An Architecture for Building Self-Configurable Systems

Lakshminarayanan Subramanian and Randy H.Katz
{lakme, randy}@cs.berkeley.edu
Department of Electrical Engineering and Computer Sciences, U.C. Berkeley

Abstract-

Developing wireless sensor networks can enable information gathering, information processing and reliable monitoring of a variety of environments for both civil and military applications. It is however necessary to agree upon a basic architecture for building sensor network applications. This paper presents a general classification of sensor network applications based on their network configurations and discusses some of their architectural requirements. We propose a generic architecture for a specific subclass of sensor applications which we define as *self-configurable systems* **where a large number of sensors coordinate amongst themselves to achieve a large sensing task. Throughout this paper we assume a certain subset of the sensors to be immobile. This paper lists the general architectural and infra-structural components necessary for building this class of sensor applications. Given the various architectural components, we present an algorithm that self-organizes the sensors into a network in a transparent manner. Some of the basic goals of our algorithm include minimizing power utilization, localizing operations and tolerating node and link failures.**

I. Introduction

Integrating low-power sensors will permit remote object monitoring and tracking of the physical environment. The deployment of such networks can vastly increase the accuracy of the information through collaboration between the various sensors. SmartDust [10], a technology developed at UC Berkeley will enable a rich collection of diverse applications ranging from sensor-rich "smart spaces" to self-identification and history tracking of virtually any physical object.

Networking the sensors to empower them with the ability to coordinate on a larger sensing task will revolutionize information gathering and processing in many situations. For example, large scale robust sensors could be deployed in inhospitable physical environments such as remote geographic regions and interesting information can be gathered from these networks. Sensors can also help in security environments where they can track the movement of certain objects in a coordinated fashion. Wireless sensors can also enhance remote access by connecting these networks to the Internet with the help of sink nodes.[7]

There are different kinds of sensor network applications in which sensors perform a wide range of activities. Among these, a certain set of applications require that sensor nodes collectively form an ad-hoc distributed processing network and provide information about the physical environment. Each sensor node operates autonomously without a central node of control. However, there are a lot of networking challenges associated with sensor nodes. These nodes must consume extremely low power and must communicate with its neighbors at bit-rates measured in kilobits per second and potentially need to operate at high volumetric densities. These requirements dictate the need for novel ad-hoc routing mechanisms and media access solutions which are power constrained.

This paper makes the following contributions:

- **Taxonomy of Sensor Applications:** We classify sensor applications based on the network configuration of the sensor nodes.
- **Self-configurable systems:** We identify the architecture and infra-structural components required for a specific class of sensor applications which we define as *self-configurable systems.*
- **Self-Organizing Algorithm:** We present an algorithm for self-organizing the sensor nodes in a transparent and distributed manner. We also discuss the strengths and the weaknesses of our algorithm.

Our taxonomy of sensor applications is based on the network configuration of the sensor nodes. Network configuration, in this scenario, refers to the physical placement of the various sensors and the connectivity of these nodes to nodes in the wired infrastructure. The network configuration determines the amount of routing intelligence that needs to be put into sensor nodes. Building an architecture for every sensor application requires an understanding of the interactions between the sensors and nodes connected to the wired infrastructure. The process of data discovery and data dissemination are two orthogonal components present in every sensor application. The mechanism through which data dissemination is achieved, depends completely on the network configuration and the routing mechanisms used.

We present an architecture for supporting a special class of sensor applications which we define as self-configurable systems. We make a few policy decisions and assumptions for building a generic architecture for self-configurable systems. They are:

- **Heterogeneous nodes:** A general architecture should be able to support heterogeneous types of sensors and should also provide a common framework for these different nodes to interact.
- **Data discovery vs Data dissemination:** In our architecture, we clearly divide the task of data discovery and data dissemination into two orthogonal components. We clearly identify nodes which can perform data discovery and distinguish them from nodes that can perform data dissemination.
- **Memory and Power Constraints:** We assume that every node has both memory and power constraints. We attempt to reduce the state stored in every node and also employ energy aware routing to decrease energy con-

sumption.

- **Application Specific Infrastructure Requirements:** The requirement of infra-structural components like a naming, routing or broadcasting system is completely dependent on the application. We provide a general self-organizing algorithm that can provide a wide variety of features and the applications turns on only those components based on its requirements.
- Mobility/Immobility of nodes: In our architecture, we assume the data discovery nodes to be mobile and the data dissemination nodes to be immobile.

In previous works [7], [3], all sensors have been treated to be alike and are assumed to have similar functionality. One of the important assumptions that we make in our architecture is the presence of heterogeneous types of sensors. We also assume that not all nodes are capable of performing data discovery and data dissemination. Even in applications like remote object tracking, it may be inappropriate to assume that specialized camera or acoustic sensors also perform the task of data dissemination. However one problems that arises when we make these two assumptions is how do we self-organize a large collection of heterogeneous sensors in the wide-area with a large collection of nodes performing data discovery alone. In order to solve this problem, We introduce a large number of router sensors whose only job is to perform data dissemination and interconnect the specialized sensors into a network.

