A Topology Based Localization in Ad Hoc Mobile Sensor Networks

Sridhar Gangadharpalli, Uday Golwelkar, and Sridhar Varadarajan

Applied Research Group, Satyam Computer Services Ltd.,
14 Langford Avenue, Lalbagh Road, Bangalore 560 025 INDIA
{sridhar_gangadharpalli, sridhar, uday_golwelkar }@satyam.com

Abstract. Localization in an ad hoc mobile sensor network is an important requirement as most of the applications that use sensor data require sensor location information to complete the processing. A typical sensor network has over hundred to thousand sensor nodes, and considering the size and cost of a sensor, using only GPS for localization is not very attractive. The mobility of sensor nodes could lead to network topologies wherein accurate computation of absolute position of all the sensor nodes may not be possible. In this paper, we propose a topology based localization approach that suggests a best possible approximate position for sensor nodes for which computation of exact absolute position is not possible. We have identified four basic topological configurations that help compute position with varied degree of accuracy. These atomic configurations have been identified keeping in mind the simplicity of the computational procedures associated with these configurations. In order to put less demand on a computational capability of a sensor node, we suggest that only a pre-defined number of sensor nodes are compute-enabled (c-nodes) in the sense that they have adequate computational power. Similarly, only a pre-defined number of sensor nodes are GPS-enabled. In such a sensor network, the distributed computation of localization is achieved by distributing the computational requirements of individual sensor nodes across the c-nodes. Each sensor node strives to improve its localization by constantly monitoring its neighborhood and requesting an associated c-node to recompute position whenever neighborhood topology changes. We provide some initial results that bring out the merits of the proposed approach.

1 Introduction

Recent technological advancements and availability of wireless devices have increased the demand for self-organizing networks without the need for any dedicated infrastructure. These ad hoc networks consists of multiple nodes, with each node self-sufficient in terms of communication and computation powers, interact with each other in a cooperative way to address issues at network level and in interacting with a central station. A sensor network with wireless capability is a constrained wireless ad hoc network with limited power, communication, and computational abilities. Further,

R. Battiti, M. Conti, and R. Lo Cigno (Eds.): WONS 2004, LNCS 2928, pp. 16–28, 2004.

the sensor network tends to be large in size with hundreds and thousands of sensor nodes. An ad hoc sensors network is static if the nodes that are part of the network, after an initial configuration, do not change their position. On the other hand, in the case of an ad hoc mobile network, the nodes of the network move arbitrarily, independent of each other, resulting in dynamic and ad hoc changes in the network topology. It is a challenging task to define a self-organizing network in the case of a mobile ad hoc network.

A smart sensor network [11] consists of a number of sensors spread across a geographical region and each sensor node has adequate intelligence. Such sensor networks are deployed in a variety of application scenarios such as (a) military sensor networks to detect the presence of hazardous material; (b) environmental sensor networks to detect and monitor environmental changes; (c) traffic sensor networks to monitor vehicle traffic on highways; and (d) surveillance sensor networks to monitor for security purposes. In each of these cases, it is required for a smart sensor node to communicate the sensed data (such as temperature and atmospheric pressure) to a central station for intelligent processing of aggregated data from multiple sensors and decision making. An important attribute of the data communicated by a sensor node is its current location that is essential in many of the applications such as traffic monitoring or surveillance.

GPS [4] is well-known approach for obtaining an absolute position of a node. When a receiver is outside a constellation of transmitters, standard iterative techniques may not converge to a correct solution. In this case it is required to use the known solutions to pseudo-range equations. [10] describes a five dimensional optimization procedure derived from the pseudo-range equations to compute position in the absence of navigation data. However, from the point of view of deploying this technology in a large sensor network where each sensor node is constrained by power, size, form factor, and cost factor, one can only selectively deploy this technology in a sensor network.

Localization refers to the problem of computing the position of a sensor node in an ad hoc network. In a large sensor network, it is not possible to configure the position of a sensor node even in the case of a fixed ad hoc network as the process of installation of sensor nodes might be just randomly dropping of the sensors over a region of interest. The position of a sensor node can be either an absolute position or a relative position. However, it is useful to determine the absolute position as relative position would have to be redetermined if there is a topology change even though the node under consideration might not have moved. Being able to know their absolute position is one of the important characteristics of the nodes of a self-organizing network.

