Bloom filters for data aggregation and discovery: a hierarchical clustering approach

[#] Peter Hebden, Adrian R. Pearce
*N*ICTA Victoria Laboratory
Department of Computer Science and Software Engineering, University of Melbourne
Melbourne, Australia, {phebden,pearce}@cs.mu.oz.au

Abstract

This work addresses the data aggregation problem by tightly integrating the processes of building and maintaining distributed data structures while optimally serving queries over the network. Importantly, the approach goes beyond existing clustering techniques by addressing the incremental problem at the same time as the query problem.

1. Introduction

Wireless Sensor Network (WSN) design poses a fundamental question. Given thousands of power constrained data sources and one data sink, what is the best way to implement in-network processing such that communication costs are minimised? In this paper we present techniques for data aggregation and data discovery.

We propose a form of hierarchical clustering to facilitate data aggregation and reporting. In our framework, the nodes in a WSN are organised into a hierarchy of clusters. Clusterheads summarise and forward data up the hierarchy to an application and guide queries down the hierarchy to an appropriate data source. Bloom filters are integrated with hierarchical clusters so that queries may be served more efficiently.

In Section 2 we introduce Bloom filters with a theoretical description and an exemplary Internet application. In Section 3 we provide some details about WSNs in terms of benefits and challenges. In Section 4 we discuss data aggregation. In Section 5 we discuss hierarchical clustering, and present a low cost protocol for hierarchical clustering. In Section 6 we discuss data discovery and how Bloom filters may be used to help locate data of interest. Finally we present preliminary results and our conclusions.

2. Bloom Filters

Bloom filters were introduced in 1970 when computer memory was extremely scarce [14]. They have be used in many database and Internet applications. While not nearly as resource constrained as WSNs, their resources were finite and optimising space-time tradeoffs with Bloom filters was practical. We believe that Bloom filters are even more valuable in the WSN domain than in less resource constrained domains such as the Internet. In fact, Bloom filters should be considered as an essential and pervasive component in WSN design

A. The Bloom Filter Data Structure

A Bloom filter is a space efficient, randomised data structure for representing a set and supporting set membership queries [14]. It is a *filter* in the sense that it can be used to *filter out* an element that does not belong to the set. The space efficiency benefit is gained at the cost of a loss of information and a low rate of false positives.

In conventional hash coding, the hash area is organised into an array cells, each of which may hold an element. Each element is mapped to an array index by a hash function. But if that cell is not empty, then there is a collision and one or more attempts will be required to find an empty cell. When an empty cell is found, the element is stored there.

If the application can tolerate a small percent of false positives, then there is no need to store the element itself and, consequently, there is no need to find empty cells or explicitly resolve collisions. The collision problem is reduced by using multiple hash functions to hash each element to multiple locations in the filter.

To construct a filter, the following procedure is used. Suppose n elements in set S from a large universe U, and we have m bits of memory available. The hash area may be defined as m individual bits with indices 0 through m - 1. Generate k different indices with k different hash functions applied to each element, and set those bits to 1. To test an unknown element for set membership, follow the same procedure except just compare each of those k bits to 1. If all are set to 1, then the element probably is a member, else it definitely is not a member.

Usually we want to minimise the rate of false positives. Given n and m, what is the optimal number k of hash functions? An optimal k turns on 50% of the bits when the filter is built [15].

$$k = \frac{m}{n}\ln 2 \tag{1}$$

The error rate depends on m, k and n. Let p equal the probability of a bit being zero after the filter has been built. Let f equal the probability of a false positive when the filter is being tested.

$$p = \left(1 - \frac{1}{m}\right)^{kn} \tag{2}$$

$$f = (1-p)^k \tag{3}$$

Before providing an example Internet application, it is worth considering some of the properties of Bloom filters.

- No false negatives.
- The false positive rate may be made arbitrarily small by using more memory.
- The space requirement does not depend on the size of the element being stored.
- Lossy: not possible to reconstruct the stored set S because $|U| >> |S|$.
- Bitwise operations on Bloom filters are extremely efficient.

