This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2008 proceedings.

Location-Aided Topology Discovery for Wireless Sensor Networks

Mingze Zhang, Mun Choon Chan and A. L. Ananda
School of Computing, National University of Singapore
Email: {zhangmi3,chanmc,ananda}@comp.nus.edu.sg

***Abstract*— Topology discovery in sensor network is useful in any practical deployment of the network. Topology information helps the administrators monitor, debug and perform proactive control and management. Obtaining topology for dense wireless sensor network is expensive due to limited bandwidth and contention for the wireless channel. In this paper, we consider the problem of efficiently obtaining network topology where the locations and associated node identifiers are known by the central controller (sink). Although the locations of the nodes are known, the connectivity between any two nodes cannot be determined by distance information alone due to irregular radio coverage and obstacles. We propose a location-aided topology discovery algorithm (LAD) that requires $O(\log(|M|))$ data units per node in common cases, and $O(|M|)$ data units per node in the worst case, where M is the set of directly communicable neighbors of a particular node. Simulation results show that the overhead reduction ranges from 70% to over 95%.**

I. Introduction

Network topology refers to the underlying physical or logical connectivity among the network devices. It is one of the fundamental attributes of network management in traditional IP networks, as it is a prerequisite to many critical network management tasks, including reactive and proactive resource management, server siting, event correlation, and root-cause analysis [1].

The role of network topology continues to be important in wireless sensor networks (WSNs). Firstly, topology information is crucial for the network administrators to monitor and debug [2] various sensor protocols, in particular, the protocols that are topology-dependent, such as topology control, coverage control, and routing. In these protocols, knowing the connectivity information (neighbor table) of each node inside some region is the most direct way in pinpointing root-cause problems of that region. Secondly, the connectivity statistics can also aid in sensor network protocol design as well as parameter tuning [3], such as computing mean topological density, studying the impact of link asymmetry, evaluating geographical routing algorithms, and accessing behaviors of algorithms that depend on spatial correlation.

However, collecting the network topology in large and dense [4] sensor networks can be costly for small and tiny sensor nodes, which limits the potential usefulness of topology information. Previous topology collection protocols try to reduce such cost either by reducing the number of nodes who will send their neighborhood information to the central controller, or by limiting the number of neighbors sent at each node [5], [6], [7]. Both methods result in significant loss of neighborhood information and make it less useful in the applications mentioned above.

In this paper, we focus on the problem of efficiently retrieving network topology information at the central controller. We call the process of retrieving network topology or network connectivity *Topology Discovery*. We propose a Location-Aided Topology Discovery (LAD) protocol which allows the central controller to discover the complete network topology efficiently.

In LAD, nodes send abstracted neighborhood information to the server in multiple rounds. By making use of these partial neighborhood information, the central controller requires $O(\log(|M|))$ data units[1] per node in common cases or at most $O(|M|)$ data units from each node to construct the entire topology. Specifically, the overhead is $(1 + c)$ data unit per node in the best case, and $(\log_{\frac{1}{1-\alpha}} \frac{|M|}{t} + |M| + 1 + c)$ data unit per node in the worst case, where $|M|$ is the number of directly communicable neighbors of a particular node, α, t are constants, and c is a small constant which represent some fixed overhead. Simulation results show that the overhead reduction ranges from 70% to over 95% in average.

The rest of the paper is organized as follows. In Section II, related work is presented. Section III presents the techniques in reducing communication cost as well as the design and analysis of LAD. Section IV shows the simulation and testbed evaluation results. Conclusion is drawn in Section V.

II. Related Work

In [2], N. Ramanathan et al. propose a sensor network debugging system called Sympathy which requires connectivity information from the sensor nodes for root-cause cause analysis. The authors simply assume that each sensor node periodically sends its neighbor table to central controller. Since the testbed on which they experiment is small, this may not be a serious issue.

In [5], the authors propose TopDisc algorithm for sensor networks with its applications to network management. The algorithm finds a set of distinguished nodes (minimum dominating set), using their neighborhood information to construct

This work is supported by National University of Singapore under research grant R-252-000-265-112.

[1]We use log to represent the logarithm of base 2 in the paper.

