Building an Efficient P2P Overlay for Energy-Level Queries in Sensor Networks

S. Sioutas
Department of Informatics
Ionian University
49100 Corfu, Greece
sioutas@ionio.gr

K. Oikonomou
Department of Informatics
Ionian University
49100 Corfu, Greece
okon@ionio.gr

G. Papaloukopoulos
Department of Computer Engineering and Informatics
University of Patras
26500 Patras, Greece
papalukg@ceid.upatras.gr

M. Xenos
School of Sciences and Technology
Hellenic Open University
26500 Patras, Greece
xenos@eap.gr

Y. Manolopoulos
Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
manolopo@csd.auth.gr

ABSTRACT

After the debunking of some myths about why P2P overlays are not feasible in sensornets, many such solutions have been proposed. None of the existing P2P overlays for sensornets provide "Energy-Level Application and Services". On this purpose and based on the efficient P2P method presented in [16], we design a novel P2P overlay for Energy Level discovery in a sensornet, the so-called ELDT (Energy Level Distributed Tree). Sensor nodes are mapped to peers based on their energy level. As the energy levels change, the sensor nodes would have to move from one peer to another and this oparation is the most crucial for the efficient scalability of the proposed system. Similarly, as the energy level of a sensor node becomes extremelly low, that node may want to forward it's task to another node with the desired energy level. The adaptation of the P2P index presented in [16] quarantees the best-known query performance of the above operation. We experimentally verify this performance via an appropriate simulator we have designed for this purpose.

Categories and Subject Descriptors

H.2 [**Database Management**]: [Emergent Systems]

General Terms

Algorithms, Data Structures and Indexing, Networks

Keywords

Peer-to-Peer Overlays, Sensor Networks

MEDES 2009 October 27-30, 2009, Lyon, France

1. INTRODUCTION

In the last years sensornet research primarily focused on data collection, finding applications in ecology (e.g., environmental and habitat monitoring [12]), in precision agriculture (e.g., monitoring of temperature and humidity), in civil engineering (e.g., monitoring stress levels of buildings under earthquake simulations), in military and surveillance (e.g., tracking of an intruder [6]), in aerospace industry (e.g., fairing of cargo in a rocket), etc.

P2P query processing in sensor networks is a new concept. Traditionally, sensors are used as data gathering instruments, which continuously feed a central base station database. The queries are executed in this centralized base station database which continuously collates the data. However, given the current trends (increase in numbers of sensors, together collecting gigabits of data, increase in processing power at sensors) it is not anymore feasible to use a centralized solution for querying the sensor networks. Therefore, there is a need for establishing an efficient access structure on sensor networks in order to contact only the relevant nodes for the execution of a query and hence achieve minimal energy consumption, minimal response time, and an accurate response. We achieve these goals with our peer-to-peer query processing model on top of a distributed index structure on wireless sensor networks.

In sensor networks any node should be able to introduce a query to the system. For example, in the context of a fire evacuation scenario a firefighter should be able to query a nearby sensor node for the closest exit where safe paths exist. Therefore, a peer-to-peer query processing model is required. A first P2P program for spatial query execution presented in [7]. In the context of "energy-level discovery", assuming that a sensor is responsible for executing some program task but unfortunately it's energy-level is lower than a pre-defined threshold. Then, this sensor should be able to introduce a query to the whole system in order to discover efficiently another sensor with the desired energy level, in which the task overhead must be eventually forwarded. In this way, the "Life-Expectancy" of the whole network could be increased. Never before, this context has been examined. According to [1], the benefits of the P2P overlays in sensor-

P2P Architectures	Lookup/update key	Data Overhead-Routing information	Join/Depart Node
Chord	$O(logN)$	$O(logN)$ nodes	$O(logN)$ w.h.p.
H-F-Chord(a)	$O(logN/loglogN)$	$O(logN)$ nodes	$O(logN)$
LPRS-Chord	$O(logN)$	$O(logN)$ nodes	$O(logN)$
Skip Graphs	$O(logN)$	$O(1)$	$O(logN)$ amortized
BATON	$O(logN)$	Two (2) nodes	$O(logN)$ w.h.p.
BATON*	$O(log_m N)$	m nodes	$O(mlog_m N)$
NBDT	$O(loglogN)$	$O(loglogN)$ or $2^{2^{i-1}}$ for nodes at level i of left spine	$O(1)$/periodical restructuring

Table 1: Performance Comparison between NBDT, Chord, BATON and Skip Graphs

nets are the following: Efficient Data Lookup, Guaranties on Lookup Times, Location Independence, Overlay Applications and Services, Elimination of proxies/sinks with undesirable central authority, Limited Broadcast. P2P design, for Internet-like environments, has been a very active research area and there are many P2P Internet protocols and systems available like CAN [14], Pastry [15], and Chord [17]. The main arguments against P2P designs in sensornets were the following: Logical Topology=Physical Topology, Route Maintenance Overhead, Sensor Nodes are Not Named, DHTs are Computationally Intensive.