Localization approaches depend on some sort of communication between anchor points (or reference points) and the node whose location needs to be determined. There has been significant research in studying location identification and some of these results are briefly described in the following. Bulusu et al [2] describe an approach wherein multiple nodes in a network that form a mesh serve as reference points and transmit periodic beacon signals containing their reference positions. From the beacon signals received by a node from a set of reference points, the node localizes to the region that coincides with the intersection of the connectivity regions of the set of reference points. Doherty et al [3] describe a method for estimating unknown node positions in a sensor network based on connectivity-induced constraints. Feasible solutions to the position estimation are determined using convex

optimization. [6] describes a self-localization method based on time-of-arrival and direction-of-arrival measurements by a subset of sensor nodes from a number of source signals placed at unknown locations. The method is for solving self-calibration problem with minimum number of sensor nodes and sources and provides an initial estimates for an iterative descent computation needed to obtain maximum likelihood calibration parameter estimates. [7] describes Ad hoc Positioning System that is a distributed, hop by hop positioning algorithm and works as an extension of both distance vector routing and GPS positioning in order to provide approximate location for all nodes in a network where only a limited fraction of nodes have self-location capability. [5] describes an algorithm in the context of a distributed sensor networks using which each sensor node determines its position in physical space based on their location in the network topology. The algorithm is based on determining, for each sensor node, the number of hops it is away from each of the basis nodes (those nodes that are aware of their location) and converting these hop-based distances into Cartesian coordinates. [9] describes a distributed technique for achieving fine-grain location awareness based on a limited fraction of beacons. The technique called as ad hoc localization system enables nodes to dynamically discover their own locations through ranging and estimation processes.

[8] describes a distributed algorithm for determining the positions of nodes in an ad-hoc, wireless sensor network in two phases: the startup-phase addresses the issues related to sparse availability of GPS-enabled nodes and uses a cooperative mechanism to spread location information of the anchor nodes throughout the network; and refinement phase addresses the issues related to reducing the error in initial position estimates.

[1] describes efficient algebraic tools for solving explicitly nonlinear geodetic problems such as GPS pseudo-ranging based on the algebraic techniques of Grobner bases and Multipolynomial resultants. The problems of localization can also be posed as a solution of a system of quadratic equations and the approaches suggested in [1] can be used to solve these pseudo-range equations.

In this paper, we propose a topology based localization approach. Briefly, we consider a sensor network in a field, with each sensor node having the property of wireless communication and low mobility. Some of the sensor nodes are GPS-enabled and are called as *g-node*s. Similarly, some of the sensors are compute-enabled and these nodes are called *c-node*s. Given this scenario, our objective is to achieve self-localization in each of the sensor nodes. Figure 1 depicts a typical sensor network. An *ng-node* is either a c-node or w-node.

2 Atomic Configurations

The process of localization is to be able to assign the absolute position to a node in a sensor network. Such a position is necessary for an intelligent sensor node to undertake location-specific sensing. Self-localization refers to a process wherein a node is able to establish its position based on the neighborhood information. In order to achieve self-localization, we propose to define a few topology based configurations