978-1-4244-2075-9/08/$25.00 ©2008 IEEE

the approximate topology of the network. Only those distinguished nodes will reply back to the topology discovery probes, thereby reducing the communication overhead of the process. [6], [7] propose a multi-resolution topology retrieval protocol which makes a tradeoff between topology details and resources expended. The algorithm makes use of Minimal Virtual Dominating Set (MVDS) to define the distinguished nodes that will response the topology probes. By adjusting the virtual radius of MVDS, the multi-resolution can be achieved.

In [8], the authors propose a mesh based topology retrieval algorithm with slow moving nodes in wireless ad hoc networks. Each node has multiple parents to which the local communicable neighbor information will be sent, and thus the algorithm is more error resilient.

Unlike previous topology discovery protocols, we focus on how the neighborhood information can be efficiently represented. Our proposed LAD makes use of location information of the sensor nodes (which is not presented in traditional Internet world) and effectively reduces the cost to transmit the neighborhood information to the central controller.

III. Location-Aided Topology Discovery (LAD)

A. Problem Formulation

In this work, we assume that each sensor node can be uniquely identified and has a unique identifier. Each sensor node knows its neighbor set M with size $|M|$. The problem can be formulated as, how can central controller know the complete neighbor sets of all the nodes in the region of interests using least amount of overhead (in terms of data transmitted)? This is generally a hard problem in distributed sensor networks. We reduce the problem further to how the neighbor set of each node can be optimally represented without or with little information loss. Note that we assume that links can be symmetric or asymmetric.

B. Location Information and Topology Discovery

Without any prior information at central controller, the problem is equivalent to optimally compress the neighbor set M at each sensor node. The minimum information required to identify $|M|$ objects from a set of $|T|$ object is theoretically bounded by $\log \binom{|T|}{|M|}$, where T is the set of all sensor nodes in the network. Since $\log \binom{|T|}{|M|} \geq |M|(\log |T| - \log |M|)$, this can still be very large when $|M|$ is large and $|T| \gg |M|$.

However, one can observe that location information can play an important role in reducing the size of neighborhood information. The two nodes that are closer to each other are more likely to be neighbors. As a simple example, with unit disk communication model, a node only needs to send to the central controller the ID of the neighbor node who is farthest away from itself. Since all nodes closer than the indicated node can be heard, the central controller can easily deduce all the neighbors by searching through the distances among the sensor nodes. The example is shown in Figure 1(a). When node A sends the ID of node B to the central controller, the central controller just needs to find all the nodes that are within the range $|AB|$, and assume that they are directly communicable to node A.

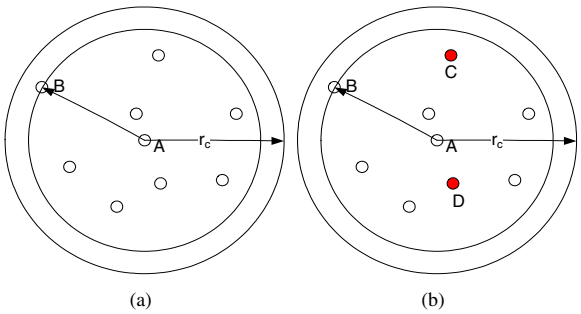

Fig. 1. Location Information and Topology Discovery: (a) Under unit disk model, given location information on each node and the farthest neighbor node B, it is easy to compute the whole neighbor list of node A (b) Given B, the neighbor list of A cannot be obtained because there might be nodes (C and D) that cannot communicate to A, within range $|AB|$

In this work, we assume that the location of each sensor node is known via localization algorithms, and each node knows the location information of all its connected (one-hop) neighbors. We also assume that the location information of each node is known to the central controller. This assumption is not unrealistic because most sensor networks do require some form of localization. We do not require accurate location information. LAD is robust to localization error and will work correctly as long as the nodes and central controller have the same location information.

The simple example shown in Figure 1(a) is purely based on unit disk model, and it does not work in realistic environments. In a real sensor network, not all the nodes that are located within the maximum range of node A can communicate directly to node A. This can be due to reasons such as obstacles, environment and fading effects, and possible localization errors. This is shown in Figure 1(b). Hence, given a node's farthest neighbor node, all of its neighbors must locate within the maximum distance, while the reverse is not true. We try to solve this problem in the next section.