By overcoming the arguments above (for details see [1], [2] and [3]), in [2] and [3] the first DHT (Distributed Hash Table) based protocols for sensornets were presented, the CSN and VRR respectively. In [1] the Tiered Chord (TChord) protocol was proposed, which is similar to, and inspired by, CSN. TChord is a simplified mapping of Chord [17] onto sensornets. Unlike CSN the design of TChord is more generic (to support a variety of applications and services on top instead of just serving incoming data queries). Gerla et al. argue for the applicability and transfer of wired P2P models and techniques to MANETs [8].

Most existing decentralized discovery solutions in practice are either DHT based, like Chord [17] or hierarchical clustering based, like BATON [10],[11] or Skip-Graphs [9]. Up to now, none of the existing P2P protocols for sensornets were designed in hierarchical clustering fashion, because the latter adds needless complexity to the design. On the contrary, all the existing P2P overlays for sensornets were designed in a DHT fashion and the best current solution is the TChord. Furthermore, none of the existing P2P overlays provide Energy-Level applications and services, close related to the so-called "life-expectancy" of a sensornet. This paper presents a novel variation of the existing optimal P2P method presented in [16], for desired Energy Level discovery, which combines the benefits of both DHT and hierarchical clustering methods. The variation above is called Energy Level Distributed Tree (ELDT) and its main functionalities attempt to increase the "Life-Expectancy" of the whole sensor network, providing support for processing: (a) exact match queries of the form "given a sensor node with low energy-level k', locate a sensor node with high energy-level k, where $k >> k'$" (the task will be forwarded to the detected sensor node) (b) range queries of the form "given an energy-level range $[k, k']$, locate the sensor node/nodes the energy-levels of which belong to this range" (the task will be forwarded to one of the detected sensor nodes) (c) update queries of the form "find the new overlay-peer to which the sensor node must be moved (or associated) according to it's current energy level" (the energy level of each sensor node is a decreasing function of time and utilization). ELDT overlay adapts the novel idea of NBDT P2P infrastructure presented in [16] providing functionalities in optimal time. For comparison purposes, an elementary operation's evaluation is presented in table 1 between NBDT, Skip-Graphs, Chord and its newest variation (F-Chord(á) [17], BATON [11] and its newest variation (BATON* [10]). The rest of this paper is structured as follows. Section 2 and 3 describe the ELDT system while section 4 analyses its basic functionalities. A new simulator and experimental evaluations via this simulator are presented in section 5 and 6 respectively. Section 7 concludes.

2. THE SNBDT PROTOCOL

SNBDT, is a simplified mapping of NBDT [16] onto sensornets. Like NBDT, at the heart of SNBDT is one main operation; the lookup operation. Given a set of sensor nodes, we hash the unique address of each sensor node to obtain node identifiers. Meta-data keys, generated from the data stored on the nodes, are hashed to obtain key identifiers.

Figure1 shows a SNBDT hierarchical arrangement of some *master nodes* (the big devices). As meta-data keys are basically information about data, they are much smaller than the actual data itself and replicating meta-data keys amongst neighbors of a sensor node will not require a lot of storage. The master node of level i maintains information (in its local finger table) about all its *slave nodes* (small devices) and $2^{2^{i-1}}$ other master nodes. All queries are resolved in a distributed manner with a bound of $O(\log \log N)$ messages. When a master node receives a query it first checks its own keys to resolve the query, if the lookup is not successful (note this means that the data element is not at the master node or any of its slaves) the master node then checks its local finger table. The finger table contains information about $2^{2^{i-1}}$ other master nodes and if the key can be located according to the information stored in the finger table, the query is directly forwarded to the master node storing the data. If the lookup on the local finger table also fails then the master node routes the query to the master node closest to the target according to the finger table. We handle the master_node joins/leaves and fails according to join/leave and fail operations respectively presented in [16]. In particular and concerning the fault tolerance issues, for each master_node, we maintain a cache of k redundant nodes (see figure 3) with each of them storing a replicated copy of a data item, where k>1 is a small positive constant, and make the assumption that the P2P overlay is "k-robust", meaning that the simultaneous failure of all these nodes is impossible, thus, at least one peer is alive in the overlay.

Slave nodes do not store information about their neighbors. If a slave node directly receives a query, it checks its own data and if the lookup fails it simply forwards the query to its master node. For simplicity, in the SNBDT proposal we opt for not connecting the slave nodes in a NBDT arrangement and lookups are not implemented in slave nodes (unless future experiment results prove otherwise). The master nodes of our proposal could be thought as "virtual sinks" with a NBDT overlay between these virtual sinks. The sensornet protocol (SP) by Polastre et al. [13] allows different MAC and link-layer technologies to co-exist by providing a standardized "narrow waist" interface to MAC, and provides an important step towards building a larger sensornet architecture. Unlike IP in the Internet, SP is not at the network layer but instead sits between the network and data-link layer (because data-processing potentially occurs at each hop, not just at end points). Figure 2 shows how P2P overlays can be implemented on top of SP. The P2P overlay (shown as P2P Overlay Management in Figure 2) could be built on top of any generic network protocol. An underlying DHT or Hierarchical Clustering routing protocol (e.g., VRR, CSN, TChord or SNBDT) is not necessary but recommended as it simplifies the job of overlay management and Caeser et al. show that it might be more efficient to build DHT-based routing directly on top of the link layer instead of implementing it as an overlay on top of a routing protocol [3]. P2P Services and Applications (e.g. event notification, resource allocation, and file systems) can then be built on top of the P2P overlay and sensornet applications could either use these services or communicate with the P2P overlay themselves.