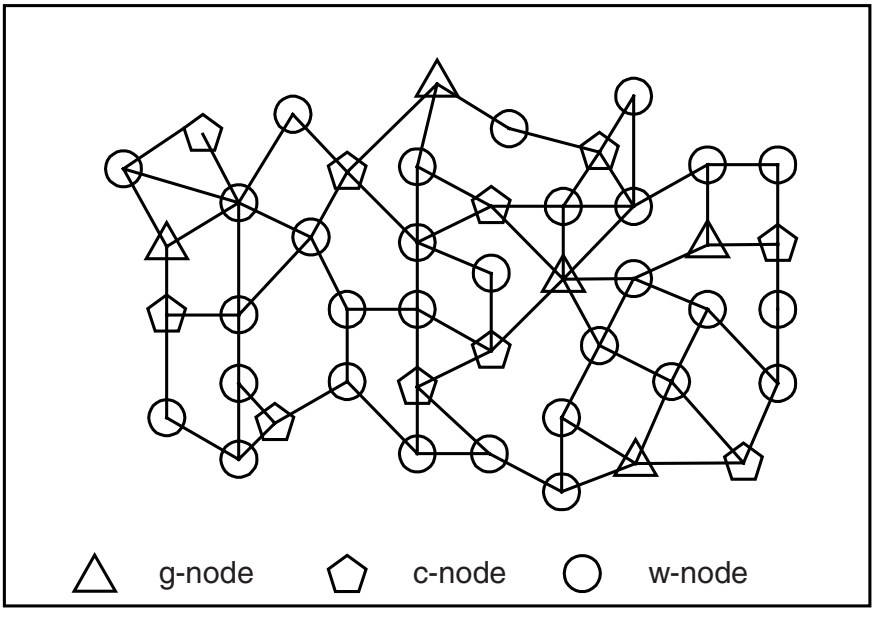

Fig. 1. Typical Sensor network. An ng-node is either a c-node or w-node

and embed a configuration identification procedure in each of the nodes. These configurations are called as atomic configurations as they are, in some sense, minimum amount of information that is required to localize a node within some bounds. The reason for identifying multiple atomic configurations is that in an ad hoc mobile sensor networks, depending on the various factors and the intended application, not all sensor nodes may be equipped with GPS capability. Addition of such a GPS feature would make a sensor node not only costly but also bulky and stresses the battery power. On account of the fact that only a few nodes are GPS-enabled and with sensor node mobility, there are certain topological possibilities in which exact computation of position of all the sensor nodes is not possible. In order to address such a situation, additional configurations have been identified so that where exact position identification is not possible, an approximate position could be assigned. An effort has been made to suggest atomic configurations keeping in mind computational simplicity. While position recomputation could be on rare occasions in a fixed sensor network, it is required to recompute the position quite often in the case of a mobile sensor network and this recomputation frequency not only depends on the self-mobility of a sensor node but also the mobility of the other nodes in the sensor network.

In the following, we describe four atomic configurations giving details such as configuration identification mechanism and position computation procedures. In the following, note that a g-node has been used to stand for a GPS-enabled node or location-aware node.

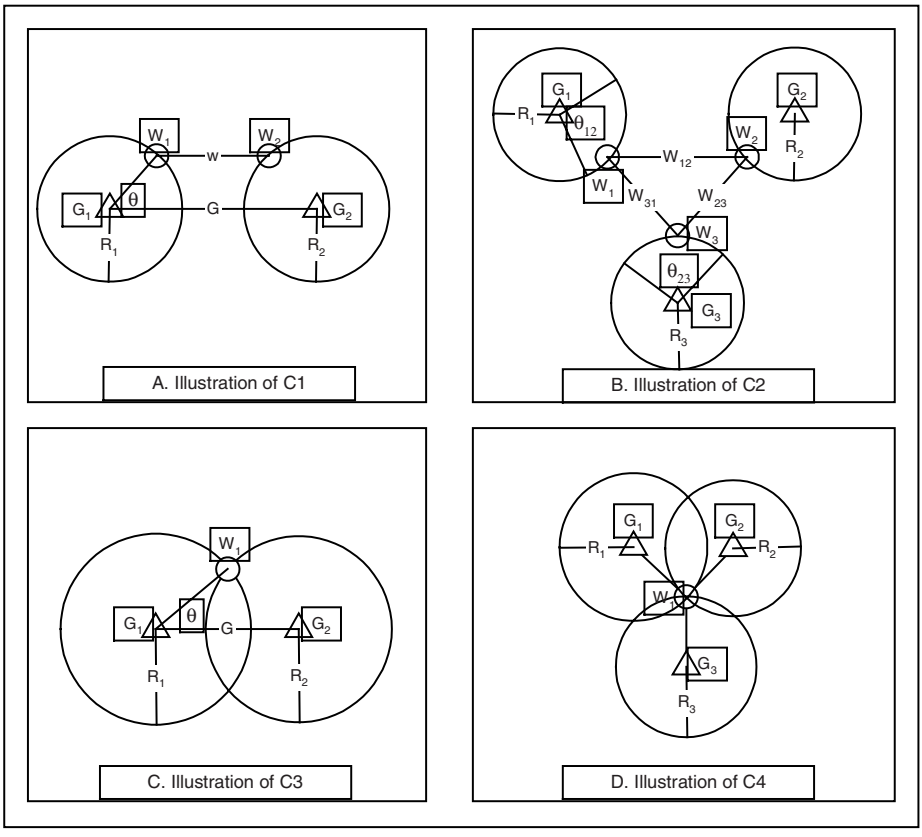

Fig. 2. Illustration of atomic configurations. A,B,C and D respectively illustrate the atomic configurations C1, C2, C3, and C4.