Previous works [3], [7], [8], [9], use a powerful concept of data centric networking for sensor applications. Though this model is very interesting, it may not be applicable to many sensor applications. Certain applications like parking lot networks may require addressability for every sensor node and a method for routing messages to specific nodes. In the case where a large amount of data is discovered, the state that needs to be maintained by certain critical nodes (nodes connecting two components of the network) is very high in a data centric environment. If cut nodes of the network (nodes whose absence partitions the network) do not maintain the state of a particular data in a data-centric network, then that data does not propagate to all the nodes in the network. Our architecture can support both data-centric networking and non-data centric networking. However, we assume that none of the data dissemination nodes maintain a lot of state about the data discovered by sensors. This information is specifically extracted from specialized *sink nodes* (nodes with high capacities). Therefore, some of the data-centric models like directed diffusion [9] need to be slightly modified in order to be supported over our architecture.

The infra-structural components required by an application is completely dependent on the needs of the application. Every self-configurable system requires one or more of these four infra-structural components. They are

1. Naming/Addressing System
2. Routing
3. Broadcasting
4. Multicasting(for connecting sensors of a specific class alone)

We argue that it would be necessary to build a routing and/or broadcasting infrastructure for sensor applications which belong to the class of self-configurable systems. Routing infrastructure is required in applications where one may need to pass on some important information to sensors with specific functionality (eg., alarm actuators in security networks, sensors in a home-networking environment). In such environments, one needs to allocate a unique address for every sensor and route messages to particular nodes. Broadcast is necessary in scenarios where an important message (eg., intruder detected by a sensor) be broadcast to all nodes. This broadcast message can be sent as a *wakeup* message for sensors in a security network. The challenges involved in building such infrastructures must be met in the face of energy-constraints for computation and communication.

One of the important challenges in building large sensor networks is to self-organize the sensors into a network in a scalable fashion. In this paper, we describe our distributed algorithm for self-organizing a large number of sensors and build a routing infrastructure in the network. Using our algorithm, every sensor obtains a $O(\log n)$ bit unique node identifier in a distributed fashion and one can route information between any pair of sensors. Our algorithm also computes fault-tolerant broadcast structures in the network for broadcasting data in the network. The main contributions of our algorithm include:

- **Reduction of State and Localized Operations:** Any algorithm for sensor networks must reduce the amount of state maintained in every node. We maintain $O(\log n) + O(|N(v)|)$ state information at a node v, where n is the number of nodes in the networks and $|N(v)|$ is the number of neighboring nodes of v in the network.
- **Power Efficient and Reliable Paths:** Our algorithm keeps track of the power requirements at every node and computes paths which are reliable and power efficient. We propose two greedy power-constrained routing metrics for transmitting information.
- **Hierarchical Routing Architecture:** The sensor nodes self-organize themselves in a hierarchical structure and the size of the routing table at every node is reduced to $O(\log n)$ to reduce the state and computation and storage power requirements at every node.
- **Fault-Tolerant Broadcast Trees:** It is necessary to build broadcast trees in the network in a transparent manner so that important data could be broadcast to all the sensor nodes. The fault tolerance is achieved through our new technique *Local Markov Loops(LML)*.
- **Reduce frequency of Updates:** By defining discrete power levels in a sensor, we reduce the number of dynamic cost updates that need to be performed in the network.

Other than these contributions, we could also prove that our algorithm is loop-free and does not have the count-to infinity problem.

In Section 2, we classify sensor applications based on the network configuration and certain properties of the application. In Section 3, we detail the architectural and infrastructural requirements for building a specific class of sensor applications which we call as self-configurable systems. In Section 4, we describe the various phases in our algorithm and

present certain interesting properties of our algorithm. In Section 5 we present the related work and in Section 6, we conclude.

A. Terminology

In this subsection, we give a brief description of the terminology used in this paper.

- **Sensor Mote:** A sensor mote is an equivalent of a sensor node which performs data discovery.
- **Sink Node:** A sink node is a node with high processing capabilities and high capacity for data storage.
- **Specialized Sensor:** A sensor that performs a specific data discovery operation. (eg., camera sensors, acoustic sensors, temperature sensors)
- **Router Sensor:** A router sensor can collect data from other nodes and transmit them to neighboring nodes.
- **High-end system:** A high-end system is equivalent to a sink node.
- **2-connected graph:** Refers to a topology in which there are two edge disjoint paths from every node to every other node.
- **Broadcast graph:** Refers to a subset of edges in the network used for broadcasting data. These edges together form the broadcast graph.
- **Directed Acyclic Graph(DAG):** A graph which has directed edges and no cycles.

II. A Taxonomy of Sensor Network Applications

In this section, we give a brief description of the types of sensor network applications. We try to classify the applications into distinct classes based on the network configuration of the sensors in the system. The important factors used in the classification process are the size of the system, the number of sensors used, the maximum distance of the sensors to the wired infrastructure and the distribution of the sensor nodes. The size of the system and the number of sensors determine the effort needed to configure the system for that particular application. The distance of sensors to the wired infrastructure determines the amount of intelligence needed in a sensor for routing information to specific high processing nodes. The distribution of sensor nodes in sensor applications can be either *deterministic* or *non-deterministic*. In *deterministic* distributions, the administrator would have control over the placement of sensor nodes and the user can perform remedial operations in case of faults. In *non-deterministic* applications, fault-tolerance is increased by increasing the number of sensor nodes. Determinism normally decreases with an increase in the number of sensor nodes.