B. *Internet Application*

Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol by Fan (1998 and 2000) is one of the most cited papers on using Bloom filters in the Internet context [17]. The authors noted that caching has been recognised as one of the most important techniques to reduce Internet traffic, and that caching within Web proxies is very effective. Here a *proxy* is an server that stands in for another server and appears to perform the first server's functions. A server may have many proxies that serve its web pages to clients.

As of 1998, sharing caches was not widely deployed due to the overhead of existing protocols. In [17] the authors describe a web cache sharing protocol called *Summary Cache*. Essentially, each proxy stores a summary of the cache directory of each proxy in the group, and checks each summary for a *hit* before sending a query to that proxy. A Bloom filter was used to represent each proxy's cache directory - referred to as a summary cache. Each directory entry could be represented with just 8 bits. This reduced storage requirements and communication overhead. Summary Cache reduced the number of protocol messages by a factor of 25, reduced bandwidth consumption by over 50%, and reduced CPU overhead by from 30% to 95% [17].

In WSNs, clusterheads may play a role similar to web proxies. As we will explain in Section 6, they could store some data locally and use Bloom filters to determine where to forward a query.

3. Wireless Sensor Networks

Radio communication is by far the most power intensive task, and this cost rises dramatically with distance. Each bit transmitted may consume about as much power as executing 3000 instructions. For example, the energy cost of transmitting 1Kb a distance of 100 meters is approximately 3 joules. With the same amount of energy, a general-purpose processor with 100MIPS/W power could execute 3 million instructions. [9] Communication costs may be reduced by processing and storing data locally, using multihop routing and coding - all done with an an eye to facilitating data aggregation, discovery and delivery.

When WSNs are deployed in a physical environment, researchers discover that radio communication is far less reliable than lab simulations would suggest [13]. In general the output power required to transmit over distance d is proportional to d^n. The exponent n varies from about 2 to 4, and is closer to four for low lying antennae and near ground channels [9].

A. *Application Domain*

We are interested in monitoring the environment in order to enable automated irrigation control. In an agricultural setting, node density is expected be lower than for many other applications. On average, the sensors are likely to be 10s of meters apart and each sensor's radio range would normally be limited to adjacent sensors. Because communication costs increase so dramatically with distance, our target application should use even more efficient communication protocols than a dense WSN where sensors are only 2-3 meters apart.

WSNs have been deployed to monitor the conditions (temperature, soil, moisture, light, and humidity) of large vineyards. The goals included precision harvesting, watering, fertilizing, delivering pesticides, frost protection, predicting insect/pest/fungi development and developing new agricultural models [11]. WSNs should be able to provide much more than the average temperature over a wide area [12].

B. *Design of WSNs*

Design may be influenced by many factors including fault tolerance, scalability, production costs, operating environment, sensor network topology, hardware constraints, transmission media, and power consumption [8]. For many applications, the most critical factor is power consumption. Even if some nodes in a network are able to harvest energy from their environment, other nodes may not be able to due to their location, or the additional financial cost of components such as solar panels.

The fundamental challenge is to reduce power consumption by minimising communication costs. This will help to extend the operational life span of battery powered sensors, reduce congestion, and expedite queries.

Early research projects defined a WSN as a large scale, ad hoc, multihop, unpartitioned network of largely homogeneous, low cost, randomly deployed sensor nodes. Some projects assumed that nodes would not even have unique addresses or IDs within their hardware. However, by late 2004 many systems were being built using a variety of devices. Each device may differ in terms of its sensors, processing power, power source, routing ability, location awareness, and so on [10]. The degree of heterogeneity influences the complexity of software running on the sensors and other devices in the hierarchy. We believe that a heterogeneous set of devices, especially where clusterheads do not rely solely on batteries, is needed to build a robust and cost effective network.

4. Data Aggregation

Communication costs may be greatly reduced by data aggregation in a hierarchical framework. "The huge number of sensing nodes may congest the network with information. To solve this problem, some sensors such as the clusterheads can aggregate the data, do some computation (e.g., max, min, mean, median, summation, etc.), and then broadcast the summarised new information." [7] The obvious framework for aggregation is a hierarchy. Group similar data points together, summarise random samples taken from each group, and transmit a report. If less specific data is required, then the reports from a number of groups may be summarised, and so on. Higher and higher levels in the hierarchy maintain summaries of more and more groups. This structure may be built using a hierarchical clustering algorithm. However, clustering sensor nodes is not the same as clustering a traditional data set.

