

Managing Enterprise Resource and Environment through Real-Time Tracking, Monitoring and Actuation of Enterprise Objects

Anurag D¹, Siuli Roy¹ and Somprakash Bandyopadhyay¹

¹Management Information Systems

Indian Institute of Management Calcutta

anurag@email.iimcal.ac.in, {siuli,somprakash}@iimcal.ac.in

Abstract

In this paper, we present a unified framework for both asset and environment visibility of an enterprise through real-time tracking and monitoring of enterprise objects using the Internet. The objects that are monitored can be personnel, equipment or an environmental condition. The architecture also enables the control of asset utilization by triggering actuators. The wireless sensor network is 2-tiered where the sensors form the leaves of the network and do not participate in routing. In this paper, we develop the motivation to revisit the hierarchical routing in mesh networks and highlight the control packet overhead in AODV and the lack of fault tolerance in Cskip of ZigBee. We develop a new routing scheme that merges the flexibility of AODV and the structure of hierarchical addressing to provide fault tolerance at a lower overhead. We provide the algorithm and experimental simulation of the routing scheme.

INTRODUCTION

The availability of consistent, accurate and timely information greatly improves the quality and speed of planning and decision making in any organization. Total Asset Visibility (TAV) [10] is a term, used in US Department of Defense's logistic practices, that implies knowing where assets are at all times. It also implies having a unique identity for each item and knowing what is happening to it, as it happens. TAV can usefully be applied to any organization for improving enterprise visibility. Timely and accurate information on the location, movement, status, and identity of units, personnel, equipment, material and supplies can improve the resource utilization by a great extent. It also improves the capability to act upon that information for better performance of the organization. The core of this kind of Asset Visibility System is the Automatic Object Identification technology, like RFID, integrated with wireless mesh networks to communicate this identification data to a remote station.

An Enterprise visibility system is not confined to asset visibility only. The same framework can be used to monitor

the enterprise environment, to protect it against possible damage by terrorist attacks, fire or emission of toxic gases (in case of chemical industries). Wide variety of sensors connected together using wireless sensor networks forms the core of this **Environment Visibility** System.

Moving a step further towards a "better" visibility, all this information should be available in real-time from anywhere. This "anywhere" accessibility of information through the Internet, which in turn is linked with the core visibility system in an organization, enables a corporate manager to access relevant enterprise information in real time, as and when needed.

This paper presents a unified framework for both asset visibility and environment visibility of an enterprise through real-time tracking and monitoring of enterprise objects using the Internet. It also allows controlling the asset utilization and the environment by triggering actuators, which are also part of the enterprise objects. The core objective is to develop a system that helps not only to view but also to control enterprise objects through the Internet.

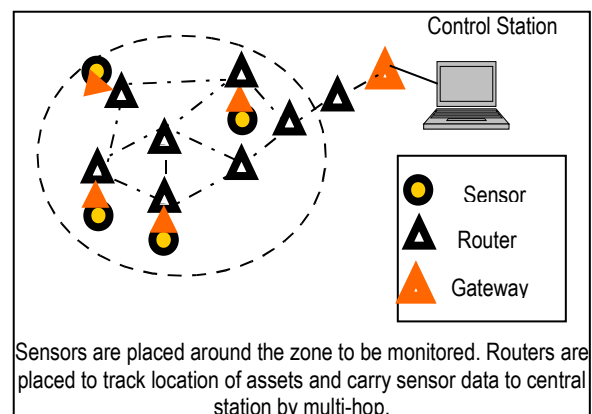


Figure 1. A conceptual Framework for Tracking, Monitoring and Actuation of Enterprise "objects" through the Internet

The paper is structured as follows. We provide the motivation for a new routing protocol in section 2 and the related work in section 3. The new protocol is presented in section 4. Section

5 provides the evaluation and we conclude with scope for further work in section 6.

MOTIVATION

We study the deployment of a wireless sensor network with sensors being placed at particular points in the region to be monitored. The sensed parameters include temperature, vibration and (harmful) gases. The sensors are placed at pre-determined points and it is well understood by the sensor network user that the sensed parameters correspond to the point of the sensor location rather than the region (for e.g. a vibration sensor is placed on a specific asset and is immobile). The sensed information is to be transported to the centralized data repository located at the sink. The sensors do not participate in the packet forwarding and relays are placed at appropriate locations to transmit the data in a multi-hop fashion to the sink. Such networks are known as 2-tiered and are the most common form of sensor network deployment.