C. Hashing of Node ID

With the help of location information, for each node, the central controller is able to identify a set of nodes N with size $|N|$ who are possible neighbor candidates. N is a superset of M. Although $|N|$ may be much smaller than the total number of nodes $|T|$ in the network, the central controller still cannot completely identify M using the location constraints. Due to the fact that the sensor node itself is not aware of its neighbor candidate set N in advance and thus it cannot efficiently compress its neighbor set M based on superset N.

Hashing is one of the standard ways in solving such problems. Using hashing, elements in the set M can be represented by a much smaller size data. Each node sends the hashed value to central controller and as long as the collision probability is small, the central controller is likely to be able to test which node in N shall be in M. The initial data required at the central controller from each sensor node are thus: ID of the

node with largest distance to itself and $h(M)$, where $h()$ is the hashing function.

The number of searches required at the central controller is given by,

$$\sum_{m=0}^{|N|} \binom{|N|}{m} = 2^{|N|}, \tag{1}$$

which is exponential to $|N|$.

Assume that the hashing function h is able to uniformly distribute the hash values onto the hash space 2^b, where b is the number of bits of the hashed value, the collision probability is given by,

$$1 - \left(1 - \left(\frac{1}{2}\right)^b\right)^{2^{|N|}} \approx 1 - e^{-2^{(|N|-b)}}. \tag{2}$$

It can be seen that b has to grow linearly with $|N|$ to keep the collision probability constant. b is counted into communication cost since the hashed value has to be sent to the central controller. In practice, due to the fact that each node cannot know $|N|$ in advance, it is hard for it to keep b at the optimal value and keep a constant collision probability.

Another design choice is by letting each node send one additional information $|M|$. The number of searches required at central controller will then be $\binom{|N|}{|M|}$ which is smaller than $2^{|N|}$. The increase of communication overhead will be small because in practice $|M|$ is not a very big value.

The probability of collision is given by,

$$1 - \left(1 - \left(\frac{1}{2}\right)^b\right)^{\binom{|N|}{|M|}} \leq 1 - \left(1 - \left(\frac{1}{2}\right)^b\right)^{\binom{|N|}{\lceil \frac{|N|}{2} \rceil}} \tag{3}$$

The size of the hash value b required to keep the collision probability to be smaller than γ is given by,

$$b \geq \log \left(\frac{1}{1 - (1-\gamma)^{\frac{1}{\binom{|N|}{|M|}}}} \right) \tag{4}$$

D. Set Partitioning: Tradeoff Between Communication and Computation Cost

The values of $|N|$ and $|M|$ determine the computational cost at the central controller, as well as the hashing space required to remain a small collision probability. When $|N|$ or $|M|$ increases, $2^{|N|}$ or $\binom{|N|}{|M|}$ may increase quickly to a value that is not feasible for the controller to perform an exhaustive search.

This can be solved by a set partitioning technique. A large set can be divided into smaller subsets and solved individually. In our application scenario, each node can sort its directly communicable neighbors in some order and divide the neighbor set M into K subsets M_1 to M_K. The IDs of the separation nodes, the number of nodes in each set $|M_k|$ and the hash value $h(M_k)$ will be sent to the central controller (depends on the design choice). The central controller can sort the set N in the same order, divide N into K subsets N_1 to N_K according to the IDs of separation points, and search the M_1 to M_K individually. The example is shown in Figure 2.

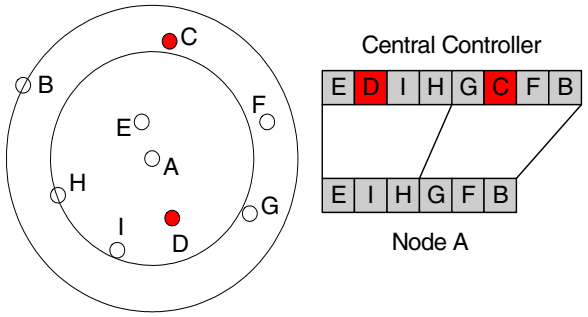

Fig. 2. Both controller side and local side sort the nodes according to ascending order in distance to A, A locally divides the set M into 2 subsets, for each subset, the problem now actually becomes simpler for central controller (computation cost is $2\binom{4}{1} = 8$ now, which is smaller than $\binom{8}{2} = 28$)

The total amount of searches required at central controller is now given by,

$$S = \sum_{k=1}^{K} \binom{|N_k|}{|M_k|} \tag{5}$$

where $|N_k| \geq |M_k|$ for all $k \in [1, K]$, and $\sum_{k=1}^{K} |N_k| = |N|$, $\sum_{k=1}^{K} |M_k| = |M|$.