A. Traditional Clustering

Clustering is generally considered to be the unsupervised classification of unlabeled patterns (data points) into groups. Clustering techniques fall in two broad categories: hierarchical and partitional. Hierarchical clustering produces a nested series of partitions - at each time step the closest nodes are merged. This process continues until all nodes have been merged into one top level cluster. K-means is an example of a partitional technique. It partitions the data set (all nodes) into k clusters. It uses iterative relocation of cluster centroids to reduce within cluster variance, and is in the class of iterative optimisation procedures [1], [2].

B. Related Work on Clustering Sensors

Clustering in general has been studied extensively and the hierarchical clustering of sensor nodes has been explored by a number of researchers. In [5] the authors point out that many clustering algorithms have been proposed, but none aim at minimising the energy spent by the system to gather data from the network. They also suggest that an algorithm that generates the minimum number of clusters might not ensure minimum energy usage.

1) Topological Awareness: The goal of the clustering algorithm proposed in [6] is to maximise network lifetime. However, it assumes that each node is aware of the whole network topology. This is not feasible for large WSNs. Other clustering algorithms were designed to generate stable clusters in environments with mobile nodes [5]. In our case, the sensors have fixed location and the instability of clusters due to mobility of sensors is not an issue.

The ideal cluster hierarchy uses minimal energy to communicate information from data sources to the local base station. Bandyopadhyay and Coyle proposed a fast, randomised, distributed algorithm for organising sensors into a hierarchy of clusters. Their objective was to minimise the energy spent during operation, i.e. in communicating the information to the information processing center [5].

The ideal algorithm organises sensors using minimal energy where each node has minimal topological awareness. Even so, in some domains we will care more about optimising cluster formation than operation.

2) One Level Cluster Formation: Heinzelman, et al, proposed a distributed algorithm where sensors elect themselves as clusterheads with some probability and broadcast their decisions. Unclustered sensors bind to the clusterhead that requires minimum communication energy. However, their algorithm allows only 1-hop clusters to be formed, it is not clear how to compute the optimal number of clusterheads, and the clusters are all on one level. The algorithm is rerun periodically so that none of the sensors are overloaded due to performing clusterhead tasks [4].

3) Optimal Number of Clusters: Bandyopadhyay and Coyle assume a network of simple sensors that transmit at a fixed power level. Their sensors forward data up to k hops when necessary. Their algorithm generates one or more levels of clusters with an optimal number of clusterheads on each level [5].

In order to calculate the optimal number of clusterheads, they defined a function for the energy used by an operational network and found values for the parameters p and k that minimise it. p is the probability of a sensor volunteering to be a clusterhead and k is the maximum number of hops that a clusterhead's advertisement will be forwarded [5].

C. Association Criteria

Just as in [4], our unclustered sensors elect themselves as clusterheads with some probability and broadcast their decisions by emitting a beacon. However, our sensors emit short range beacons, self elect and broadcast over time, and unlike [4], our sensors do not join the closest cluster in terms of the energy required to reach it. They associate with the first clusterhead they hear.

In our system, the range of a clusterhead's beacon is short, and the sensor's have only local knowledge of the network, so binding to the first heard clusterhead is the simplest and least costly decision making process. Further, given the beacon's limited range, the sensor's cost of transmitting to the clusterhead is low by definition.

D. Transmission Range

In [5] they "assume a multi-hop network of very simple sensors that transmit at a fixed power level; data exchanged between sensors not within each other's radio range is forwarded by other sensors in the network." In contrast, we assume that when a sensor acts as a clusterhead, it also has the capability to adjust the power level of its beacon. It can directly recruit sensors up to d distance away. This capability enables a clusterhead to control the number of sensors in its cluster.