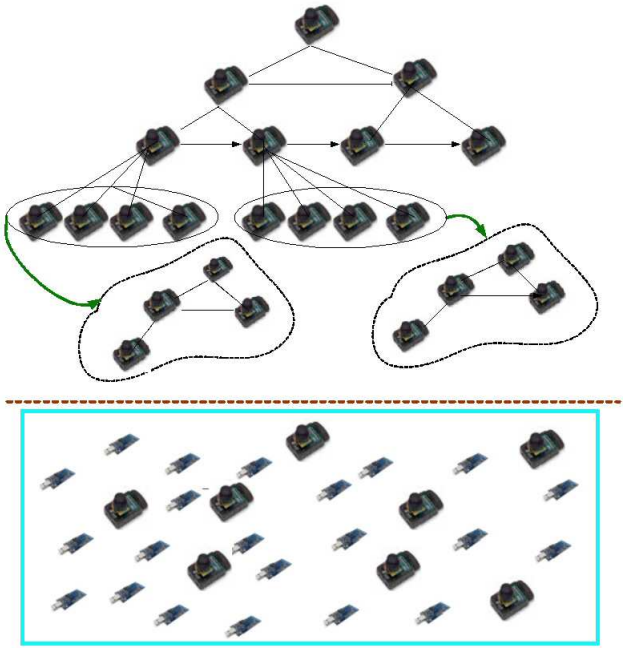

Figure 1: The SNBDT protocol

3. THE ELDT P2P OVERLAY

Let G a network graph of n sensor nodes and ELDT the respective overlay of N peers. With each overlay peer p

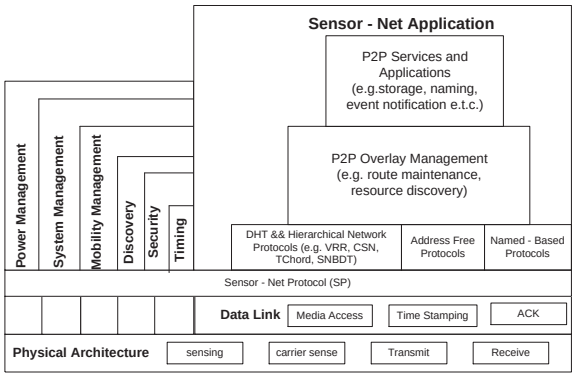

Figure 2: P2P Overlay in SP Architecture

$(1 \leq p \leq N)$ we associate a set of pairs $S_p = (g, L_g)$, where g is a sensor node $(1 \leq g \leq n)$ and L_g its current energy level. The criterion of associating the sensor node g to peer p depends on it's current energy level. Obviously, it holds that $N << n$. Let's explain more the way we structure our whole system.

The degree of the overlay peers at level i is defined to be $d(i) = t(i)$, where $t(i)$ indicates the number of peers present at level i. This is required to hold for $i \geq 1$, while $d(0) = 2$ and $t(0) = 1$ (see figure 3). It is easy to see that we also have $t(i) = t(i-1)d(i-1)$, so putting together the various components, we can solve the recurrence and obtain for $i \geq 1$: $d(i) = 2^{2^{i-1}}$, $t(i) = 2^{2^{i-1}}$. One of the merits of this tree is that its height is $O(loglogN)$, where N is the number of elements stored in it.

We build our overlay tree by repeating the same kind of tree-structure in each group of nodes having the same ancestor, and doing this recursively. This structure may be imposed through another set of pointers (it helps to think of these as different color pointers). The innermost level of nesting will be characterized by having a tree-structure, in which no more than two nodes share the same direct ancestor. Figure 3 illustrates a simple example (for the sake of clarity we have omitted from the picture the links between nodes with the same ancestor). Thus, multiple independent tree structures are imposed on the collection of peers inserted. Each element inserted contains pointers to its representatives in each of the trees it belongs.

Lemma 1: The maximum number of nesting of trees that we can have is itself $O(loglogN)$.

Proof: See [1].

Each overlay peer stores tuples $(g, L[g])$, where $L[g]$ is a $k - bit$ key belonging in universe $K = [0, 2^k - 1]$, which represents the current energy-level of the sensor node g. We associate to i^{th} peer the set $S_i = \{(g, L[g])\}$, where $L[g] \in [(i-1)lnK, ilnK - 1]$. Obviously, the number of peers is $N = K/lnK$ and the load of each peer becomes $\Theta(polylogN)$ in expected case with high probability (for more details see[1]). Each energy-level key is stored at most in $O(loglogN)$ levels. We also equip each peer with the table LSI (Left Spine Index). This table stores pointers to the peers of the left-most spine (for example in figure 3 the peers 1, 2, 4 and 8 are pointed by the LSI table of peer 5) and as a consequence its maximum length is $O(loglogN)$. Furthermore, each peer of the left-most spine is equipped with

the table CI (Collection Index). CI stores pointers to the collections of peers presented at the same level (see in figure 3 the CI table of peer 8). Peers having same father belong to the same collection. For example in the figure 2, peers 8,9,10 and 11 constitute a collection of peers. It's obvious that the maximum length of CI table is $O(\sqrt{N})$.