Configuration 1 (C1): A pictorial depiction of C1 is shown in Figure 2. Note that, in this configuration, there are two g-nodes and two w-nodes and W_1 is only within the range of G_1 and W_2 while W_2 is only within the range of G_2 and W_1.

Configuration 2 (C2): A pictorial depiction of C2 is shown in Figure 2. Note that this configuration is an improvement over C1 in terms of the accuracy of position computation. In this case, there are three g-nodes (G_1, G_2, and G_3) and three w-nodes (W_1, W_2, and W_3) with the restriction that W_1 is only within the range of G_1, W_2, and W_3 and similar restrictions apply for W_2 and W_3.

Configuration 3 (C3): A pictorial depiction of C3 is shown in Figure 2. Note that this configuration is an improvement over C1 and C2 in terms of the accuracy of position computation. In this case, a single w-node (W_1) is within the range of two g-nodes G_1 and G_2 thereby improving the position computation accuracy.

Configuration 4 (C4): A pictorial depiction of C4 is shown in Figure 2. Note that this configuration is an improvement over C1 through C3 in terms of the accuracy of position computation. In this case, one w-node (W_1) is within the range of three g-nodes (G_1,G_2, and G_3) resulting in a well-know configuration.

```
Configuration Identification
// Executed by w-node
// NeighborhoodData contains all possible instances of one or more identified configurations

{ Identify all the neighboring nodes (n-list) (1 hop away) and compute range from itself (w-node) to each of
the identified neighboring node;

// each neighboring node is within the communication range of w-node

        Let GN be the number of g-nodes in n-list;
        If GN == 0 { return null set;}
        If GN >= 3 { select three g-nodes from n-list;
                Form NeighborhoodData with configuration id as C4; return NeighborhoodData;    }
        If GN == 2 { select two g-nodes from n-list;
        Form NeighborhoodData with configuration id as C3; return NeighborhoodData;    }
        Let WN be the number of w-nodes in n-list; Let w-list be n-list without g-node;
        While (w-list is not empty)

        { Identify two w-nodes in w-list such that they are within the range of each other and each of the
        two w-nodes has only one g-node within its range and let w2 be the list of these two w-nodes;
                If w2 is not empty { Append w2 with configuration id as C2 to NeighborhoodData;
                        Remove w2 from w-list; Continue; }
        Identify one w-node in w-list such that it has only one g-node within its range and let w1
        be the list of this w-node;
        If w1 is not empty { Append w1 with configuration id as C1 to NeighborhoodData;
                        Remove w1 from w-list; Continue;        }

        Return NeighborhoodData;
        }
}
```

Fig. 3. Describes the steps involved in identifying the configurations in a sensor network. Note that this algorithm is executed by each w-node and hence, the configuration identification is from the point of view of a w-node.

Figure 3 describes the procedure for identifying the atomic configuration of the sub network.

In the following, we describe a computational procedure for each of the identified four atomic configurations.

In Algorithm C1, θ is computed by applying cosine rule to the triangle formed by W_1 with G_1 and G_2 as shown in Fig. 2. Observe that in the algorithms C1 through C3, the position of a w-node is not determined uniquely and hence, each of these algorithms provide a narrowed search space for computing the possible positions. Further, the multiple instances of the configurations C1 and C2 help in progressively reducing the solution search space. Also, multi-hop neighbors of a w-node whose locations are known would help in providing more constraints for reducing the search space.