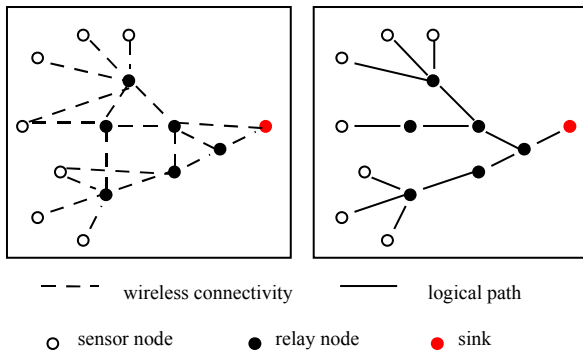


Figure 2. A mesh network reduces to a hierarchical minimum spanning tree when data communication is required only between the nodes (sensors and relays) and the sink.

The data propagates from the sensors to the sink and vice-versa. There is no requirement for one sensor to communicate to another sensor, or one relay to communicate to another relay. If we plot the topology of the best path (the metric can be hop counts, signal strength, etc) from the sensors to the sink, it would result in a hierarchical structure as shown in figure 1. This is analogous to the minimum spanning tree.

The AODV routing algorithm builds the route tables by a flooding the network with RREQ control packets. These control packets are costly since all devices are battery powered and packet transmissions are the biggest source of energy dissipation. In case of node failures, RRER control packets are generated and subsequent RREQ packets to establish the new route.

The C_{skip} algorithm used by ZigBee allocates addresses to each node such that routing can be made without having to generate any routing table. It earmarks each node to have a fixed number of children. This fixed allocation restricts the total depth (number of hops) that the network can support. For example, if every relay is expected to support upto 9 relays as its children, the maximum depth is 4. As shown in

figure 2(a), a device at depth 5 cannot join the network. The ZigBee hierarchical routing algorithm also does not support fault tolerance. When a node in the network goes down, all its child nodes look for a new parent. If a new parent is found, the child node's address changes. This in turn results in the changing of the address of all the children of the child node and continues recursively till the leaves of the network. While obtaining a new parent, if the depth of the node increases, devices that were part of the network earlier may not be able to join. This is shown in figure 2(b). The link between devices numbered 2 and 1 goes down. Device 2 connects to device 93 and is reassigned an address of 104. Note that its depth has also increased. As a consequence the children of 2, are renumbered as 105 and 106 while the sensors, earlier numbered 4, 5, 6 and 14 are no longer able to be part of the network although they are in wireless range. This is due to the address wastage of C_{skip} where it has earmarked addresses for non-existent devices.

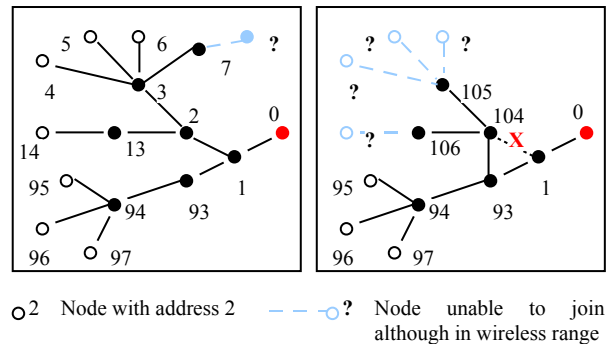


Figure 3(a). The nodes given addresses according to C_{skip} with each relay expected to support upto 9 children. In such a case, depth larger than 4 is not possible. **(b).** Break in link between 2 and 1 (shown as X) results in 3 devices being assigned a new address and 4 sensors no longer part of the network.

As a further example, if the number of child relays a relay is able to support is set to 5, the maximum depth is 6. Similarly, if the max child relay is set to 15, the maximum depth is 4.

In this paper, we develop a routing algorithm that includes both the properties of the hierarchical as well as AODV protocols. As devices join a network, they are given address to satisfy the hierarchical property. Here, the address generation is made based on the expected number of devices to join the network. As the network grows with more devices joining, the expected number of devices to join falls. Thus the address space (and wastage) is reduced as the network grows. When a condition occurs where the hierarchical property can no longer be supported, a route table entry, similar to AODV, is created only for this non-conforming node. This route table entry is also created when a node/link fails thus preventing other nodes from having to change their addresses or depth. Our routing algorithm can be tuned to the probability distribution of node deployments. For example, a dense network with low depth can be tuned to a uniform distribution. In case of a network with long chains (effectively

large depth) a geometric distribution can be used. We term our probability based routing algorithm as a hybrid of the hierarchical and AODV algorithms.