5. Hierarchical Clustering

We propose a framework for data aggregation and discovery that aims to minimise both the cost of building hierarchical clusters and the cost of their operation. The algorithm proposed in this paper is similar to the ones proposed in [4] and [5].

A. Design Goals for Stochastic Hierarchical Clustering

This algorithm was designed to address the dynamic and complex nature of an environmental WSN. Nodes will be added and removed from the network over time. It is very likely that some nodes, clusterheads in particular, will fail due to physical damage or power loss. So at least parts of the network will have to be reclustered periodically. We expect that in some applications the cost of building and rebuilding will dominate.

The network it will be composed of a variety of devices. Some will be solar powered and able to serve as clusterheads. Others will be battery powered sensors for measuring a combination of physical phenomena. Clusters may be formed based on a variety of distance metrics. Sensors could form associations based not on the first beacon heard, but on a combination of attributes such as euclidean distance, temperature, and moisture level. In that case, multiple hierarchical clusters would coexist and a richer variety of queries could be served. However, such complex distance metrics are likely to result in less compact clusters and, therefore, higher communication costs.

So, given the dynamic and complex nature of the network, it is extremely important to minimise the time and energy cost of hierarchical clustering and reclustering. In the following description, we will use the *first heard* criterion as a proxy for *closest*.

B. Stochastic Hierarchical Clustering

Clustering causes each node in the network to discover the best clusterhead to communicate with. Sensors bind to clusterheads, which bind to other clusterheads. At each time step, suitable nodes will volunteer to be a clusterhead with a probability of p. New clusterheads will transmit a beacon that can reach sensors up to k hops away.

A sensor that has not yet associated with a clusterhead is referred to as a *free sensor*. A sensor reports to its clusterhead, and each clusterhead reports to a clusterhead at a higher level - although the final parent child relationship may not be determined until after the last cluster has formed. A clusterhead's primary tasks are to summarise and forward messages up and down the hierarchy.

The pseudo code in Figure 1 provides a high level description of this distributed, asynchronous algorithm from the point of view of the local base station.

Individual sensors and their data are at level 0. A clusterhead is at level 1 when it aggregates data reported by sensors that are bound to it. A clusterhead is at level $h+1$ when it aggregates its own summary with summaries reported by clusterheads from levels 1 to level h.

It is not necessary for all level h clusters to form before level $h+1$ clusters. At each time step during cluster formation, a small fraction of random free sensors wake up, decide they will function as clusterhead C_x, and transmit a beacon that travels distance d. Free sensors within distance d hear the beacon and bind to C_x. If a bound sensor hears C_x's beacon, then that sensor asks its clusterhead C_{id} to bind to clusterhead C_x. If C_{id} agrees, it notifies C_x.

Consider Figure 2 and suppose non overlapping clusters 1 and 2 have formed and have several members each. They are level 1 clusters, i.e. their clusterheads are not bound to any other clusterhead. Next, clusterhead C_3 starts to form cluster

```
LBS: local base station;
LBS transmits command to organise;

While LBS detects that percent of sensors
  free is > maxFree
{
  P free sensors wake up and transmit a
    clusterhead beacon;
  Free sensors that hear a clusterhead
    beacon bind to its source;
  Clusterheads update their Bloom filters;
  Bound sensors that hear a beacon ask
    their clusterhead to bind to the
    beacon’s source;
      Clusterheads will refuse if already
        bound to another clusterhead;
}

LBS transmits message to determine parent
child relationships between clusterheads
{
  Clusterheads forward parent message;
  Clusterheads update parent child pointers;
  Bloom filters percolate up from bottom;
}
```

Fig. 1: Pseudo code for the stochastic hierarchical clustering algorithm. Distributed, asynchronous actions and events are described from the point of view of the local base station.

3. Neighboring free sensors bind to it. However, if one of the cluster 1 and/or 2 sensors also hears C_3's beacon, they will ask their clusterhead to report to C_3. Clusters 1, 2, and 3 comprise a level 2 cluster and C_3 will aggregate its own summary with the summaries from C_1 and C_2.