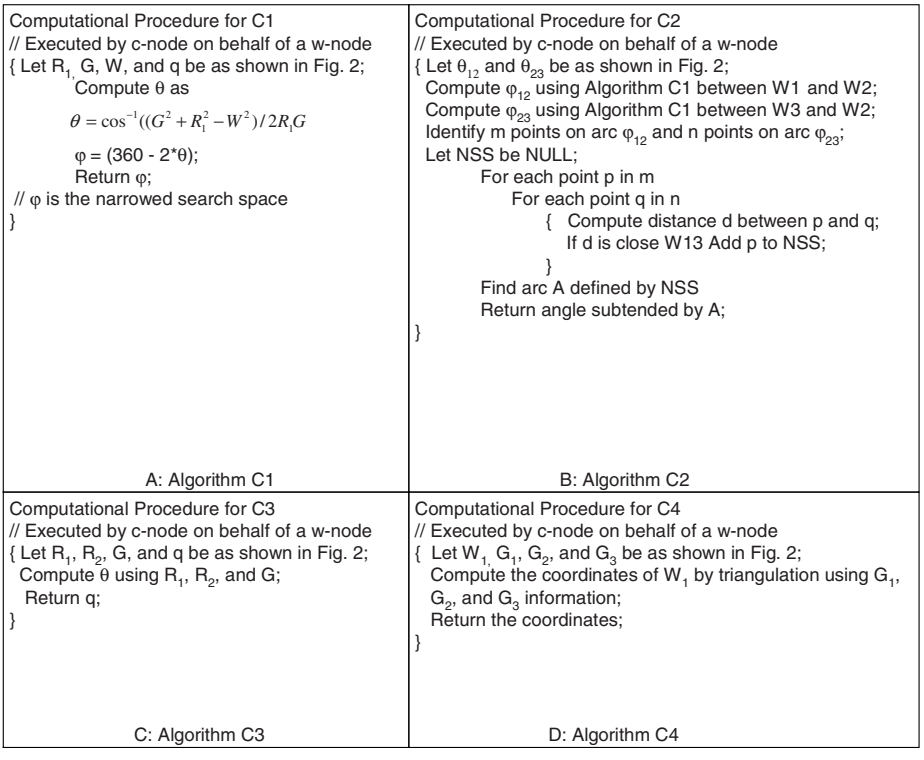

Fig. 4. Computational procedure for each of the identified four atomic configurations.

3 Self-Calibration of Position

Given a sensor network as shown in Figure 1, we describe an approach in which a sensor node uses the state of the sensor nodes in its neighborhood to self-calibrate its estimated position. The sensor node uses Configuration Identification Mechanism discussed in the previous section to improve its estimate over time. As the nodes move around, the state in the neighborhood changes and the node uses this change to its advantage to improve its position awareness. On initialization, the node performs the algorithm described in Figure 5 to obtain an initial estimate of its position. Note that the accuracy of its estimate depends on the neighborhood topology and due to the ad hoc nature of the mobile sensor network, this topology changes dynamically. After the successful initialization, the node constantly monitors for (a) any change in the neighborhood topology; and (b) any change in the position estimate of its neighbors. In either of the cases, it recalibrates itself using the algorithm described in Figure 5. Observe that in the algorithm described in Fig. 5, the node interacts with a c-node to help compute its position. Fig. 6 describes the algorithm executed by a c-node in order to compute the position of a w-node on receiving the inputs from the w-node. Note that c-node receives multiple instances of a configuration related data and

computes the possible angular position of w-node. Finally, when it has received and processed all the data sets, the resulting angular position is converted to possible positions and is returned to w-node.

```
Localization - Client
// Executed by w-node on need basis; Interacts with c-node assigned to w-node;
// A c-node is assigned during initialization statically and subsequently gets reassigned during
dynamic assignment
{   Analyze neighbors to determine the best possible configuration; // Use algorithm described
in Fig. 3 and obtain NeighborhoodData
        For each NeighborData in NeighborhoodData
        {  Send NeighborData to c-node;  Wait for Ack;
        }
        Receive PositionData;
        If Exact Position { Store Exact Position; return;          }
        Analyze Possible Positions based on past position and mobility information;
        Store the analysis result as Feasible Positions;
}
```

Fig. 5. Localization Algorithm – Client

Observe that in the algorithm described in Fig. 5, a w-node interacts with a particular c-node for computational purposes. There are three ways to assign a c-node to a w-node so that position computation can be distributed across the available c-nodes. In a static assignment, a w-node is configured to have an associated c-node. While it is easier to achieve load balancing by equally distributing the w-nodes to the available c-nodes during configuration, depending on the topology, this may have an excessive traffic across the network.

