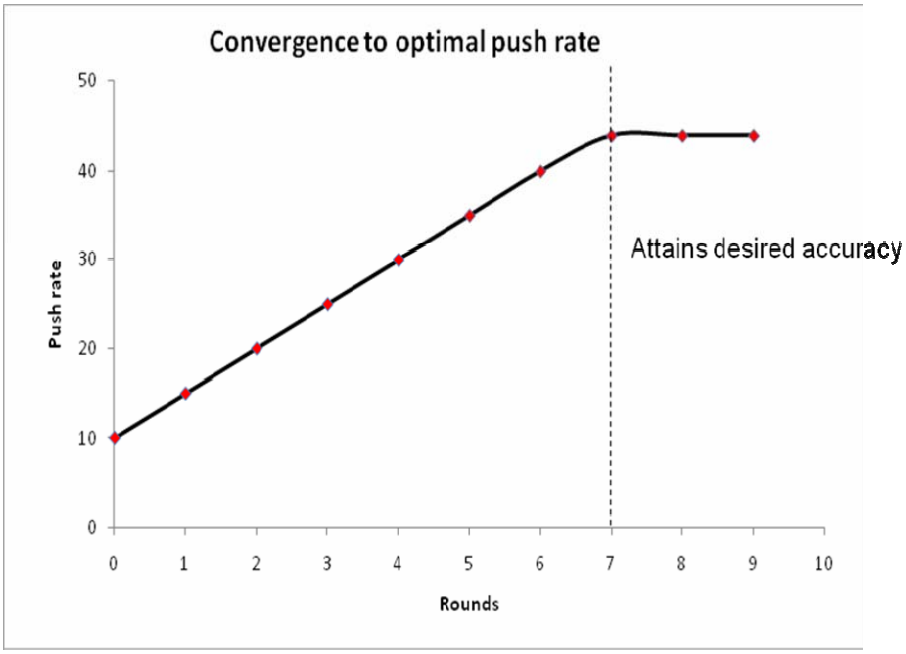

Figure 3.5 The time taken by the dynamic system, in terms of the number of rounds of adjustments made to converge to an optimal push rate value from an initial push rate of 10, has been indicated in the graph above.

CHAPTER 4

CONCLUSIONS & FUTURE WORK

Sensor Web, a new technology has been developed recently to effectively network large scale deployed heterogeneous sensors with Internet to make sensor data discoverable, accessible and controllable via World Wide Web. To integrate WSNs with the Internet, proxy is a possible architecture proposed in recent papers. But this architecture is not energy efficient as there is no control to the continuous backhauling of data from the sensors to the proxy. Since sensors are highly resource constrained, continuous transfer of data is not always feasible. After a certain amount of time of transfer, the nodes are sure to die out one by one ultimately resulting in the degradation of the service provided to the clients of the network. Thus, there needs to be a dynamic solution that solves this problem by minimizing the overall energy of sensor network while still maintaining high quality of data delivered to the users.

In this paper, dynamic push/pull service architecture is proposed where sensory quality is dynamically monitored and controlled. The architecture is implemented in a web proxy residing at the gateway of the network. To start with, an initial push rate or number of active sensors is defined for pushing data into the proxy database. Next, Shepard's method of interpolation is implemented for determining unknown values at predetermined data points using known values at sample points. The accuracy of the interpolated data is then measured using leave-one-out cross validation method which is used when there is a predictive function that is to be evaluated according to some data

set. If the accuracy of interpolated data does not match the desired accuracy, the system will dynamically pull data from the sensors until the number of pulls exceeds the threshold. On exceeding the threshold, the push rate is increased by beta and the whole process is carried on until the interpolated data matches the desired accuracy. On satisfying the desired accuracy, the sum of push rate and no_pulls parameters gives the optimal number of active sensors or push rate for achieving the desired confidence.

Simulation of web proxy was carried out multiple times with data from 52 sensors that were available for different timestamps. From the simulation results, the optimal number of active sensors or push rates can be seen for different timestamps to achieve sensory data quality of desired confidence. The interpolated data before and after the dynamic controlling of sensory data has been plotted and it can be clearly seen that the interpolated data that was obtained with initial push rate of 10 sensors has quite a pronounced deviation from the actual values when compared to the interpolated data that was obtained after dynamically controlling the push and pull rate for achieving sensory data of desired confidence. It can also be observed that the energy level or the number of active sensors tends to increase with an increase in the accuracy of interpolated data. Last but not the least, it can be seen that the time taken by the dynamic system to converge into an optimal push rate value depends upon the number of rounds of adjustments made to the push rate for arriving at a minimal push rate value that achieves the desired data accuracy of sensory data. This number of adjustments in turn depends upon the threshold and beta value of the dynamic system that can be tuned accordingly to suit different applications.

Our implementation can be extended in future to incorporate temporal characteristics of the application by interpolating unknown data of one particular sensor at a specific timestamp using data from that sensor known at different timestamps[19]. It can also be integrated with different visualization techniques for displaying data to the user.

REFERENCES

1. Mainwaring, A., et al., *Wireless sensor networks for habitat monitoring*, in *In First ACM Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*. 2002: Atlanta, GA, USA.

2. Sinem, C., Y. Sing, Cheung, and V. Pravin, *Sensor Networks for Monitoring Traffic*, in *In Allerton Conference on Communication, Control and Computing*. 2004.