Based on a broad picture of different sensor applications, we classify sensor network applications into three types. They are:

1. Non-propagating systems
2. Deterministic routing systems
3. Self-configurable systems

In non-propagating systems, sensor nodes need not perform any intelligent functions for routing messages from them to high-end systems. In these systems, the sensor nodes are generally very close(one hop) to the wired infrastructure. The wired infrastructure is the main connecting component in these systems. These nodes discover data and report their measurements to nodes connected to the wired network which take the responsibility of routing information to the end system. Smart spaces installed in buildings or within restricted areas belong to this category. These systems are normally manually configurable and highly deterministic. The sensor nodes have no notion of routing in these systems.

In Deterministic routing systems, the wired and the wireless infrastructures play an important part in routing messages. In these applications, sensor nodes have to route through a few wireless hops in order to reach the wired infrastructure. However, the routes to the wired infrastructure are deterministic and can be configured manually. In home networking systems, the sensor nodes are in pre-specified positions and route information through pre-determined routes. The number of nodes in such a system may be restricted.

Systems in which sensor nodes need to self-organize themselves into a network belong to the class of self-configurable systems. Many self-configurable systems are non-deterministic, but when the number of nodes is large(more than 100) then even deterministic systems need to be self-configurable. Large parking-lot networks and security networks are examples of deterministic system which belong to the category of self-configurable systems. Remote object tracking is an example of a non-aggregating self-configurable system. In these systems, specific sensor nodes would have connectivity to the wired infrastructure for transferring information to high end-systems. The number of nodes in these systems can be anywhere between 100 and 1 million. Fault tolerance in these systems is achieved by re-organizing the network in the presence of node and link failures.

One can also classify sensor applications into *aggregating* and *non-aggregating* systems. In *aggregating* systems, the data obtained from the different source nodes can be aggregated and transmitted along the network. Intermediary nodes in the network would have the capability to fuse the information obtained from different sources. In *non-aggregating* systems, the information gathered by every source node is independent and have to be transmitted separately in order to get a complete picture of the system. Weather forecasting and monitoring systems are examples of aggregating systems and parking-lot networks is an example of a non-aggregating system.

Non-propagating and deterministic routing systems which are also aggregating systems have most of their aggregation functionality performed in the wired infrastructure or at the gateway connecting the wireless network to the wired infrastructure. Therefore, such systems do not require specialized aggregating functionality to be embedded into the sensor nodes in the network. In self-configurable systems, sensor nodes within the network should also perform the functionality of aggregation of data.

Self-configurable systems have a lot of open research issues and is by far the most challenging system to build among these systems. In this paper, we mainly concentrate on self-configurable systems and detail some of the architectural and

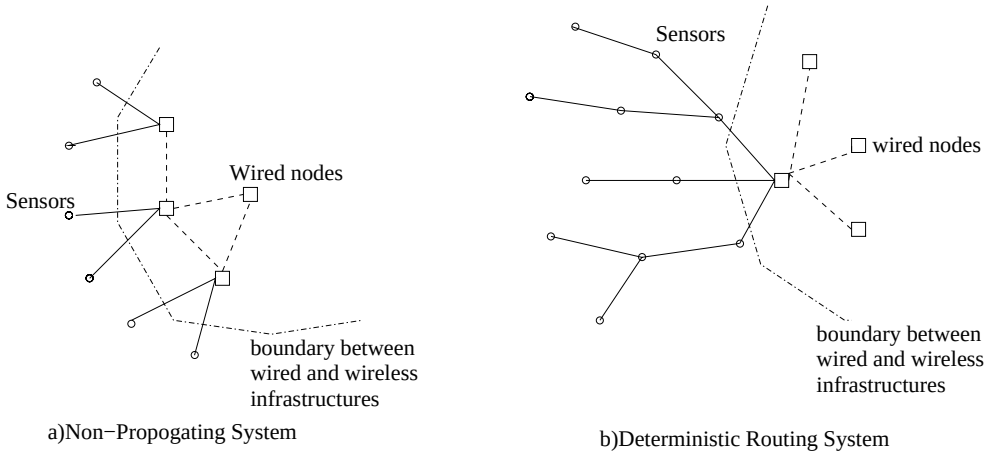

Fig. 1. Non-propagating and Deterministic routing systems

infra-structural components required for these systems. We present an algorithm that self-organizes the nodes into a network and builds an addressing, routing and broadcasting infrastructure over the network.

III. Architecture

In this section, we give a detailed description of the various components and functionalities provided by the architecture. We also give scenarios of applications requiring a particular functionality. We also discuss the different components of the infrastructure to support specialized networking functionalities. In this architecture, we assume that the sensors are in fixed locations and do not move. We believe that the presence of these components would be necessary and sufficient for supporting many heterogeneous sensor applications.

Figure 2 gives an illustration of the various types of nodes in the architecture and how they are self-organized into a network.

A. Architectural Components

We consider two main types of sensors in our design. They are *specialized* and *router* sensors. Our architecture also considers the presence of special nodes called *sink nodes* [7] which have the general characteristics of a computer system. In other words, these nodes have huge storage capacity, high processing power, connectivity to the wide-area(Internet) and do not have strict power constraints.