A dynamic assignment, on the other hand, tries to minimize the network overloading and hence, reducing the delay in communicating the results back to a w-node. However, in this case, additional computational effort is required to achieve load balancing. As a compromise, in quasi-dynamic assignment, instead of reassigning whenever there is a change in topology, the reassignment is made at regular, long intervals.

Observe that this periodicity could vary over time based on the nature of mobility of sensor nodes. A distributed dynamic assignment algorithm is described in Figures 7 and 8.

C-nodes implement the position computation algorithms described in Section 2 for the various atomic configurations. Further, as it receives multiple data related to a particular configuration from a w-node, the c-node computes position for each of these data related to the configuration to finally return a best approximate value if the configuration under consideration is not C4.

Our initial simulations involved defining networks over a region of 100 square units with 120 nodes each with one unit range. We have implemented the algorithms related to the four atomic configurations and the configuration identification procedure. The following figures, Figs. 9 and 10, describe some of the experimental results.

```
Localization - Server
// Executed by c-node in response to data received from a w-node
{       Receive NeighborData from w-node;  Analyze the data and check for configuration;
        If no more of neighbor data from w-node
        {  Convert ThetaSet to Possible Positions;   return Possible Positions to w-node }
        Switch {
           Case C4:   Use algorithm C4 and compute Position;   return Exact Position to w-node;
           Case C3:   Use algorithm C3 and obtain Theta;
           Case C2:   Use algorithm C2 and obtain Theta;
           Case C1:   Use algorithm C1 and obtain Theta;
        }
        ThetaSet = ThetaSet  n Theta;
Send ACK to w-node;
}
```

Fig. 6. Localization Algorithm - Server

```
Dynamic Assignment  - Client
// Executed by w-node
{   Discover all c-nodes in the network;
    Arrange c-nodes in non-decreasing order of hops into c-list;
    // c-node and w-node are separated by a number of hops;
    Start Timer;  // set to available time before which all w-nodes should have their c-nodes assigned;
                  // this time includes guard band time also
        While (not yet assigned a c-node)
        {       Select next c-node from c-list;   Send assign request to c-node;
                Wait for ACK or NACK
                If ACK is received {c-node gets assigned; return;          }
                If Timer expires { Reassign previously assigned c-node; return;  }
        }
        Reassign previously assigned c-node;
}
```

Fig. 7. Algorithm for dynamic assignment at client

Note that, in the Figs. 6A and 6B, the nodes connected by solid lines indicate that the position computation is by using C4 configuration, the nodes connected by long broken lines indicate that the position computation is by using C3 configuration, and the nodes connected by dotted lines indicate that the position computation is by using C2 and C1 configurations.

```
Dynamic Assignment  - Server
// Executed by c-node; w-list contains granted requests
// ow-list contains pending requests in the received order with the most recent at the head
// W is the limit number of w-nodes that can be assigned to c-node

{  Start Timer; // set to available time before which all w-nodes should have their c-nodes  assigned
   While (1)
        {  While (1)
           {     Check for request; If received Break;
                 If Timer Expires Break both loops; Sleep (0);
            }
           if |ow-list|  < W  {Add request.w-node at the head of ow-list; Continue; }
           For each w-n in ow-list
           if  w-n.Hops > request.w-node.Hops { Remove w-n from ow-list; Send NACK to w-n;
                        Add request.w-node at the head of ow-list; Break;
            }
        }
// Timer has expired
        If |ow-list| >= W {  Copy W requests from ow-list to w-list on first-come-first-served basis;
                Send ACK to each of the w-nodes in w-list;
                Send NACK to the remaining requests in ow-list;
                Check for requests from w-nodes and for each such request send NACK;
                // these are the requests sent by w-nodes just before timer expiry to c-node
                Return; }
// c-node can accommodate a few more w-node requests received within Guard Band time interval
        Start Timer // set to guard band time interval;
        While (1)
        {  Check for request;
           If received request
                        If |w-list| < W {  Add request.w-node to w-list; send ACK to request.w-node;}
                        Else Send NACK to the corresponding w-node;
                Else Sleep (0);
                If Timer Expires Break;
        }
}
```

Fig. 8. Algorithm for dynamic assignment at server

It is observed that, in Experiment 2, there were more number of nodes that were localized using C2 and C1 configurations as compared with those of Experiment 1.