4. ANALYSIS OF BASIC FUNCTIONALITIES

We assume that we are located at sink sensor node $S \in G$, which has a very low energy level k' and as a consequence we want to search for another sink node $T \in G$ with energy level at least k. First, we locate the associating peer at the p2p overlay, let $s \in ELDT$, and then we find the range where the key k belongs. Generally assuming that $k \in [(j-1)lnK, jlnK-1]$, where N the number of current overlay peers, we have to search for the peer j. For example in figure 3 we are located at (green) peer 5 and we are looking for a key $k \in [13lnn, 14lnn-1]$. In other words we are looking for (green) peer 14. The first step of our algorithm is to find out the level of ELDT structure where the desired peer j (14 in our example) is located. For this purpose we have to exploit a nice arithmetic property of ELDT architecture. This property says that for every node x located on the left-most spine at level i, the following formula holds: $value(x) = father(x) + 2^{2^{i-2}}$ (1)

For each level i, $0 \leq i \leq loglogN$, we compute the left most peer, let x it's value, by applying the equation (1). Then we compare the value j with the computed value x. If $j \geq x$ we repeat the same process, otherwise we stop the loop process with current value i. The latter means that node j is located at i^{th} level as well as the maximum number of repeated loops is $O(loglogN)$. Then, we follow the i^{th} pointer of table LSI located at peer s. Let x the destination peer, meaning the left most peer of level i. Since the processing overhead compared to communication overhead is negligible, we ignore the $O(loglogN)$ processing factor at each peer, thus we need $O(1)$ hops only for locating the desirable peer x. Now, we must compute the collection in which the peer j belongs to. Since the number of collections at level i equals to the number of nodes located at level $(i-1)$ or $t(i-1) = 2^{2^{i-2}}$, we must divide the distance between j and x by the factor $t(i-1)$. Let $\left\lceil \frac{j-x+1}{2^{2^{i-2}}} \right\rceil = m$, that means we have to follow the $(m+1)^{th}$ pointer of table CI. Since, the collection indicated by $CI[m+1]$ pointer is organized in the same way at a next level of nesting, we continue the above process recursively. Due to the fact that the maximum number of nesting levels is $O(loglogN)$, the whole searching process requires $O(loglogN)$ hops or lookup messages. Let $r \in ELDT$ the target node at the $p2p$ overlay, which stores sensor nodes of G with the appropriate energy level k. Let $R \in G$ the one randomly chosen. Node R becomes the new sensor in which the task overhead will be eventually forwarded. The theorem 1 follows:

Theorem 1: Assume an embedding $ELDT$ lookup $P2P$ system into a sensor network G. Then, all the queries of the form (a), (b) and (c) require $O(loglogN)$ number of hops in worst-case.

Let G the sensor network and T the basic $ELDT$ overlay, without nesting levels. We are located at sensor node $S \in G$ with low energy level k' and we are looking for a sensor node $R \in G$ with the desired energy level k. Algorithm 1 depicts the pseudocode for the Sensor_Net_Search routine.

Let T the basic $ELDT$ overlay, without nesting levels. we are located at peer $p \in T$ and we are looking for the peer r, which stores sensor nodes with energy-level key k, let say $k \in [(j-1)lnK, jlnK-1]$. Algorithm 2 depicts the pseudocode for the Overlay_Search routine.

Let G the sensor network and T the overlay structure. We are located at sensor node $S \in G$, the energy level of which has been decreased from $k1$ to $k2$. We have to find the new overlay peer to which the update node S is going to be associated. Algorithm 3 depicts the pseudocode for the update_overlay_node routine.

Algorithm 1 Sensor_Net_Search(G,S,T,k',k,R)

1: Find the peer node to which S is associated {Let $p \in T$ the respective overlay peer}
2: $r = send\_overlay\_search(T, p, k)$ {Let $r \in T$ the peer node which stores sensor nodes with the desired energy-level k, let say R a randomly chossen one}
3: Return R