A.1 Specialized sensors

The heterogeneity of sensors in a network is specified by the presence of specialized sensors in the network. There are special sensors for monitoring climatic parameters like temperature, pressure, humidity etc., tracking sensors for tracking or detecting motion, vision sensors which can photograph images and a lot of other types of sensors. It is important to know how these sensors would co-exist in a common environment. In our architecture, each specialized sensors identifies itself with a class and can communicate with other sensors either of its own class or with some other class. The class of a sensor denotes the functionality of that sensor. The specialized sensors in a self-configurable system can be mobile.

A.2 Router sensors

It would be necessary to understand why router sensors may be essential in sensor nets. Previous works [7], [3], [10] have considered all sensor nodes in a network to have the same functionality. As discussed earlier, there are quite a few sensor applications like parking-lot networks which require data to be routed to specific sensors and actuators. The router sensors in the architecture can be used as routers in the network or can be used as data dissemination nodes These nodes self-organize themselves to form the backbone of the sensor network. In our architecture we assume that router sensors are strictly immobile. If these sensors are mobile, then the backbone of the network needs to be continuously reorganized. This reorganization could be very expensive. Data dissemination is useful in scenarios of broadcasting data in the network.

The use of *router* sensors in the architecture has the following advantages:

1. The process of data dissemination and routing data is separated from the process of data discovery. In other words, specialized sensors perform the task of data discovery and the router sensors are responsible for routing and disseminating data.
2. It is a well known fact that the power consumed by using N short hops is approximately N times smaller than the power consumed in 1 long hop. The presence of router sensors makes most of the hops in the network to be short hops rather than long ones and thereby helps in reducing power consumption.
3. Certain specialized sensors may be expensive to build and our architecture reduces the density of these sensors in the network. The backbone of the network is very built only using cheap router sensors and can be highly cost-effective in certain systems.

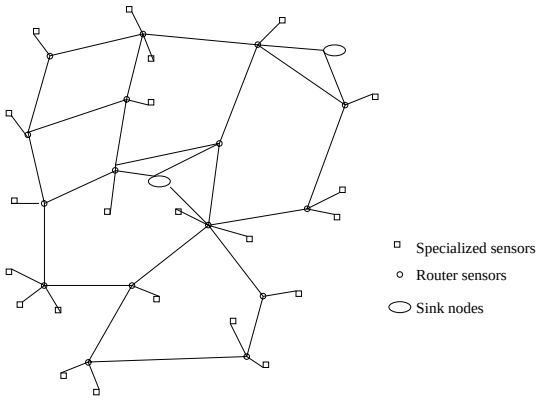

Fig. 2. A self-organizing system with router sensors, specialized nodes and sink nodes

4. The presence of a large number of router sensors in the backbone can increase the lifetime and the fault-tolerance of the network.

The ratio of the number of router sensors to number of specialized sensors (R/S ratio) is completely dependent on the application. In certain applications like remote object tracking, it would be advisable to build chips which have both the capability of specialized sensors and router sensors. In this application, it would be advisable to have an R/S ratio of 1.

A.3 Aggregator nodes

Self-configurable aggregating systems require the presence of aggregating functionality in certain nodes within the network. This functionality can be either introduced in the router sensors or one can create specialized nodes which act as aggregators. These nodes collect the data from different sensors and aggregate them before transmitting them further. In weather monitoring applications, it is appropriate for placing the aggregation functionality on all router sensors.

A.4 Sink nodes

Sink nodes is a terminology that we borrowed from [7]. Each sensor node may generate a lot of interesting information and it would be necessary to store this information at a place. Sensor nodes do not have enough power to process or store large amounts of data. In other scenarios like remote tracking, it would be necessary to connect these sensor networks to the Internet so that the information generated by the sensor nodes can be transmitted to other remote nodes through the Internet. This architecture supports the presence of *sink* nodes in the network which have high processing power, high storage capacity and can connect to the wide area networks for transmission of useful data. The *sink* nodes also have the capability of sending specific messages to certain nodes to activate specific actuators and can broadcast important messages in the sensor net.

B. Infrastructure Components

In this section, we investigate the basic backbone networking functionalities required for supporting sophisticated sensor applications. We believe that the components in our infrastructure would be necessary and sufficient for many sensor-based applications. The functionalities that we support in our architecture are the following:

1. Unique address for all nodes
2. Routing information between two nodes
3. Fault-tolerant broadcasting infrastructure
4. Broadcasting information within a certain radius
5. Multicasting information to specialized nodes
6. Self-reorganization in the face of node failures and network partitions.

We would provide with scenarios where these functionalities are necessary. In the next section, we would give a detailed description of our algorithm that provides all these functionalities in a scalable fashion.