4 Conclusions and Further Work

In this paper, we have proposed a topology based localization approach in the context of ad hoc mobile sensor networks. In a mobile sensor network in which there are a limited number of nodes with GPS-capability, there is a need for localization procedure that addresses the issues related to topologies in which exact computation

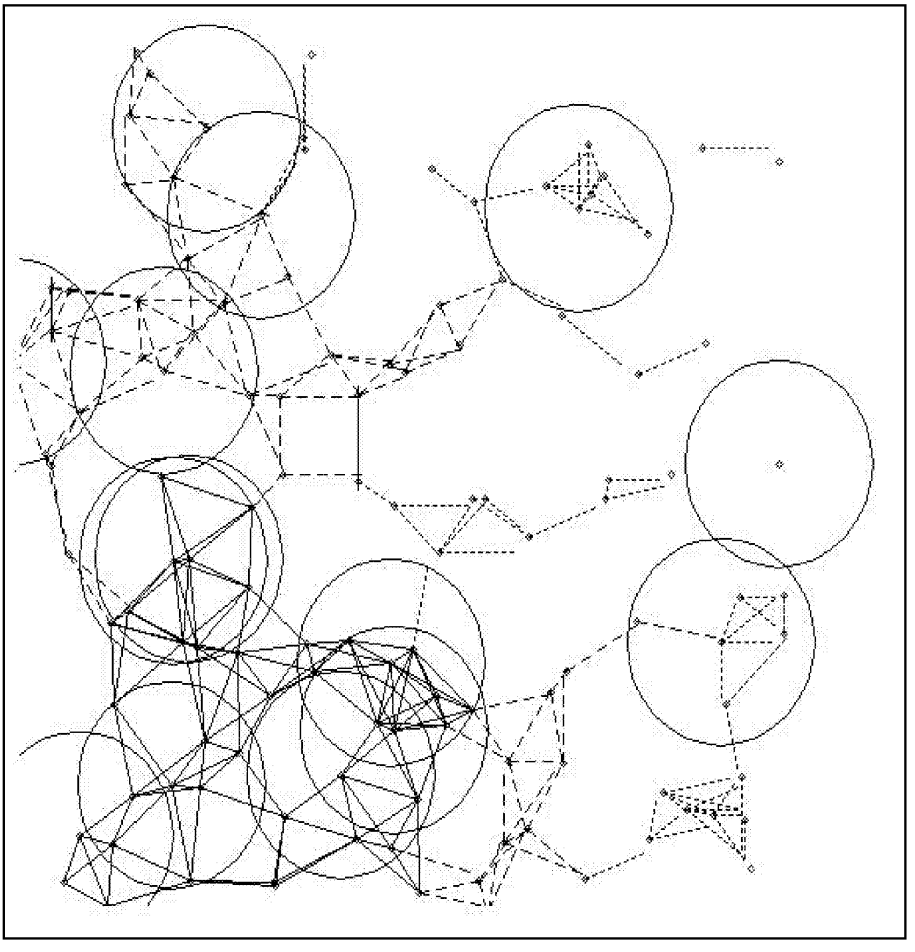

Fig. 9. Results of experiment 1

of position of many sensor nodes is not possible. Under these conditions, the approach suggested in this paper provides a best possible approximate estimate of the position of such sensor nodes. In order to achieve cost minimization while deploying a sensor network, we have suggested that the nodes in a sensor network could be of three types: g-nodes – those nodes that are GPS-enabled or location-aware; c-nodes – those nodes that have adequate computation power; and w-nodes – a wireless mobile sensor node.

We have suggested algorithms for atomic configurations that get executed on a c-node on behalf of a w-node during the self-computation of the position by the w-node. The idea behind identifying atomic configurations is to simplify the computational