As shown in Figure 2, sensors bind to one clusterhead, and each clusterhead binds to a higher level clusterhead if one is created nearby. The edges indicate that C_1 and C_2 are bound to C_3.

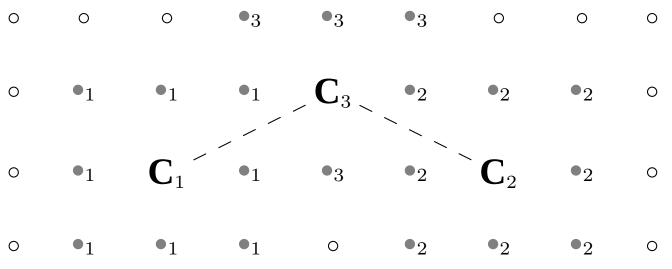

Fig. 2: Field of sensors during cluster building. Nodes 1,2, and 3 volunteered in order. Sensors are bound to clusterheads 1, 2, and 3. Clusterheads 1 and 2 are bound to clusterhead 3.

The clustering process continues until enough sensors are bound to a clusterhead. We only need to collect data from a representative number of sensors. Given a sufficiently redundant network, it is not essential that all sensors are able to report all the time. Nor is it essential that all clusterheads bind to another clusterhead until there is only one at the top level.

After enough sensors are bound, each top level clusterhead transmits a beacon that descends the hierarchy - some clusterheads will need to rebind to the clusterhead on the shortest path to the top. Remaining free sensors, if any, will go into hibernation until reclustering. Figure 1 shows the pseudo code for this clustering algorithm.

C. Recalculating Centroids

The form of hierarchical clustering proposed in this paper is different from traditional clustering. Perhaps most importantly, the hierarchy of clusters are not necessarily formed to identify meaningful patterns in the data. In addition, centroids (clusterheads) are not recalculated during the clustering process. Recalculating centroids during clustering would not be cost effective since all we really want to do is build hierarchical clusters using minimal energy. We do not refer to clusterheads as centroids. Each clusterhead is a stationary physical device, not an abstraction of a subset of data points.

We do not recalculate to find the most central clusterhead for the following reasons. The network is dynamic - nodes are added and deleted over time - and the hierarchy will need to be reclustered periodically, so the cost of clustering, and the cost of recalculating *centroids*, can not be amortised over a long period of time. In addition, we may need to build multiple coexisting hierarchies, so the cost of building is multiplied.

D. Cluster Quality

The clusters are formed for the efficient delivery of aggregated data to the local base station. The network may be queried on any level from 0 to N. Bloom filters may be used to efficiently route specific queries to an appropriate data source (Section 6).

"In sensor networks, two scenarios for data dissemination exist: query driven and continuous update." [7] We envision a query driven network, so the network must be able to serve queries. In general, there are two types of addressing in sensor networks: data centric and address centric. In data centric, the query is sent to specific region in the network. In address centric, the query is sent to an individual node [7]. A network should be designed to facilitate both kinds of queries.

E. Cluster Fault Tolerance

It is inevitable that some clusterheads will fail due to physical damage, power loss, or some other hardware failure. When a clusterhead fails, the sensors bound to it will gradually become aware that their clusterhead no longer transmits. After a period of time, a qualified sensor will volunteer, assume the previous clusterhead's ID, emit a beacon, and the sensors will bind to it. Over time the WSN will be able to report less and less detailed information about its environment, but this degradation will be graceful.

6. Data Discovery in Hierarchical Clusters

At this point we are assuming that the WSN has self-organised into a hierarchy of clusters, and that during the clustering process elements, such as sensor IDs and measurements, have been stored in Bloom filters in each node. Naturally we would like to exploit the hierarchical structure and the information stored at each node to route queries to appropriate data sources. Bloom filters provide an extremely space efficient way guide a query down the hierarchy.

A. Resource Routing

In this section we assume that clusterheads are location aware, but ordinary sensors are not - they only know which clusterhead to report to.

The general framework for resource routing protocols was described in [18]. Their framework assumes a tree topology with a root node and child nodes that hold resources. Resource requests descend the tree. Each node maintains a list of resources it holds and that exist in any of its descendants. Upon receiving a request, a node can determine if it holds the resource, and if not, then where to forward the request. The same algorithm could be applied to directing a query down a cluster hierarchy to a sensor with a specific identifier.