The optimal total communication cost will be,

$$C = \sum_{k=1}^{K} (\log |N| + \log |M_k| + b_k), \tag{6}$$

where $\log |N|$ represents the size of node ID, $\log |M_k|$ represent the size of data to hold the number of nodes in M_k, and b_k is the size of $h(M_k)$. If $|M_k|$ is not sent to the central controller, $S = \sum_{k=1}^{K} 2^{|N_k|}$, and $\log |M_k|$ will not be counted in Equation 6.

The problem can then be formulated as to find a suitable division scheme such that C is minimized, given that $S \leq S_{\max}$, where $S_{\max}$ is the maximum number of searches can be performed at central controller for each node. However, since the local nodes do not know the set N, this is generally a hard problem.

In LAD, we propose a simple algorithm that works well in practice. We observe from our testbed measurements that, nodes that are farther away from each other have less probability to be neighbors. In fact if the distance between two nodes are within some threshold, the probability that they are neighbors is very high. This is not a surprising result and it has been shown by many other measurements like in [9]. The results show that most of the nodes that are close to a node A tend to be neighbors of A, i.e., the value of $|M|$ is very close to $|N|$. However, as the distance to A increases, more nodes may start losing connections to A.

Based on this observation, we propose a simple scheme to divide the set M described as follows. Assume that the neighbors are sorted in ascending order of distance, each node

divides the first α potion of M into M_1, where α is a value between 0 and 1. The node then divides α potion of the remaining set into M_2, and so on. The process stops until the node finds out the number of neighbors remaining is smaller than some threshold. The idea here is to make the largest progress in M_1 because most of the nodes in N_1 may be in M_1 and the number of searches required can be small. Thus we make $|M_1|$ largest among all the subsets to save the communication cost. As the distance increases, we let each node divide less neighbors to save the computation cost at central controller. Moreover, LAD only requires each node to send the neighbor ID that is farthest away from itself, total number of neighbors, and the hash values for each subset (it does not require IDs the subset separation node and the number of neighbors in each subset).

The selection of α should be based on the distribution of disconnection patterns. The choice and effect of α will be studied by simulations.

E. LAD Algorithm Description and Analysis

In this section, we describe our LAD algorithm in details. The LAD algorithm proceeds in two phases. In phase 1, each node divides the subset and hashes them individually. The node will then send the data to central controller. Upon receiving the data packets from sensor nodes, the central controller can apply searches on the neighbor set for each node. In phase 2, using the information obtained in phase 1, the central controller sends query to specified nodes to refine the topology information (for those nodes that have some subsets which cannot be determined by central controller due to either searching time is too long or there is a collision). Note that in general, the central controller does not have to obtain the entire topology but can stop any time when it is sufficiently accurate according to application needs.

1) Phase 1: In phase 1, each node tries to calculate the hash value and send it to the central controller. The algorithm in phase 1 is shown in Figure 3.

The information sent to central controller is the neighbor ID that is farthest away from itself, total number of neighbors, and the hash values for each subset. When the central controller receives these messages from a node x, it first calculates the set N, then it sorts N in distance order to node x. The search for match of the first hash value is performed starting from the first $\alpha|M|$ nodes in N. The controller sets an upper bound for the amount of searches to be performed and will proceed to phase 2 if no match is found. Note that the controller does not need to search all possibilities for collision. Given the very small collision probability, it is very likely the first match is the solution. If a match is found, matching will be performed for the next hash value, and so on. For every match made, the amount of uncertain or search space goes down substantially.

Based on the computation capability and trade-off desirable, we define the parameter α and t_1, where α is the progress factor and t_1 is number of remaining elements at which phase 1 can stop. Using the location information, node x sorts the nodes in M in ascending distance from x. The number of hash values computed is given by the following Theorem.