3. Jaspal, S., A. Alice, and A. Adrian, *Wireless Sensor Networks for Commercial Lighting Control: Decision Making with Multi-agent Systems*, in *In AAAI Workshop on Sensor Networks*. 2004.

4. Xingchen, C. and B. Rajkumar, *Service Oriented Sensor Web* 2006.

5. Jason, H., *System Architecture for Wireless Sensor Networks*. 2003, UC Berkeley.

6. Philip, B., Gibbons, et al., *IrisNet: An Architecture for a Worldwide Sensor Web*. IEEE Pervasive Computing, 2003. **02**(4): p. 22-33.

7. Flavia, C., Delicato, et al., *A flexible web service based architecture for wireless sensor networks*, in *Proceeding of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCDSW'03)*. 2003.

8. Dunkels, A., T. Voigt, and J. Alonso, *Making TCP/IP Viable for Wireless Sensor Networks*, in *In Proceedings of the First European Workshop on Wirelss Sensor Networks (EWSN'04)*. 2004: Berlin, Germany.

9. Dunkels, A., et al., *Connecting Wireless Sensornets with TCP/IP Networks*, in *In WWIC2004*. 2004.

10. Jinglun, S. and L. Weiping. *A service-oriented model for wireless sensor networks with Internet*. 2005.

11. Chi-Fu, H. and T. Yu-Chee, *The Coverage Problem in a Wireless Sensor Network*, in *In WSNA '03: Proceedings of the 2nd ACM International conference on Wireless sensor networks and applications*. 2003, ACM Press. p. 115-121.

12. Xiang-Yang, L., P.-J. Wan, and O. Frieder, *Coverage in wireless ad-hoc sensor networks.* IEEE Transactions on Computers, 2003. **52**(6): p. 753-763.

13. Fan, Y., et al., *A robust energy conserving protocol for long-lived sensor networks*, in *In ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. 2003, IEEE Computer Society. p. 28.

14. Di, T. and D. Nicolas, Georganas, *A coverage-preserving node scheduling scheme for large wireless sensor networks*, in *In WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. 2002, ACM Press. p. 32-41.

15. Deepak, G., et al., *Highly Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks.* ACM SIGMOBILE Mobile Computing and Communications Review, 2001. **5**(4): p. 11-25.

16. Ayad, S. and S. Loren, *Power Aware Metrics for Wireless Sensor Networks.* International Journal of Computers and Applications, 2004. **26**(4).

17. Donald, S. *A two-dimensional interpolation function for irregularly-spaced data*. in *Proceedings of the 1968 23rd ACM national conference*. 1968. New York, NY, USA.

18. Peter, R. and L. Lixin, *Representation and querying of interpolation data in constraint databases*, in *Proceedings of the 2002 annual national conference on*

Digital government research. 2002, Digital Government Research Center: Los Angeles, California. p. 1-4.

19. Tynan, R., et al., *Interpolation for Wireless Sensor Network Power Management*, in *International Workshop on Wireless and Sensor Networks (WSNET-05)*. 2005, IEEE Press: Oslo, Norway.

20. Erik, M., *A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing.* Proceedings of the IEEE, 2002. **90**(3): p. 319-342.

21. Lixin, L. and R. Peter, *The relationship among GIS-oriented spatiotemporal databases*, in *Proceedings of the 2003 annual national conference on Digital government research*. 2003, Digital Government Reseach Center: Boston, MA. p. 1-4.

22. Deepak, G., et al., *Coping with irregular spatio-temporal sampling in sensor networks.* ACM SIGCOMM Computer Communication Review, 2004. **34**(1): p. 125-130.

23. Ferenc, S. *GIS Functions - Interpolation*. [cited; Available from: http://www.agt.bme.hu/public_e/funcint/funcint.html.

24. Nigel, M.W. *UNIT 40- Spatial Interpolation I*. [cited; Available from: http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u40.html.

25. Chin-Shung, Y., et al., *Twelve Different Interpolation Methods: A Case Study of Surfer 8.0*, in *XXth ISPRS Congress*. 2004: Istanbul, Turkey.

26. Nina, S.-N.L., *Spatial Interpolation Methods: A Review.* The American Cartographer, 1983. **10**(2): p. 129-149.

27. Fred, C., Collins and V. Paul, Bolstad, *A Comparison of Spatial Interpolation Techniques in Temperature Estimation*, in *Proceedings CD rom. Third*

International Conference/Workshop on Integrating GIS and Environmental Modeling. 1996: Santa Fe, New Mexico.

28. Dubrule, O., *Two methods with different objectives: Splines and kriging.* Mathematical Geology, 1983. **15**(2): p. 245-257.

29. Mehdi, S. and S. Cyrus, *Supporting Spatial Aggregation in Sensor Network Databases*, in *In 12th Annual ACM International Workshop on Geographic Information Systems*. 2004. p. 166-175.

30. Jaroslav, H., C. Tomas, and S. Marcel, *Optimisation of Interpolation Parameters Using a Cross-validation*.

31. Peter, B., et al. *Intel Lab Data*. 2004 [cited; Available from: http://berkeley.intel-research.net/labdata/.

BIOGRAPHICAL INFORMATION

Rohita Mohan received her bachelor degree honors in computer science from Birla Institute of Technology and Science, Pilani, India in 2005. She then pursued master's degree in computer science at University of Texas at Arlington. Her research interests are mainly in wireless networking area. She also has a lot of industry experience in software programming languages as well as web designing technology.