To find a resource, we must name it. To find a specific sensor, we must first know its ID. Of course maintaining lists of names and/or attributes requires storage and communication, and can consume large amounts of memory and power. Since all we really need to do is to check for set membership, we can use Bloom filters.

If clusterhead maintains a set of Bloom filters. One represents all sensors that report to it, and the others represent data that may be found in its children, then it can easily determine whether data is probably in its subtree without transmitting a query. However, suppose we want data from sensors in region R that measured moisture M, temperature T, and salinity S. Such a tuple's format could be represented by a string: *RMTS*. Assuming a clusterhead maintained a Bloom filter where such tuples where stored, it could check whether any of its sensors or any sensors in its subtree collected the requested data.

Naturally the application that makes queries and the nodes of the WSN must agree on a protocol that defines how measurements are hashed to Bloom filters. Suppose we want to determine which sensors in region R have detected moisture levels M ranging from 40 to 49, temperatures T from 20 to 29, and salinity S equal to 15. First it could be agreed that measurements are binned into deciles before hashing. So this tuple would have been stored in a filter as *R402015*. If the query tests positive at a clusterhead, then such data is probably in the clusterhead's subtree.

Bloom filters are well suited for data discovery in a hierarchy because two Bloom filters may be ORed to represent the union of two sets of data. The union can be stored in the root node of a subtree. If the root's filter tests positive, then each child node should be tested until one tests positive. If one does test positive, then search that branch. The descent down a branch stops when the target is found or when a filter tests negative. Filters are less saturated at lower levels so the probability of a false positive decreases as the descent proceeds [19], and compression may be cost effective [16]. In other words, filters can be utilised to perform a probabilistic depth first search of a hierarchical structure.

This technique was used to implement *attenuated Bloom filters* in the P2P domain [20]. Their goal was to ensure that given a file request and a nearby replica, the shortest path to its server could be determined quickly. Something very similar to attenuated Bloom filters could be used in our WSN to determine whether or not a data source exists within d hops from a clusterhead.

7. Preliminary Results

We explored WSN designs that utilise Bloom filters and where sensors associate based on different criteria.

Reclustering is integral to self organisation and enables the system to adapt to its environment. During reclustering, the network should be able to draw upon past experience to regenerate clusters that will service queries more effectively. Ideally, with experience, network performance will improve over time because it learns the relative importance of each feature.

In order to test our approach, we have built a WSN simulator as shown in Figure 3. The simulator is presently being utilised to explore the optimality of our data aggregation and discovery algorithm. Preliminary results indicate that a simple, low cost clustering algorithm can be used to generate a near optimal number of clusterheads, and that Bloom filters are cost effective for data discovery in WSNs. Further, our studies indicate that for optimum data aggregation and discovery, cluster formation must take into account optimisation with respect to Bloom filter performance in addition to clustering performance.

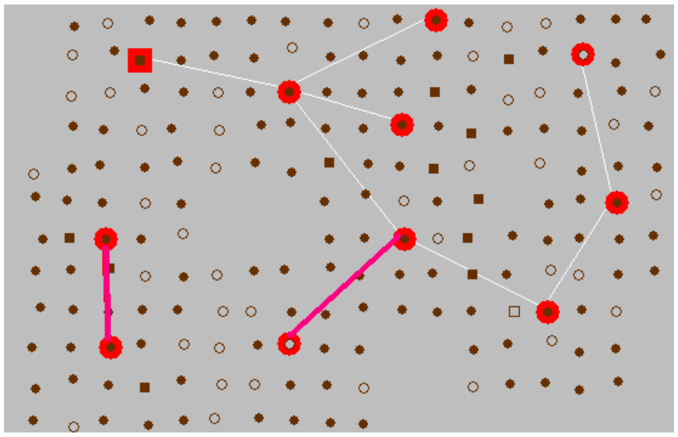

Fig. 3: We are utilising the WSN simulator shown above to explore data aggregation and discovery operations. Sensors are in various states and are represented by small circles and squares. The clusterheads are larger. Each sensor is bound to one clusterhead (no edges shown), which is bound to at most one parent clusterhead (thin edges). A bold edge binds a clusterhead to a top level clusterhead. The path from a sensor to a higher level follows each clusterhead's parent pointer.