Location-Aided Topology Discovery Algorithm

Sensor Node x***:***
Sort the neighbor list M_x in the order of increasing distance to x
let $m_x = |M_x|$, $r = m_x$
let $start = 0$, $end = 0$
Append packet with the farthest neighbor node ID, m
while $r > t$
  let $start = end + 1$, $end = start + \alpha r$
  let $hash = h(M_x[start : end])$
  Append packet with $hash$
  let $r = r - \alpha r$
end while
let $start = end + 1$, $end = m_x$
let $hash = h(M_x[start : end])$
Append packet with $hash$
send(packet)

Central Controller (Upon Receiving from Node x***):***
Calculate the set of possible neighbors of node x, save in N_x
Sort N_x in the order of increasing distance to x
Search N_x sequentially for a match of each $hash$ in packet
if running time is too long or there are possible more than one solutions
then stop and wait for phase 2
else Update neighbor list of node x
end if

Fig. 3. Description of Location-Aided Topology Discovery Protocol: Phase 1

Theorem 1: The number of hash values computed in phase 1 of LAD is given by,

$$k = \left\lceil \log_{\frac{1}{1-\alpha}} \frac{|M|}{t_1} \right\rceil + 1 \tag{7}$$

Proof: Assume the number of hash values is K, the set M of size $|M|$ is divided into K subsets with $(K-1)$ dividing point. It is easy to see that,

$$\frac{t_1}{(1-\alpha)^{K-1}} = |M|$$

Thus, $K = \left\lceil \log_{\frac{1}{1-\alpha}} \frac{|M|}{t_1} \right\rceil + 1$. ■

2) Phase 2: If the topology cannot be computed completely in phase 1 for a node x, phase 2 is required. In phase 2, the central controller first identifies the point where the searching requires too much time, say n_i. It then sends request message for the hash value from n_i to n_{i+t_2}. Of course the node x may not have the values n_i and n_{i+t_2}, but they can be defined as distance to x. Because M and N are both sorted in distance order, the node x then identifies m_j to $m_{j+t_2'}$ that are in the distance ranges, and sends back the new hashed value to the central controller again.

The central controller chooses the value of t_2 such that the computation cost is reasonable even in worst case $\binom{t_2}{\lceil \frac{t_2}{2} \rceil}$. Once the response from the node x comes back, it updates the neighbor list. The new known neighbors of x is likely to reduce the search complexity of the remaining hash values

from phase 1. If the controller still cannot find all the members of M, phase 2 continues.

In the worst case, $\frac{2|N|}{t_2}$ data units are supposed to be needed to completely find the topology (considering information exchanged between central controller and x). However, if $\frac{2|N|}{t_2}$ is larger than $|M|$, then the controller can simply ask the node to transmit all its $|M|$ neighbor. Hence, the worst case of Phase 2 is $|M|$ in terms of communication cost.

3) Algorithm Analysis: Number of data units sent in phase 1 is $\left\lceil \log_{\frac{1}{1-\alpha}} \frac{|M|}{t_1} \right\rceil + 1$ which is $O(\log |M|)$. If $\alpha = 1$, the number of data units required is only 1 (hash only 1 time). In phase 2, the worst case is $|M|$ data units. Hence, the worst case for the combination of phase 1 and 2 is $O(|M|)$ and the best case is O(1). In simulation, we show that the protocol performs much better than worst case in practice.

IV. Evaluation

A. Simulation Setup

The simulation is based on the following setup of the network. The world size is 16 by 16 with node density varying from 5 to 50 (uniform distribution), communication radius is 1. Thus, on average each node has 15 to 150 neighbors. In the simulation, each node has an unique 4-byte ID, the hashing result is in 2^{32} space and also occupies 4 bytes. The server will search all combinations for a match of the hash value, and if the search space is too large (in the simulation larger than 10^6), the server will give up searching and enter phase 2 for that node.