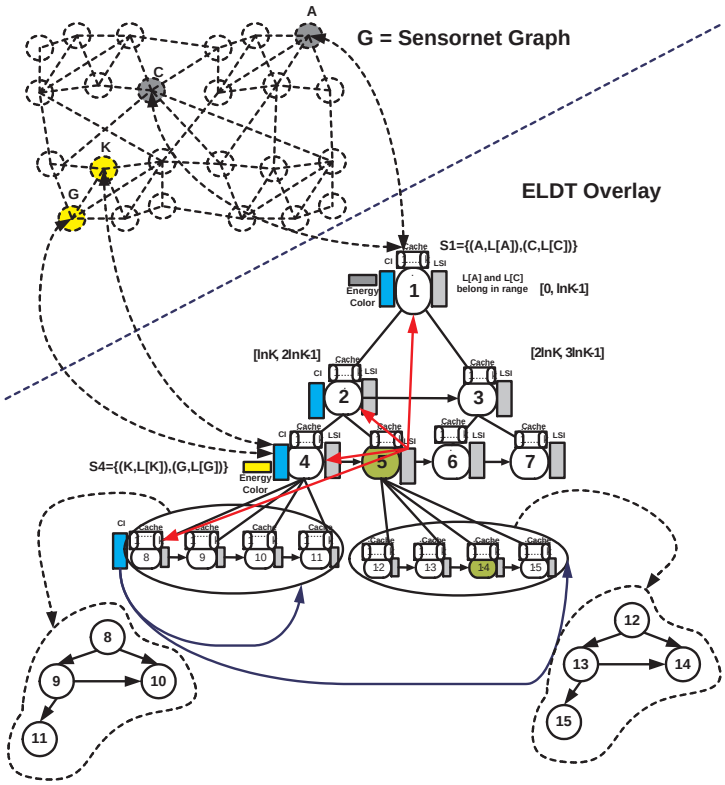

Figure 3: Building the ELDT Bipartite P2P Overlay

5. A P2P OVERLAY SIMULATOR FOR SENSORNETS

The basic architecture of the Java P2P simulator (see Figure 4) is based on the message passing environment. The peers exchange messages in order to build the overlay network and to carry out the search, insert and delete operations of a sensor with the specific energy level. We have

Algorithm 2 Overlay_Search(T,p,k)

1: Compute j such as $k \in [(j-1)lnK, jlnK-1]$;
2: if j=1 then i=0;
3: else if (j=2 or j=3) then i=1;
4: else
5: begin
6: x=4;
7: For (i=0; i<loglogN; ++i)
8: begin
9: $x = father(x) + 2^{2^{i-2}}$;
10: if j<x then break();
11: end
12: end
13: follow the LSI [i] pointer of peer p;
14: Let x the correspondent node;
15: $m = \left\lceil \frac{j-x+1}{2^{2^i-2}} \right\rceil$;
16: Follow the $CI[m+1]$ pointer of peer x;
17: Let y the first peer-node of the correspondent collection;
18: Let T' the $ELDT$ structure of the collection above at next level of nesting with root the node y;
19: $p = y$;
20: $r = send\_Overlay\_Search(T', p, k)$;{recursive call of the basic routine}
21: Return r;

Algorithm 3 Update_Overlay_Node(G,T,S,k_1,k_2)

1: Find the peer node to which S is associated according to old energy level k_1;
2: Let $p \in T$ the respective overlay peer;
3: Delete (S, k_1) from p;
4: $r = send\_overlay\_search(T, p, k_2)$;
5: Insert the tuple (S, k_2) into r;

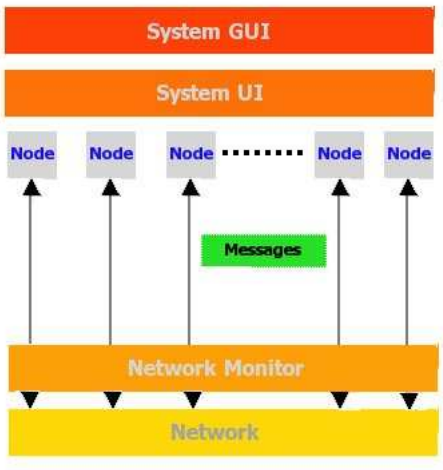

Figure 4: The Simulator's Model

implemented a class Message and a class Data. The class Message has four private fields which hold the id of the transmitter and the receiver, the type of the message and a pointer to the Data object. Also, this class has static final variables, which declare the type of the message (join, insert, delete, etc.). To simulate the network's behavior we have implemented the class Network, which consists of a buffer. The buffer is implemented with the Vector type of Java and stores objects of type Message. This class also implements several methods such as: sendMessage(), recvMessage(), broadcast() and msgForNodeId(). When a node wants to send a message it calls the method sendMessage which takes one argument, the message. This method which is synchronized attends to store the message into the buffer. If the type of the message is "search", "insert" or "delete", then the algorithm writes to a logger a record with the following format: i.e. "Search message for node 3 to node 45." The produced log messages are used by the graphical user interface to show how an operation is incrementally completed. On the other hand, each peer observes consecutively the network in order to verify if any message has been arrived. This operation can be done with Network's method msgForNodeId. If there is a message for this node the method returns the index where the message will be stored into the buffer, otherwise it returns -1. In the first case, the node can receive the message by calling the method recvMessage. The broadcast() method is used during the construction of the overlay for sending the same message over a range of peers. This method which is also synchronized is a wrapper function of sendMessage and is used only at the time of the network construction, since no broadcast is needed during a search operation.

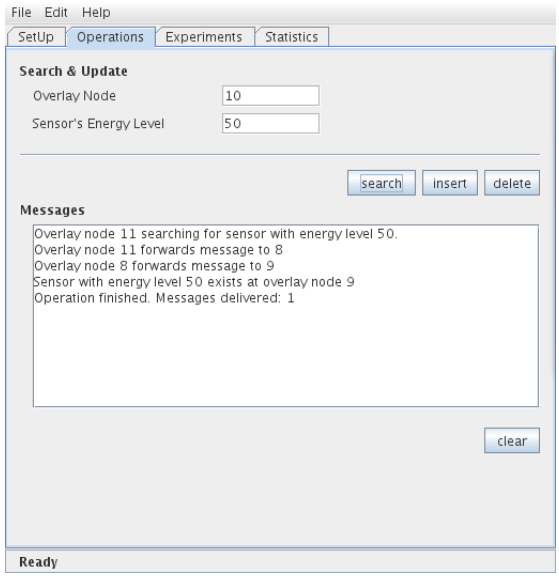

Figure 5: The tab "Operations"