We detail certain applications which require many of the listed infrastructure components. Consider the following scenarios:

- **Security Sensors:** Assume a simple scenario where sensors keep track of safety vaults in a bank and form a security network. When some sensors detect intruders, they must trigger an alarm and must alert other specialized sensors like vision, acoustic and path detection sensors about this event. For triggering the alarm, the security sensor must send a control signal to an actuator, must activate the vision sensors to take pictures of the intruder, inform the object tracking sensors to track the motion of the intruder and each specialized sensor must pass critical information to the sink node. For performing this whole functionality, each specialized sensor must have a unique address so that other sensors can pass critical information to that sensor. We need a routing architecture to send messages between sensor nodes and to the sink nodes. We also need a multicasting infrastructure to coordinate the actions of specialized sensors of a particular

type. For example, the vision sensors coordinate amongst themselves to take pictures of the intruder. Note that the multicast infrastructure would be restricted to a certain locality and may not span the entire network. The object tracking sensors must coordinate amongst themselves to track the path of the intruder. We also need a broadcasting infrastructure to alert all nodes within a certain radius to get ready for performing some important actions.

- **Parking-Lot Networks:** Assume that every parking spot in a certain area contains a sensor which keeps track of the status of the spot on a continuous basis. A person traveling around the region can get instantaneous access the available parking spots and can also make advanced reservations to certain spots. In this example, the sensor application requires both an addressing and a routing infrastructure for routing messages to sensors and actuators controlling one particular spot. Broadcasts are not highly relevant for these networks but can be used to send specific messages to a group of nodes (eg., No parking in street #37 between 4 and 6pm).

B.1 Addressing Infrastructure

We have discussed many scenarios where addressing nodes is very essential. However there are applications as described in [3], [2] which do not require addressing of sensor nodes. Any sensor application which is a self-configurable system that does not require an addressing mechanism for individual nodes is an aggregating system. *Traffic monitoring* along highways with the help of sensors requires self-configuration of the nodes but does not require a unique address for every node.

Given that an addressing scheme needs to be built, an IP based addressing to this problem would not be a good solution. For one, IP addresses are global unique addresses but sensor networks require local unique addresses. Allocating and keeping track of IP addresses would be a very cumbersome task and one would waste a lot of IP addresses given the size of these subnets. In our architecture, sensor nodes within a certain area interact with themselves in a distributed manner and come up with an addressing scheme in which each node obtains a unique local address.

Developing a MAC layer for sensor nets would be an alternate solution. This will ensure that every sensor node obtains a unique MAC address. Is it worth that effort is still an open question.

B.2 Routing Infrastructure

Another important component of our architecture is the presence of a routing architecture which connects all the sensors in a transparent manner. The routing infrastructure is established as an interconnection network between the router nodes. Every specialized node must be adjacent to an adjacent router. This node transmits all its messages to its adjacent router node with a message header. The message header specifies whether the message is to be transmitted to a particular node or it should be broadcast or multicast in the network. Based on this header information, the router nodes transmit the information among the router nodes transparently. Once it reaches a router node which is reachable to the destination, it is sent to the destination node. The routing backbone is formed by self-organizing the router nodes into a network. We describe our algorithm for self-organizing router nodes in the network so that we minimize the power consumed at each node and also reduce the size of the routing table at every router node. In our architecture, every specialized node is addressed with the help of a router node which acts as the proxy for the special node. Many specialized nodes may be connected to the same router node.

B.3 Broadcast and Multicasting Infrastructure

In sensor applications, as mentioned in [3], it is important to concentrate on data-centric models rather than traditional IP centric applications. In data-centric applications, it would be important to broadcast or multicast some critical information to all or special nodes within a certain radius. We have illustrated some examples which would require this application in the beginning of this section. In the next section, we would describe our algorithm for developing broadcasting infrastructure. The algorithm supports the construction of a fault-tolerant broadcast tree which changes continuously by establishing Local Markov Loops(LML) in the network. We also develop a directed-acyclic structure(DAG) to support fault tolerance in paths. Multicast framework for exchanging information between specialized nodes of a certain type can be supported over this broadcast infrastructure. Since our algorithm divides the network into various levels, it is also possible to achieve localized broadcast where information can be broadcast or multicast within a certain region.

IV. Our Self-organizing Algorithm

In this section, we describe our algorithm which helps in self-organizing a set of sensor nodes randomly scattered in an area. The router sensors self-configure themselves into a network using this algorithm and the specialized sensors only keep track of the nearest router sensors which is alive. The algorithm consists of four phases. The algorithm performs the following operations in the order they are mentioned:

1. **Discovery phase:** Each node independently discovers its set of neighbors in the network and fixes its maximum radius of data transmission.
2. **Organizational phase:** During this phase the network is organized and the following operations are performed:
   (a) Node aggregate themselves into groups and groups are aggregated to form larger groups. In this way, a hierarchy of groups is formed in the network. The algorithm ensures that the hierarchy is height balanced.
   (b) Each node is allocated an address based on its position in the hierarchy.
   (c) A routing table of $O(\log n)$ is computed for every node in the network.
   (d) A broadcast tree and a broadcast graph spanning all nodes in the graph is constructed. The broadcast graph is then converted into a directed acyclic graph based on the source node in the network.