Disconnection probability among nodes are defined by function $f_x(s)$, where s is the distance to node x and $f_x(s)$ is the probability that a node at distance s away from x is disconnected from node x. The average disconnection probability is then $\int_0^1 2s f_x(s)\, ds$. We test the LAD protocol with both small disconnection probability as well as large disconnection probability. We define two different $f_x(s)$ in the simulation. For relatively small disconnection probability,

$$f_x(s) = \begin{cases} (s - d_s)^2 & \text{if } s \geq d_s, \\ 0 & \text{if } s < d_s. \end{cases} \tag{8}$$

Where d_s is an adjustable parameter. This is similar to a quasi-unit disk model. The average disconnection probability is given by $p = \int_{d_s}^1 2s(s - d_s)^2\, ds$.

For large disconnection probability,

$$f_x(s) = s^\beta \tag{9}$$

β determines where most of the disconnections are and if β is larger, there will be more disconnections at larger s.

B. Simulation Results for Small Disconnection Probability

The performance of LAD with small link disconnection probabilities are studied. The disconnection probability is given by Equation 8.

The simulation results of LAD with d_s is 0.5 and 0.6 are plotted in Figure 4.

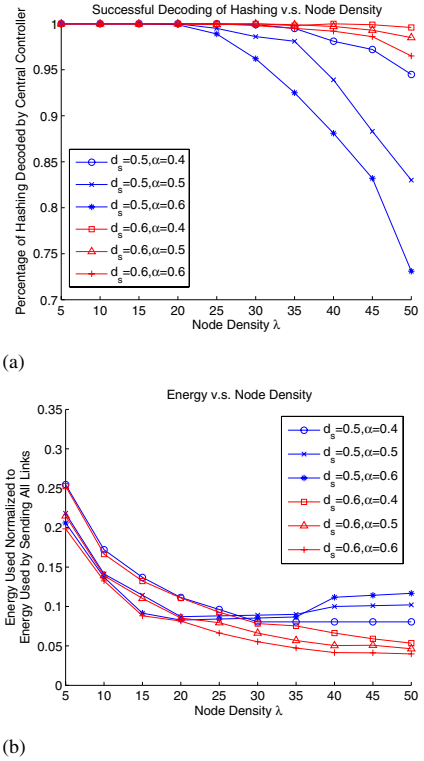

Fig. 4. Performance of LAD with Small Probability of Link Disconnection) (a) Average percentage of nodes whose neighborhood information is successfully decoded by central controller in Phase 1 (b) Average normalized total energy cost in Phase 1 and Phase 2

Figure 4(a) gives the information about the percentage of nodes whose neighbor set are successfully known by the central controller after the phase 1 of LAD protocol. It is not surprising to see that when α is smaller, the successful decoding probability becomes larger. When node density is small (e.g., $\lambda \leq 20$), almost all the nodes are able to send their neighborhood information to the central controller in phase 1.

Figure 4(b) shows the total energy cost of LAD algorithm (including both phase 1 and phase 2) for the controller to know the complete network topology, normalized to the total energy cost of letting each node send its raw neighborhood information (a set of neighbor IDs). The energy cost is the sum of both phase 1 and phase 2.

Initially when density is small and, large α will give a smaller energy cost because the number of hash values will converge more quickly than small α. As density increases, large α will cause more nodes to enter phase 2 (their neighbor sets cannot be determined by server in phase 1), which causes the increase of total energy. Thus, although a larger α may reduce the number of hash values in phase 1, with high node density, it may end up with more nodes enter phase 2 and uses more energy in phase 2 (the $d_s = 0.5$ plots in Figure 4(b)). The maximum energy reduction in the plot is 95%.