The aim is to achieve a routing algorithm suitable for 2-tiered wireless sensor network which has a low control packet overhead and at the same time supports large network depth and is fault tolerant.

RELATED WORK

Numerous studies have been made on achieving efficient routing in wireless mesh networks. Significant work includes flooding [11], the AODV protocol [2] and the Location Aided Routing (LAR) protocol [7]. In hierarchical routing, the addresses of the nodes play a critical role as they decide the routing decision. A hierarchical address allocation is made by the C_{skip} algorithm of ZigBee [14]. Here, a device joins the network by requesting an address from a particular device, now designated as the parent. The parent gives an address according to the equation:

$$A_{new} = A_{parent} + C_{skip}(d) \quad (1)$$

where d is the depth of the child and C_{skip} is the maximum number of devices that can join under all the other children of the parent. This scheme looks to reserve addresses in the event a node joins at a particular level. Since this cannot be predicted a priori, most reserved addresses are never utilized restricting devices to join in an already dense area. This restricts the depth of the network. The C_{skip} algorithm was mainly expected to be used in a star network where typically there is no multi hop data transmission. However, this address wastage problem has been thought about in the latest ZigBee specification made available to the general public on January 2008 [14]. The specification in addition to C_{skip} , has included support for a stochastic addressing scheme where addresses are assigned in a random fashion. Here, there is no guarantee for uniqueness of the address and address conflict negotiation mechanisms are defined. Stochastic addressing automatically implies AODV mechanism is used for routing.

Work on optimized deployment and fault tolerance has recently gained momentum. Sensor coverage problems [10, 12] have looked at placing enough sensors to cover the entire area of deployment. Once an initial topology is obtained, additional routers (or relays) can be placed to achieve fault tolerance [6, 13]. A $(k-1)$ fault tolerant network is one where every node is k connected and the network remains connected even after $k-1$ node failures. Existing sensors are modeled as a unit disk graph with the sensors designated as a vertex. The goal is have every vertex k -connected. Thus these optimization schemes augers well with the hierarchical topology, however no work has been made at developing an addressing and routing scheme for wireless hierarchical networks that support fault tolerance.

HYBRID ROUTING

The sensor network comprises of one sink, multiple sensors that are fixed (immobile) and form the leaves of the network. Routers are placed to relay the data from the sensors to the sink and vice-versa. Each of the sensors, routers and the sink are homogeneous and have a fixed radius of transmission. The network grows from the sink towards the leaves. The sink is the first device in the network to come up and takes the address of 0. Subsequent routers/sensors request an address from an already associated device in its range. Our algorithm provides this address in an intelligent way. There is no apriori information of which device would come up next and hence the algorithm is generic and supports any structure of the topology. A sample topology is shown in figure 3.

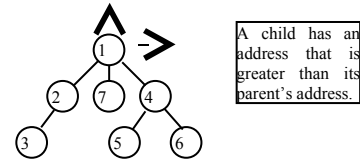


Figure 4. A sample hierarchical topology.

The address of a device uniquely identifies it in the network. Further, the address is used to make routing decisions. For example, in figure 3, if node numbered 1, receives a packet with destination address 3, it must be able to decide to which of its children (among 2, 7 and 4) the packet must be forwarded, to ensure the packet reaches the destination. In most routing protocols, typified by AODV, the node needs to maintain a list guiding it on the network topology. However, in the case of a hierarchical network, the node addresses are assigned in a way to ensure routing decisions can be made. We define this decision of a node as its routing strategy S . Note that the strategy, S , must be simple enough to run on resource (computational) constrained devices in acceptable time. Further, the strategy must be the same for all the nodes, i.e., referring to figure 3, node 1 cannot make a routing decision unless it knows the strategy followed by its children. We now define the strategy, for our proposed algorithm, S as in (2) and a new requesting device is given an address according to (3).

$$S : ChildrenAddresses > ParentAddress \quad (2)$$

$$A_{new} = A_{max} + E(n) \quad (3)$$

Where A_{max} is the maximum address reachable through the parent, $E()$ is the expected value of the probability distribution and n is the number of devices left to join the network. The algorithm thus calculates the expected number of devices for a particular distribution function, rather than earmark the worst case solution as in C_{skip} . Further, in cases where the number of devices joining exceeds the expected value, an exception list is maintained which is the route entry only for the non conforming device similar to the AODV entry. Thus, our addressing algorithm, maintains as much as possible, the hierarchical structure. However, in cases where

it is not possible, an exception is created and saved as a list. The algorithm essentially is a merge of the structured hierarchical network and the completely unstructured AODV mechanism.