8. Conclusion

We have presented a new approach to data aggregation and discovery for WSNs based on the integration of hierarchical clustering and Bloom filters. We believe it is extremely important to minimise the cost of hierarchical clustering and reclustering. Using the *first heard* criterion as a proxy for *closest* during clustering is a simple and cost effective computational process. In addition, using a variety of distance metrics we can build multiple coexisting hierarchies. Data discovery is the other half of the problem. Bloom filters provide a cost effective way to facilitate the discovery of appropriate data sources.

References

[1] A.K. Jain, M.N. Murty and P.J. Flynn, ”Data Clustering: A Review”, *ACM Comp. Surv*, 1999.

[2] R. O. Duda, P. E. Hart, & D.G. Stork, Pattern Classification, Second Edition., New York, NY: John Wiley and Sons, 2001.

[3] E. Cayirci, ”Data aggregation and dilution by modulus addressing in wireless sensor networks”, *IEEE Communications Letters*, vol.7, no.8, Aug. 2003, pp.355-7.

[4] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, ”Energy-Efficient Communication Protocol for Wireless Microsensor Networks”, *in Proceedings of IEEE HICSS*, January 2000.

[5] S. Bandyopadhyay, E. Coyle, ”Minimizing communication costs in hierarchically-clustered networks of wireless sensors”, *Computer Networks*, 44(1): 1-16 (2004)

[6] C.F. Chiasserini, I. Chlamtac, P. Monti and A. Nucci, ”Energy Efficient Design of Wireless Ad Hoc Networks”, *in Proceedings of European Wireless*, February 2002.

[7] M. Tubaishat and S. Madria. ”Sensor networks: an overview”, *IEEE Potentials*, Volume: 22 Issue: 2, Page(s): 20 -23, April-May 2003.

[8] F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, ”A survey on sensor networks” *IEEE Communications Magazine*, 40(8):102-114, August 2002.

[9] G. J. Pottie and W. J. Kaiser, ”Wireless Integrated Network Sensors”, *Communications of the ACM*, vol. 43(5), May 2000, pp. 51-58.

[10] K. Romer and F. Mattern, ”The Design of Wireless Sensor Networks”, *IEEE Wireless Communications*, December 2004

[11] Intel, ”New Computing Frontiers - The Wireless Vineyard”, http://www.intel.com/technology/techresearch/research/rs01031.htm

[12] M. Baard, ”Making Wines Finer With Wireless”, *wired.com*, http://www.wired.com/news/wireless/0,1382,58312,00.html

[13] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. ”Lessons from a sensor network expedition”, *In First European Workshop on Wireless Sensor Networks* (EWSN ’04), January 2004.

[14] B. Bloom, ”Space/Time Tradeoffs in Hash Coding with Allowable Errors”, *CACM*, 13(7):422-426, July 1970.

[15] A. Broder and M. Mitzenmacher, ”Network Applications of Bloom Filters: A Survey”, *Internet Mathematics* Vol. I, No. 4: pages 485-509, 2004

[16] M. Mitzenmacher, ”Compressed Bloom Filters”, *IEEE/ACM Transactions on Networks*, 10:5, pp. 613-620, October 2002.

[17] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. ”Summary Cache: a scalable wide-area Web cache sharing protocol”, *IEEE/AACM Transactions on Networking*, 8(3):281-293, 2000.

[18] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, Randy H. Katz ”An architecture for a secure service discovery service”, *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, August 1999

[19] Ledlie, Taylor, Serban, and Seltzer. ”Self-Organization in Peer-to-Peer Systems”, *Tenth ACM SIGOPS European Workshop*, September 2002

[20] S.C. Rhea, J. Kubiatowicz, ”Probabilistic Location and Routing”, *Proceedings of INFOCOM 2002*