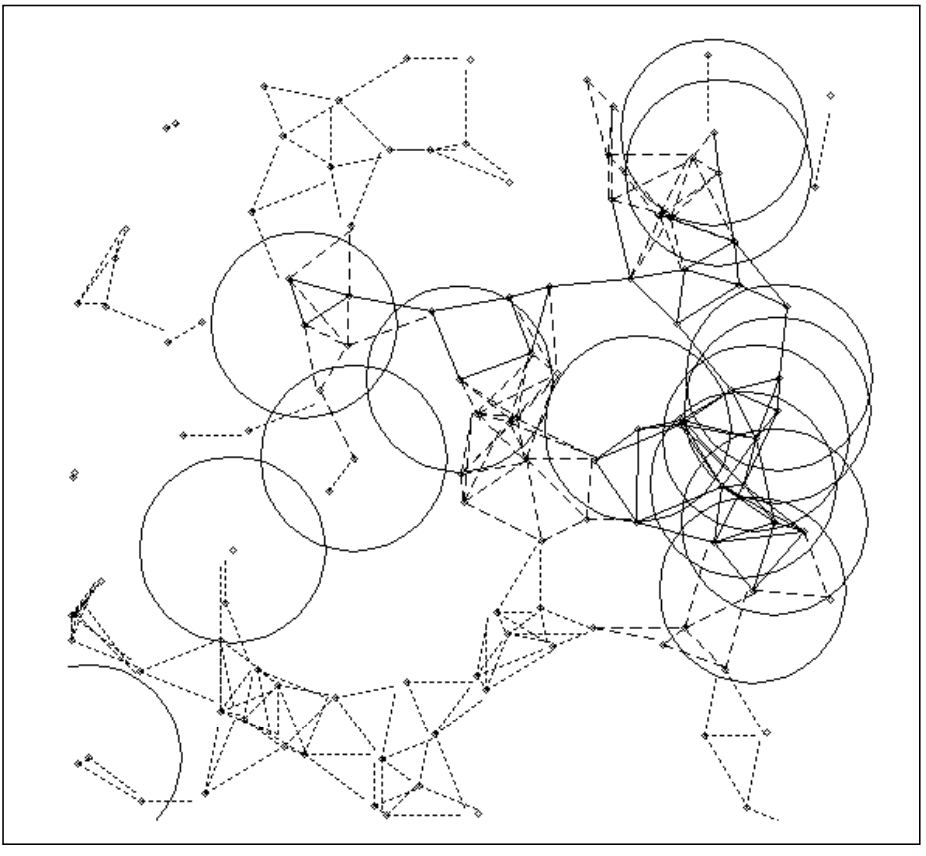

Fig. 10. Results of experiment 2

effort and handle the availability of only limited information. We have also suggested a dynamic assignment algorithm that dynamically, in a distributed fashion, assigns the best possible c-node for a w-node so that an attempt is made to achieve load balancing and minimize traffic across the network due to the interaction between w-nodes and c-nodes for position computational purposes. The results provided in this paper suggests that there are several w-nodes that require the usage of C1 and C2 configurations, along with other constraints, to determine their position. As part of the ongoing simulation work, we are working towards identifying different network topologies with a sufficiently large number of nodes and measuring the effectiveness of the suggested approach in terms of the extent of application of different configurations in localizing the nodes. Further, we are investigating the utility of distinct graph patterns such as (a) densely connected subnets; (b) sparsely connected subnets; and (c) partially localized chains. We also intend to explore the effects of density of w-nodes and sparsity of g-nodes on localization.

References

1. Awange, J.L., "Gröbner bases, multipolynomial resultants and the Gauss-Jacobi combinatorial algorithms – adjustment of nonlinear GPS/LPS observations," Ph. D. Thesis, University of Stuttgart, 2002.
2. Bulusu, N., J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," IEEE Personal CommunicationsMagazine, 7(5):28–34, October 2000.
3. Doherty, L., K. Pister, and L. El Ghaoui, "Convex position estimation in wireless sensor networks," Proceedings of IEEE INFOCOM 2001, volume 3, pages 1655–1663, Anchorage, Alaska, April 22–26 2001.
4. Hofmann-Wellenhof, B.,H. Lichtenegger, and J. Collins, Global Positioning System: Theory and Practice, Fourth Edition, Springer-Verlag, 1997.
5. McLurkin, J.D., "Algorithms for Distributed Sensor Networks," Masters Thesis for Electrical Engineering, University of California, Berkeley, December 1999.
6. Moses, R.L., D. Krishnamurthy, and R. Patterson, "A Self-Localization Method for Wireless Sensor Networks," Eurasip Journal on Applied Signal Processing, Special Issue on Sensor Networks, No. 4, Vol. 2003, March 2003.
7. Niculescu, D. and Badri Nath, "Ad hoc positioning system (APS)," Proceedings of GLOBECOM, San Antonio, November 2001.