5.1 The Peer Nodes

To simulate the overlay peers we have constructed the class Node which extends the Java's Thread class. This class holds the tables which are necessary for constructing the overlay and provides the methods for joining an incom-

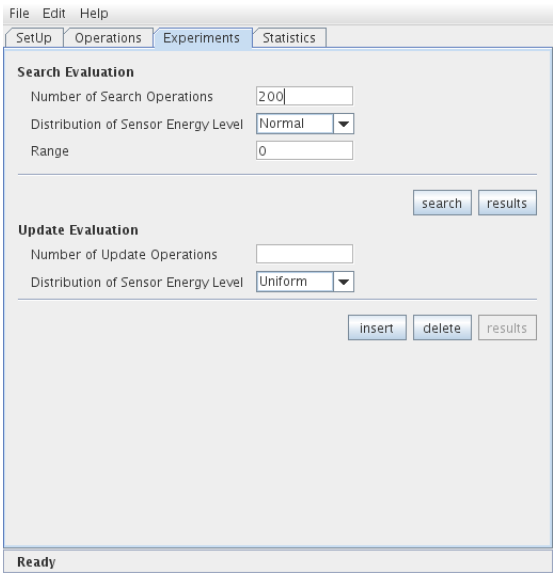

Figure 6: The tab "Experiments"

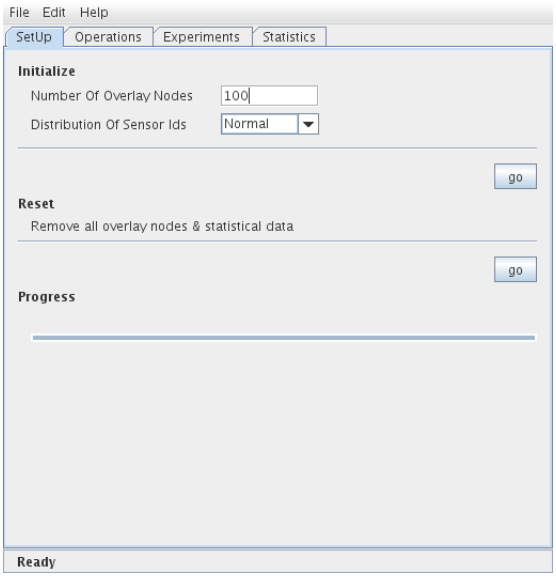

Figure 7: The tab "SetUp"

ing node, deleting an existed peer, searching and updating a sensor's energy level. The threads communicate via the network exchanging messages as described in the previous section. If a new peer wants to join, the network has to send a join message to one of its introducer nodes. The introducer node listens to the request and forwards the message to the appropriate node.

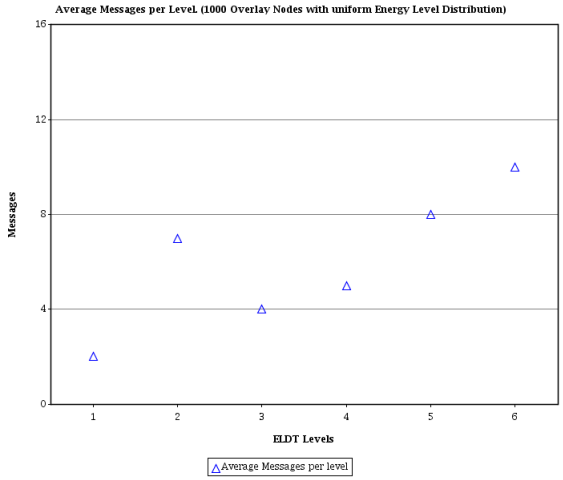

Figure 8: Average Messages per Level

5.2 System User Interface

We have implemented a class SimUI which is the interface of our system. This class provides the methods for the initialization of our system, the functions for the search, insert and delete operations and methods which provide general information about the status of our structure as the total number of sensors and peers, the load balance and the range of the sensors' energy level. This is the main class which also starts up the graphical user interface which we describe in the next.

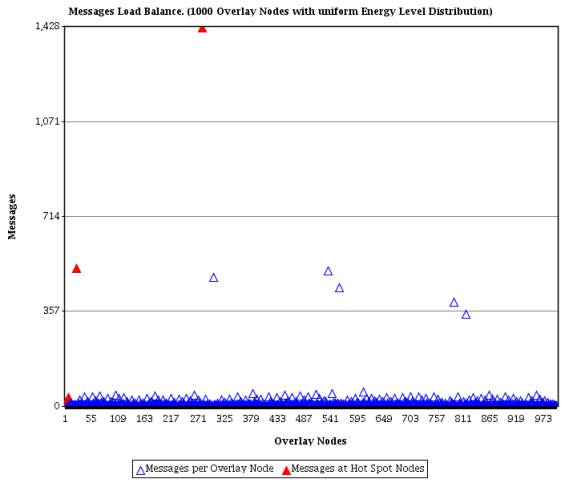

Figure 9: Load balance after 200 updates with uniform distribution.