The Hybrid Routing Algorithm

- 0: Node Start-up (switched on). Perform Initialisation.
- 1: Perform Channel Scan.
 - If no devices found in range, goto 1 else set my_parent = device with strongest signal strength
- 2: Send Associate_request to my_parent
- 3: If associate_response received from my_parent
 - set my_address = address from associate response
- 4: If Associate_request received, then
 - If I am not PAN coordinator then
 - send new_address_request to my_parent with P=my_address as parameter
 - Else
 - Calculate E = num of devices expected to join.
 - set N = last child address+ E
 - Send Associate_response with N as the new address
- 5: If new_address_request received from any node R
 - If I am not PAN coordinator then
 - forward new_address_request to my_parent
 - Else
 - calculate E = num of devices expected to join under P
 - set N = last child address of P + E
 - If N breaks hierarchical routing, create route table entry with N as destination and R as next_hop
 - reply with new_address_response; P, N as parameters
- 6: If new_address_response received from any node R
 - If N breaks hierarchical routing, create route table entry with destination as N and R as next_hop
 - If my_address = P then
 - send associate_response with N as address
- 7: If data packet for routing is received
 - If destination exists in route table forward to next_hop else forward according to hierarchy
- 8: If packet sending to my_parent fails then
 - perform channel scan
 - set my_parent=device with strongest signal
 - if device is not reachable through me
 - send route_update to my_parent with my_address as parameter
- 9: If route_update received from any node R
 - If address breaks hierarchical routing, create/update route table entry with address as destination and next hop as R
 - forward route_update to my_parent

Lemma 1: Defining the routing strategy as in (2) and the addresses as in (3), newly assigned addresses cannot invalidate the already setup routing scheme.

Proof: A sample topology is shown in figure 4. Assume an existing node with an address "A". We define all the addresses of the nodes (parents) connecting "A" from the sink

as " P_k ", where $k=1$ to D , D = depth of A. Let a new node joining the network get an address "N". Let " Q_k " be the list of addresses of the nodes connecting "N" to the sink with $k=1$ to D , D = depth of N. Select the node common to P_k and Q_k that has the highest address. (There will be atleast one node common to P_k and Q_k). Let this node be denoted C. Let the node under C with the largest address be denoted as C_n . The routing to A will fail if N joins the network at C and $N < A$. However, $N > C_n$ and $C_n \geq A$ by (3). Thus the routing to A is preserved.

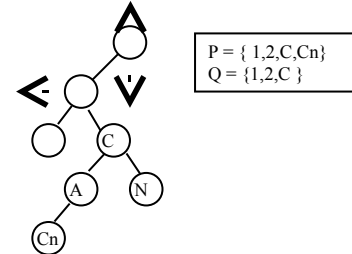


Figure 5. Sample topology for Lemma 1

Support for Large Network Depth:

In this section, we show pictorially how the algorithm handles networks of large depth. Considering the example shown in figure 2(a), the new node must be given an address to join the network. However, no address will satisfy the hierarchical property. We therefore give it an address based on the expected number of devices under node 7. The new address creates route table entry only at node 2. Thus a single route table entry is sufficient for routing. As shown in figure 6, assume the new node is given an address of 23.

A packet destined for node 23 is routed as follows. At node 1, the hierarchical strategy eqn. (2) implies the packet must be forwarded to node 2. Similarly, the hierarchical property at node 2 implies the packet must be forwarded to node 13. However, the route entry is present at node 2 directing the packet to node 3. The hierarchical property is satisfied at node 7 as well thus reaching the packet to node 23.

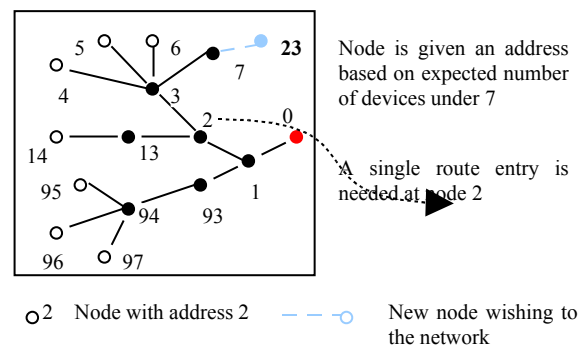


Figure 6. Support for large network depth

Support for Fault Tolerance:

The hybrid addressing topology has inherent capability to support fault tolerance. In the event of a node failure, a node

selects the new parent that would enable it to reach the sink. It presents its address to the new parent. The new parent in turn, checks for the hierarchical property and creates/updates route table entries if needed and forwards the packet to its parent. This can be considered similar to the AODV style RREQ (route request) and RREP (route reply) messages.