3. **Maintenance phase:** In the maintenance phase the following operations are performed.
   (a) In active monitoring, every node keeps track of its stored energy and constantly sends *I am alive* message to its neighbors once in 30 sec. In passive monitoring, a sensor nodes sends an activate message to its neighbors only on demand.
   (b) Every node constantly updates its routing table about the next hop in the least power consuming path and the shortest delay path to the groups as dictated by the algorithm.
   (c) Nodes also inform their neighbors of their routing tables and their energy levels to their neighboring nodes.
   (d) Fault tolerant broadcast trees and broadcast graphs are maintained using Local Markov Loops(LML).
4. **Self-Reorganization phase:** In this phase, a node may detect group partitions or node failures and change its routing table based on the new topology. If all neighbors of a node fail, then the node repeats the discovery phase. If a group partition occurs due to link or node failures, the sub groups reorganize and join with new groups. Group re-organization ensures that the hierarchy is still balanced.

We would now give a complete description of each phase and in particular show how the addressing, routing and broadcast of information is performed in this self-organized network.

A. Discovery phase

In the discovery phase, each node discovers its set of neighbors in the network. There are a lot of factors to be considered while finding the set of neighbors for a particular node. A node should not have many neighbors because the receiver antenna of the node has to be time multiplexed between the various nodes and hence the time slice obtained for each node is extremely less and thereby the delay of the system increases and the throughput decreases. Another factor to be considered is the maximum radius of transmission for every node. The energy expended on transmission is proportional to the square of the distance between the sender and receiver. Hence each node would be expending a lot of energy while transmitting data. In effect, each node bounds its set of neighbors and the maximum transmission radius. On the other hand, a node x would like to have a minimum number of neighbors $n(x)$ for performance considerations. For specialized sensors $n(x) = 1$ but for router sensors $n(x)$ must be higher. This will depend on the application and the density of sensors in the network.

Steps of the Discovery phase:

1. Every node x picks a small radius r and broadcasts a *Hello* message around a radius r and also indicates whether it is a special node or a router node.
2. Every node within a radius r reply back with a *I am here* message with their coordinates(determined by GPS).
3. If the number of nodes that responded is less than minimum threshold $n(x)$, then x broadcasts *Hello* message over a radius kr for $k > 1$. This process continues until the number of nodes N that respond satisfies $n(x) \leq N \leq N(x)$ where $n(x)$ and $N(x)$ denote the minimum and the maximum number of neighbors that can be admitted by x.

Note that a router sensor can be connected to any type of sensor but a specialized sensor is only connected with router sensors. Using the steps described above each node discovers its set of neighbors and the maximum distance of transmission.

B. Organizational phase

The organizational phase consists of various stages. They are:

1. Formation of a hierarchy with the help of group formation.
2. Performing group reorganization if necessary.
3. Generation of addresses for nodes.
4. Generation of routing table at every node
5. Generation of broadcast trees and graphs within a group and merging of broadcast graphs and trees whenever groups are aggregated.

We will describe each stage in detail. In the beginning it would be necessary to understand how a hierarchy is found in a distributed manner and how group reorganization helps in generating a balanced hierarchy.

B.1 Group Formation

The basic step in the algorithm is the group generation phase. After each node has found its set of neighbors, each router node attempts to form a small basic group with its neighbors. Each group is restricted to a size of 8 members and every node should belong to exactly one basic group. Each node in a group is allocated a $3-$bit address and every node maintains the distance and the next hop for reaching every other node in the group.

B.2 Merging of Groups

Assume 2 groups G_1 and G_2 with m and n-bit addresses respectively. By n-bit addresses, we mean that every node in G_2 has an $n-$ bit address. For a basic group, we have $n = 3$. Assume $m \geq n$ without loss of generality. Our algorithm has a basic height difference parameter namely $\Delta > 0$ which dictates the maximum height difference tolerable for merging trees. Therefore if $m-n \leq Delta$, then G_1 and G_2 are merged into one group G and all the nodes in G_1 add the bit 0 in front of their address and all nodes in G_2 add the bit 1 in front of their address. If $m - n > \Delta$ then we consider the address of some node that connects G_1 to a node in G_2. Let this node be x and with an address $(x_1, x_2, \ldots x_m)$. Consider the sub-group H_i formed by the set of all nodes in G_1 with first i bits equal to $(x_1, x_2, \ldots x_i)$ for $i \leq m - n - 1$. By the group formation property it can be seen that H_i is connected. Find out whether H_{m-n-1} has enough free address space for accommodating G_2 within its sub-group. By enough space, we require to know whether H_{m-n-1} has a sub-branch in its hierarchy where G_2 could be added in the hierarchy of H_{m-n-1} without affecting the height of the $H_{m-n-1} \cup G_2$. If it is not possible to perform this merging try the same with H_i for the maximum value of i from $m - n - 2$ to 1. If it is not possible to merge with any

value of i, then merge G_2 with G_1 in the original fashion and mark the new graph as height imbalanced.

B.3 Group Reorganization

A hierarchy of a group needs to be reorganized when it is height imbalanced at multiple levels. If the hierarchy of the group is unbalanced, then the group is broken into sub-groups of smaller size which are regrouped in a similar way to produce a hierarchy which is balanced. This reorganization does not affect the state of the rest of the network. Some routing tales of nearby nodes will have to change the addresses of their neighbors.