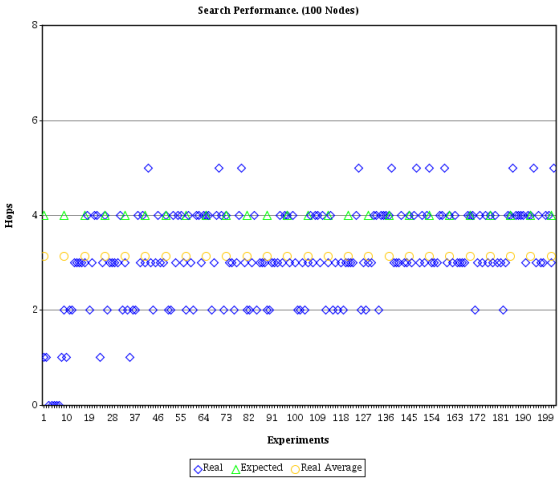

Figure 10: Lookup Performance Graph

5.3 Graphical User Interface

GUI has been implemented with NetBeans 6.0 software and consists of a window with four tabs. In the first tab (see Figure 7) the user can set the number of peers which will constitute the overlay and select the energy level distribution over these nodes. The available distributions are: uniform, normal, beta, and pow-law. After the user has set these two fields then the system's initialization can begin. In the same tab there is a progress bar so the user can obtain the overall process due to the fact that this process may take several minutes. Also there is a button, which resets the system without the need of closing and reopening the simulator if we want to carry out several experiments with different number of peers and energy level distribution. The other three panels (see Figure 5) provide the ability to search, insert and update the energy level of a sensor starting the procedure from any peer in the overlay. While one of these operations is being executed, appropriate messages are appearing at the bottom of this tab. In the third tab (see Figure 6) the user can prosecute experiments to evaluate the efficiency of the lookup/update operations. There are two panels one for each operation where the user sets the number of the experiments and selects the distribution according to the energy-level keys of the sensors picked up for the experiments. After the termination of the experiments the user can see and save the chart that has been generated. In the forth tab - statistics - the user can see the current number of peers into the system, the number of sensors that have been stored over the peers and the range of sensors' energy level that we can store in the overlay. This tab represents also performance statistics such as the minimum, the maximum and the average path of the total operations that have been performed. Furthermore, this tab generates a chart with the sensor's energy level distribution over the peers (see Figure 9), the number of messages that have been forwarded by each peer (see Figure 10)and the number of messages per tree level (see Figures 8).

6. EXPERIMENTAL RESULTS

By using the p2p java simulator described in previous section, we evaluate the search path length and the load balancing performance of the following two protocols: SNBDT and TChord. In order to understand in practice the routing performance of these two protocols, we simulated a random network graph G of n sensor and N overlay-peers, where $N = 2^k$ and $n = 2^{2^k}$, storing $K = 100 \times 2^k$ energy-level keys in all. We varied parameter k from 3 to 14 and conducted a separate experiment of each value.

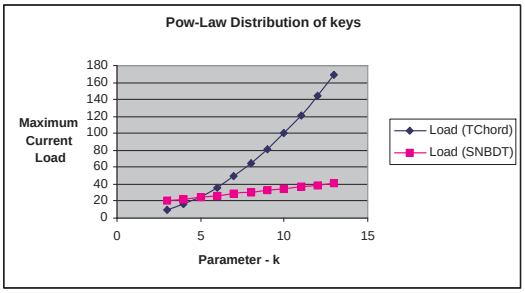

Figure 11: Load Balancing Performance Comparison with TChord when the inserted/deleted energy-level keys draw pow-law distribution

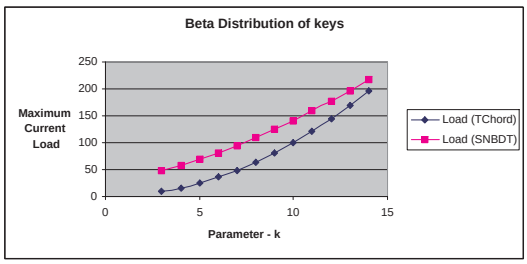

Figure 12: Load Balancing Performance Comparison with TChord when the inserted/deleted energy-level keys draw beta distribution

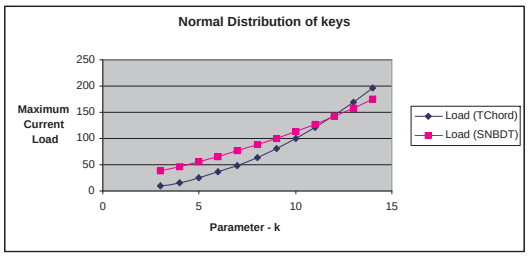

Figure 13: Load Balancing Performance Comparison with TChord when the inserted/deleted energy-level keys draw normal distribution

Each node in an experiment picked a random set of keys to query from the system, and we measured the path length required to resolve each query. For the experiments we considered synthetic data sets. Their generation was based on three bad distributions the behaviour of which even belongs to smooth family it is also far away from the known good

distributions like uniform or regular. We talk about pow-law, Beta and Normal distribution respectively. Figure 11 depicts that for skew distributions (like pow-law), the load balancing of SNBDT outperforms TChord by a wide margin. Figures 12 and 13 depict that for Gaussian like (Beta) or normal distributions the curves of SNBDT and TChord converge, while the parameter k increases. The lookup and update efficiency of our system is obvious (see figures 14 and 15) and in accordance with the aforementioned theoretical complexities. In all experiments we've done SNBDT outperforms TChord.