One can also observe that the performance of LAD is better

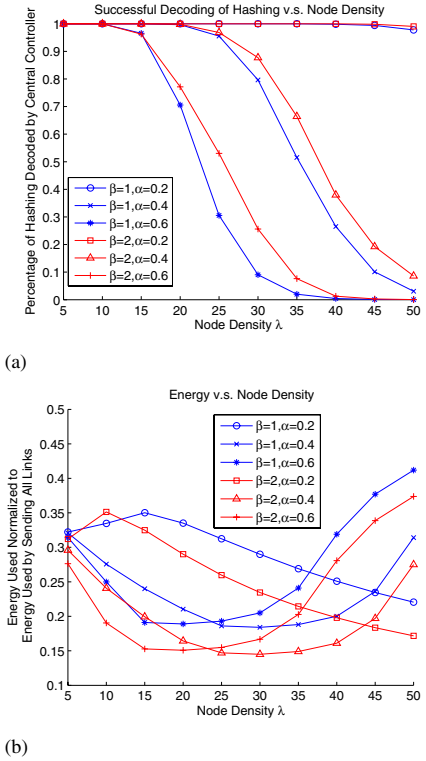

Fig. 5. Performance of LAD with Large Probability of Link Disconnection (a) Average percentage of nodes whose neighborhood information is successfully decoded by central controller in Phase 1 (b) Average normalized total energy cost in Phase 1 and Phase 2

when the connectivities among sensor nodes are good. The performance for the case $d_s = 0.6$ is always better than the case $d_s = 0.5$.

C. Simulation Results for Large Disconnection Probability

The large distribution probability is defined by Equation 9. With $\beta = 1$ or 2 (disconnection probability 67% or 50%), the simulation results are shown in Figure 5.

Compare different values of β, higher β (lower disconnection among neighbors) will also result in slightly better performance in number of decodable nodes as well as the energy cost. On the other hand, similar to the cases in previous section, when node density increases, the probability of successful decoding drops, but in a much faster rate (many drops to 0 eventually). This is mainly because larger disconnection probability causes higher computational cost to decode the hash values sent by the nodes. The central controller finally has to give up the exhaustive search and announce failure of decoding the neighbor information of a particular node.

The energy reduction in the presence of large disconnection probabilities is smaller than the energy reduction in small disconnection probability. The best energy reduction is about 85% as shown in Figure 5(b). At the worst case (with both high node density and disconnection probability), the minimum energy cost is still around 30% of sending raw neighbor information, which represents 70% of energy cost reduction.

D. Testbed Evaluation

We implement the algorithm on our sensor network testbed with 34 motes (Mica2 and Mica2dot) installed in our building. We assign locations to each of the node manually. The node ID size is 2 bytes and hash value is also 2 bytes. We use $\alpha = 0.5$ in the experiment. The average amount of data transmitted at each node using LAD is only about 5% of sending all the node IDs and has almost the same cost comparing to sending bitmaps of node IDs (represents each node as a bit in a bit string of size $|T|$). However, bitmap technique can only be applied in small networks. We show the effectiveness of our protocol in both small and large networks.

V. Conclusion

In all, in this paper we propose a location-aided topology discovery protocol called LAD. Under the help of location information, LAD allows the central server to discover the complete network topology with much less data size than letting each node send its own raw neighbor information. Specifically, the overhead is $(1 + c)$ data unit per node in the best case, and $(\log_{\frac{1}{1-\alpha}} \frac{|M|}{t} + |M| + 1 + c)$ data unit per node in the worst case. Simulation results show that the overhead reduction ranges from 70% to over 95% to retrieve the complete network information.

References

[1] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz, "Topology discovery in heterogeneous IP networks: The NetInventory system," *IEEE/ACM transations on networking*, vol. 12, no. 3, 2004.

[2] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys'05)*, San Diego, California, USA, 2005.

[3] A. Cerpa, N. Busek, and D. Estrin., "SCALE: A tool for simple connectivity assessment in lossy environments, Technical Report CENS-21."

[4] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for 'smart dust'," in *Proceedings of the 5th ACM Conference on Mobile Computing and Networking (MobiCom'99)*, 1999.

[5] B. Deb, S. Bhatnagar, and B. Nath, "A topology discovery algorithm for sensor networks with applications to network management," in *Proceedings of the IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, USA, 2002.

[6] ——, "Multiple-resolution state retrieval in sensor networks," in *Proceedings of First IEEE Workshop on Sensor Network Protocols And Applications (SNPA)*, Anchorage, 2003.

[7] ——, "STREAM: Sensor topology retrieval at multiple resolutions," *Journal of Telecommunications, Special Issue on Wireless Sensor Networks*, 2004.

[8] R. Chandra, C. Fetzer, and K. Högstedt, "A mesh-based robust topology discovery algorithm for hybrid wireless networks, Technical Report."

[9] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin, "Statistical model of lossy links in wireless sensor networks," in *Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05)*, California, USA, 2005.