An example is shown in figure 7. The link between nodes 2 and 1 break and node 2 joins node 93. Since all of the addresses under node 2 are less than 93, a route table entry is required for each of the 7 devices at node 1. However, no route table entries are needed at any other node since the hierarchical property is satisfied elsewhere.

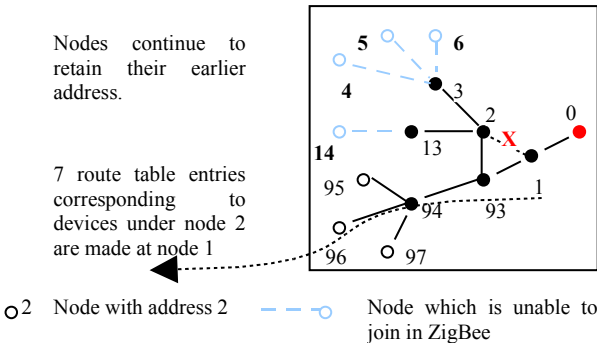


Figure 7. Support for fault tolerance

EXPERIMENTAL EVALUATION

We implemented the routing scheme assuming a uniform distribution for node generation. A node can join any of the previously joined nodes with equal probability. Referring to figure 11, “k” denotes the number of nodes in the leg where the new device is joining and “n” denotes the number of devices yet to join. The expected number of devices under a particular node is calculated in $O(n)$ time. The appendix provides the implementation details.

We compare the above address allocation algorithm with the memory requirement in AODV for storing the route tables. Intuitively, our scheme must show a significant improvement since the address allocation happens in an intelligent way where the hierarchical structure is maintained as much as possible. In cases where this is not possible, the AODV style route table for only the exception device is stored. An additional overhead in our scheme is a single unit to store the maximum address under the device as is needed by (3).

Memory Requirement

The simulation was carried out in C++ where the node placement is randomized. The placement of a new node was studied in two settings. In one, a uniform probability distribution of node placement was done. Here, a new node N would have equal probability to choose any of the previously deployed nodes as its parent. In the second setting, a geometric probability distribution was used. Here a new node

N has maximum probability of choosing the last deployed node as its parent and the probability of choosing an earlier deployed node falls proportionately. The geometric distribution is a more realistic representation of the practical scenario. Once a random network topology is generated, the memory needed for AODV and the hybrid scheme is calculated for the same topology. We calculate the network average (denoted as AVG) and the node average for the node with the biggest route table (MAX).

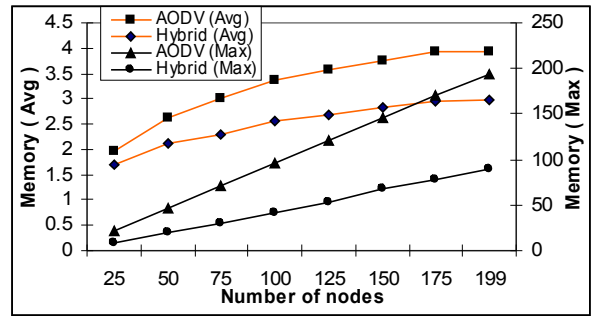


Figure 8. Memory Requirement for network topology generated through a uniform distribution.

The Hybrid routing, as expected, clearly outdoes the AODV scheme on both the network average and the node average counts. The hybrid scheme requires around 50 % less memory. Figure 9 gives the performance of the AODV protocol and the Hybrid protocol for the geometric distribution of topology generation. Interestingly, AODV suffers significantly compared to the Hybrid algorithm. This can be explained by noting that the topology would be of larger depth on most occasions. In a uniform distribution, the topology would be lower in depth and wider in width. Therefore in uniform distribution there would be more nodes that have no children. When a node has no children, it needs no route table. However, for a geometric distribution, nodes with no children are smaller in number thus leading to a larger average requirement of memory. The hybrid algorithm is more comfortable with the geometric distribution as increase in depth leads to a lower increase in the number of exceptions.