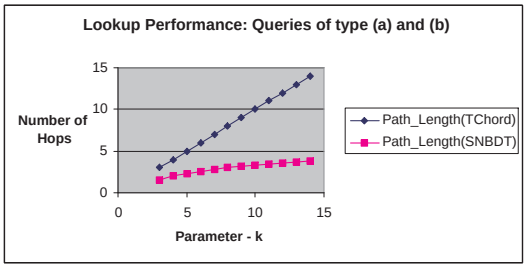

Figure 14: Lookup Performance Comparison with TChord

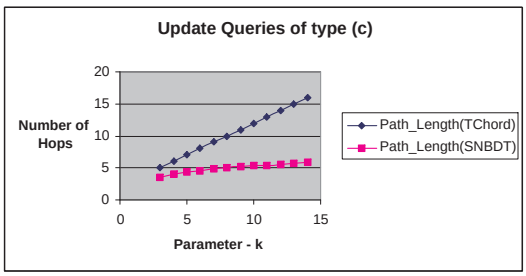

Figure 15: Update Performance Comparison with TChord

7. CONCLUSIONS

We designed a novel P2P overlay for Energy Level discovery in a sensornet, the so-called ELDT. It is the first time where a P2P overlay network was built providing "Energy-Level" functionalities. We experimentally verify these functionalities via an appropriate simulator we have designed for this purpose. An extended experimental performance comparison with all the well-known P2P mapping schemes will be included in the forthcoming journal version.

8. REFERENCES

[1] Muneeb Ali and Koen Langendoen, A Case for Peer-to-Peer Network Overlays in Sensor Networks, International Workshop on Wireless Sensor Network Architecture(WWSNA'07), pages 56-61, Cambridge, Massachusetts, USA, 2007.

[2] M. Ali and Z. A. Uzmi., CSN: A network protocol for serving dynamic queries in large-scale wireless sensor networks. In 2nd CNSR'04, pages 165-174, Fredericton, N.B, Canada, 2004.

[3] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron., Virtual Ring Routing: Network routing inspired by DHTs. In ACM SIGCOMM'06, pages 351-362, Pisa, Italy, 2006.

[4] Crainiceanu, A., Linga, P., Gehrke, J. and Shanmugasundaram, J., P-Tree: A P2P Index for Resource Discovery Applications, WWW'04, pages 390-391, New York, NY, USA, 2004.

[5] D. Clark, C. Partridge, R. T. Braden, B. Davie, S. Floyd, V. Jacobson, D. Katabi, G. Minshall, K. K. Ramakrishnan, T. Roscoe, I. Stoica, J. Wroclawski, and L. Zhang., Making the world (of communications) a different place. ACM SIGCOMM'05 CCR, 35(3):91-96, Philadelphia, PA, 2005.

[6] M.Demirbas, A.Arora, and M.Gouda., A pursuer-evader game for sensor networks. Sixth Symposium on Self- Stabilizing Systems(SSS'03), pages 1-16, San Francisco, CA, USA, 2003.

[7] Murat Demirbas, Hakan Ferhatosmanoglu, Peer-to-Peer Spatial Queries in Sensor Networks, IEEE Proceedings of the 3rd International Conference on Peer-to-Peer Computing, pp. 32-40, Linkoping, Sweden, 2003.

[8] M. Gerla, C. Lindemann, and A. Rowstron., P2P MANET's - new research issues. In Dagstuhl Seminar Proceedings, number 05152, Germany, 2005.

[9] Goodrich, M.T., Nelson, M.J. and Sun, J.Z., The Rainbow Skip Graph: A Fault-Tolerant Constant-Degree Distributed Data Structure, ACM SODA'06, Pages 384-393, Miami, FL., 2006.

[10] H. V. Jagadish, B. C. Ooi, K. L. Tan, Q. H. Vu and R. Zhang., Speeding up Search in P2P Networks with a Multi-way Tree Structure, ACM SIGMOD'06, pages 1-12, Chicago, Illinois, 2006.

[11] H. V. Jagadish, B. C. Ooi, and Q. H. Vu., Baton: A balanced tree structure for peer-to-peer networks. In Proceedings of the 31st VLDB'05 Conference, pages 661-672, Trondheim, Norway, 2005.

[12] A.Mainwaring, J.Polastre, R.Szewczyk, D.Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. ACM Int. Workshop on Wireless Sensor Networks and Applications, Pages 88-97, Atlanta, Georgia, USA, 2002.

[13] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica., A unifying link abstraction for wireless sensor networks. In 3rd ACM SenSys'05, pages 76-89, San Diego, 2005.

[14] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Schenker, S., A scalable content-addressable network, ACM SIGCOMM Computer Communication Review 31(4),161-172.

[15] Rowstron, A. and Druschel, P., Pastry: A Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, Springer Book Chapter, LNCS 2218, 329-350.

[16] S.Sioutas, NBDT: An efficient p2p indexing scheme for web service discovery, Journal of Web Engineering and Technologies, Vol. 4 (1), pp 95-113, 2008.

[17] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F. and Balakrishnan, H., Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transactions on Networking (TON) 11(1), pages 17-32, 2003.