B.4 Formation of Hierarchy

The most important part of the algorithm is the hierarchy formation. We assume that every node would have finished the basic group formation algorithm. If a node is not able to join any group, it forms a one node group with address 000. A node of a group G is a boundary node if it is connected to a node of some other group. Set the value of Δ to be the height of a basic group which is 3. The following steps are carried out to obtain the hierarchy:

1. Each group G receives advertisements from its adjacent groups through its boundary nodes. These messages are broadcast throughout the group. Each advertisement consists of the size of the adjacent group(number of address bits).
2. Each node conclude on an adjacent group G' which is closest in size to G and which has the maximum number of boundary nodes.
3. The node G sends the *join* message back to G'. If G' also decides to join G, then the two groups merge else G selects the next best group H.
4. This process continues until all groups are merged into one. At any point if a group is heavily imbalanced(hierarchy is imbalanced) then the group is reorganized with an increased value of $\Delta_{new} = \Delta_{old} + 1$.

Theorem 1: The height of the hierarchy of the network will be $O(\log n)$ where n is the number of nodes in the graph.

The above theorem follows from the height balanced property of the hierarchical tree. To give a general picture, if the value of $\Delta = 3$, then every node in a sensor network of 10000 nodes will have a 16 bit address.

B.5 Routing Table Formation

Using the above hierarchical formation algorithm, a hierarchical tree can be formed in the network and every node will have an address. Let every node have an $m-$ bit group address. In this algorithm, we can construct a routing table at each node. Let $(x_1, \ldots x_m)$ be the address of a router sensor x. Then this sensor would maintain the least cost and next hop in the shortest path to the following destinations $x'_1, (x_1, x'_2), \ldots (x_1, \ldots x_{m-1}, x'_m)$. For example let the value of m be 4. Let a router node have the address 0011. This node maintains the next hop to groups $1, 01, 000, 0010$. By this way, it is possible to maintain a routing table of $O(m)$ at every node and perform hierarchical routing.

However a cache can be employed at every node to cache the next hop for particular destination groups. But this caching scheme must pin the next hop along all nodes in the path. Even if one node drops the information, the information is lost.

B.6 Broadcast Graphs and Trees

Broadcast graphs refer to the use of generalized graphs for broadcasting rather than using spanning trees. The intuition is that a node will be accessible through multiple paths from the source and thereby fault-tolerance is added to the system. In sensor networks, broadcast graphs would consume more power than broadcast trees. In our case, broadcast graphs are transformed into directed acyclic graphs directed from the source. Therefore there are no loops in the graph. To reduce the power consumption in broadcast graphs, we denote certain links as primary links and other links as secondary links. All broadcast messages are directly transmitted through broadcast links rather than the $3-$ way handshake in [7]. Along secondary links the protocols follow the $3-$ way handshake mechanism in [7].

The broadcast trees and graphs are formed in the following way:

1. Whenever a basic group is formed, a broadcast tree and graph are constructed for the basic group. Some nodes of the broadcast graph are labelled as primary and they form the broadcast tree.
2. Whenever two groups G and H merge, we select two low cost edges which connect G and H. Call them e_1, e_2. Let the broadcast graphs and trees of G, H be $B(G), B(H), T(G)$ and $T(H)$ respectively. Let $cost(e_1) < cost(e_2)$ and let the merged group be denoted by P. Then $B(P) = B(G) \cup B(H) \cup \{e_1, e_2\}$ and $T(P) = T(G) \cup T(H) \cup \{e_1\}$.

Theorem 2: The power consumed for broadcasting messages using this approach is $(n-1)E + nE'/2$ where E is the mean power consumed for sending a long message along one hop, E' is the mean power consumed for sending a request/ACK short message along one hop and n is the number of nodes in the network.

This is better than the power consumed by the SPIN protocol in [7]. That protocol consumes $(n-1)E + 2eE'$ where e is the number of edges in the graph. In any broadcast scenario, the energy $(n-1)E$ consumed is inevitable and is a lower bound for the amount of energy that needs to be utilized. In typical sensor networks, every node would have around $10-15$ neighbors. The value of e for such a network is $6n$ and hence the total energy used per broadcast is $(n-1)E + 12nE'$. Our algorithm, saves by a factor of 24 on an average on the extra energy utilized per broadcast. Though the value of E' is very small, the value $2eE'$ might be very large for large sensor networks. For two sensors separated by a distance of 10 meters, its takes $150nJ$ per bit of information to be transmitted and $170nJ$ is the power required for receiving a bit of information. A typical request-ACK message requires around 8 bytes of information to be exchanged between the two nodes. The value of E' in such a case is $20480nJ$. Given a network of 1000 nodes with an average connectivity of 12, our algorithm

consumes an extra energy of $10mJ$ while the algorithm in [7] consumes $240mJ$ for every broadcast.

C. Maintenance phase

There are two types of maintenance that one can perform in a self-organizing system. They are: active and passive monitoring. In the maintenance phase, it would be necessary to maintain consistent routing tables at each node and also updates the costs of the nodes in each node. It would also be necessary to maintain fault tolerant broadcast trees.

C.1 Active vs Passive Monitoring

In active monitoring, every node keeps verifying the status of its neighbors periodically. Every node sends an *I am alive* message to all its neighbors once in 30 seconds to which it has not sent any message over the last 30 seconds. If a node does not receive a response from its neighbor for six consecutive time intervals, the node assumes that the link between the two nodes has failed and reorganizes its structure to tolerate the link failure. In passive monitoring, a node checks whether a particular neighbor is alive only on demand. Node A sends a particular message *Are you alive?* to node B for which node B responds with an ACK. Passive monitoring is used as a mechanism for saving the energy of a particular node.