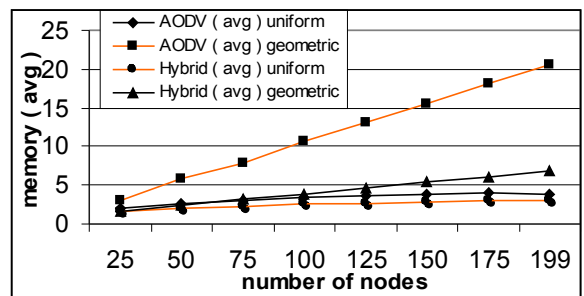


Figure 9. Memory requirement for AODV and Hybrid under uniform distribution and geometric distribution ($p=0.8$)

Table 1. Simulation Setup.

Simulation Environment	C++
Number of nodes in network	25 to 200, in steps of 25
Location of nodes (Sensitivity Runs)	pseudo random uniform distribution
	pseudo random geometric distribution with $p=0.8$
Number of runs	100 for each network size

CONCLUSION

Our proposed hybrid algorithm for addressing and routing in hierarchical networks borrows concepts from C_{skip} of ZigBee and AODV protocol. By merging the flexibility of AODV and the structure of C_{skip} , we achieve significant improvements in terms of memory needed for route tables over AODV and flexibility and support for fault tolerance over C_{skip} . Unlike AODV based addressing, where node addresses are given in a stochastic manner, we give addresses in a clever scheme where the hierarchical structure is maintained as much as possible. Where addresses can be maintained in a hierarchy, there is no need for routing tables. In cases, where the hierarchy breaks, we maintain a route entry only for the non-conforming node. This exception is the overhead in our scheme. Simulation results have shown significant reduction in the routing tables (around 50%) and much better scalability with network size than AODV. The simulation also shows the rather poor address uniqueness property of our scheme (around 33%). We have identified a mechanism to abate this concern through the use of small stochastic elements.

We look to explore the inclusion of the stochastic components in our algorithm and study the improvement in the uniqueness property in our future work. Work has also been made in a practical implementation. We seek to make a large scale implementation and study the ease of deployment. The impressive gains of our scheme inspire us to develop the algorithm into practical implementations.

REFERENCES

- [1] Akyildiz, I.F., Xudong Wang: A Survey on Wireless Mesh Networks, IEEE Communications Magazine, September 2005.
- [2] A. Wheeler, "Commercial Applications of Wireless Sensor Networks using ZigBee", IEEE Communications Magazine, April 2007.
- [3] C. Perkins, E. Royer, "Ad hoc On-Demand Distance Vector Routing", IEEE Workshop on Mobile Computing Systems and Applications, 1999
- [4] C.S. Raghavendra, A. Avizienis and M.D. Ercegovic, "Fault Tolerance in Binary Tree Architectures," IEEE Transactions on Computers, 1984.
- [5] G. Lewis et al, "A System for Simulation, Emulation, and Deployment of Heterogeneous Sensor Networks," SenSys '04.
- [6] ITU Report on Internet of Things – Executive Summary: www.itu.int/internetofthings/
- [7] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, D. Rus "Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance," MobiHoc 2005.

- [8] K. Young-bae and N. Vaidya, "Location-aided Routing (LAR) in Mobile Ad Hoc Networks", ACM MobiCom, 1998.
- [9] M. B. Lowrie and W. K. Fuchs, "Reconfigurable Tree Architectures Using Subtree Oriented Fault Tolerance," IEEE Transactions on Computers, October 1987
- [10] R. Peng, S. Mao-heng, Z. You-min, "ZigBee Routing Selection Strategy Based on Data Services and Energy-balanced ZigBee Routing", IEEE Asia-Pacific conference on Services Computing ,December 2006.
- [11] S. Toumpis, L. Tassiulas, "Optimal Deployment of Large Wireless Sensor Networks", IEEE Transactions on Information Theory, 2006.
- [12] W. R. Heinzelman, J. Kulik, H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", ACM MobiCom, 1999.
- [13] W. You-Chiun, H. Chun-Chi, T. Yu-Chee, "Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network", IEEE Transactions on Mobile Computing, 2008.
- [14] X. Han, X. Cao, E. L. Lloyd, S. Chien-Chung, "Fault Tolerant Relay Node Placement in Heterogenous Wireless Sensor Networks," InfoCom '07.
- [15] ZigBee Alliance: www.zigbee.org