C.2 Routing Metrics

Delay is not a very important constraint in sensor networks. It would be more appropriate to save on power utilization than the delay experienced by messages. The goal of the routing metric is to keep the network alive for the maximum amount of time. We suggest two different greedy metrics which can help in achieving this goal. Given a network with each node having a certain energy, it is a very hard problem to theoretically compute the capacity of the network even for a particular source and a sink. In the first metric, we always route along the path that has the minimum energy consumption per bit of information transmitted. In the second metric, we always transmit along the path that has the maximum capacity measured in terms of bits that can be transmitted. Given nodes A, B with energies $E(A), E(B)$ and that A consumes energy E' for transmitting one bit to B and B consumes E'' for receiving a bit from A. The capacity of the link between A, B is given by $min(E(A)/E', E(B)/E'')$.

C.3 Maintenance of Routing Tables

Each node constantly informs its neighboring nodes about its cost metric and this information is used by its neighbors to update their routing tables. The *count to infinity* problem can be avoided by not using the next hop entry for updating the routing table entry for a particular destination group. Both the cost metrics can be made *loop-free.*

C.4 Maintenance of Broadcast infrastructure

To maintain resilient broadcast trees in the face of node or link failures, it is necessary to detect node failures in advance by monitoring the power requirements of a node. The principle behind making our broadcast tree fault-tolerant is by changing the broadcast tree to a new tree where the node that is going to fail a leaf node. Therefore this node will not need to broadcast any information to any other node in the tree. For a node u that is going to fail consider all edges (u, v) which is present in the broadcast tree. Construct a local Markov loop(LML) by selecting a random edge (w, x) such that the edge (u, v) is part of a local loop formed by adding the edge (w, x) to the tree. Remove (u, v) from the tree. We get a new tree with degree of u reduced by 1. Perform this operation until u becomes a leaf node.

D. Reorganization phase

Re-organization occurs when either a node fails or when a network partition occurs. We enlist the type of failures and suggest our solutions:

1. **Node failure:** Every node constantly sends *I am alive* message to its neighbors. If a node does not receive any message from one neighbor over 6 period cycles, the neighbor is assumed to be dead. Every neighbor of the node updates all the entries in their routing table where the next hop is the failed node. If the node that is bound to fail is not a leaf node of the broadcast tree, then the node is made a leaf node by local loops.
2. **Link failure:** A link failure occurs when a node becomes unreachable to another node. In this case, the routing table is changed accordingly at both the nodes. If the edge is a primary edge in the broadcast tree, the algorithm converts the corresponding secondary edge that connects the two groups into a primary edge and performs a local loop to find an alternate edge.
3. **Group Partition:** If all the links connecting two parts of a group fail or if some crucial nodes fail, the group gets partitioned into two or more disconnected pieces. These disconnected pieces would reorganize themselves into new groups and merge with other neighboring groups. In such a case, the address of all the nodes in the group change.
4. **Node Rediscovery:** Assume a scenario in which all neighboring nodes of a particular node have failed. In such a case the node starts a rediscovery phase with an initial radius equal to the previous maximum radius of connectivity.

Reorganization of groups and node discovery events are very rare. The only common occurrences are node and link failures.

E. Analysis of the Algorithm

In this subsection, we list the strengths and the weaknesses of our algorithm. Some of the strengths of the algorithm include:

1. The hierarchy formed by the algorithm is strictly balanced. The maximum difference between the left subtree and the right subtree at any level is strictly less than or equal to Δ.
2. The routing state maintained by any router sensor is $O(\log n)$.

3. The algorithm incrementally computes a broadcast graph which is $2-$ connected.
4. The property of Local Markov Loops(LML) performs a random walk on spanning trees of a graph. This provides tolerance to node failures and link failures.
5. The broadcast graph can be oriented as a directed acyclic graph from any node in a unique manner. The uniqueness property is guaranteed by the presence of a hierarchy.
6. The property that every specialized sensor attaches to some router sensor allows these sensors to be mobile.

Some of the weaknesses of the algorithm include:

1. The algorithm has an initial organization phase and does not have a concept of on-demand organization. Initial organization is good for applications that require addressability and/or routing. It is very applicable in scenarios where the maintenance phase is not very costly. In extremely dynamic systems, it is better to have no implicit organization or on-demand organization. [7] is an example of a work that has no implicit organization. [9] belongs to the category of on-demand organization.
2. Forming a hierarchy in cases where there are a lot of cut nodes in the network would not be a good idea. This would increase the probability of applying the reorganization phase.
3. The algorithm does not discuss the protocol required for transmitting data from one node to another node. In particular, it does not address the issue of when a node should transmit an information to another node.

It is highly unclear about what the best energy aware routing metric is. This topic is out of the scope of this paper. Is delay or energy consumption the right metric for sensor applications? The answer to this question is very application specific.

V. Related Work

[7] is one of the first works towards building adaptive protocols for information dissemination. In their family of adaptive protocols namely *SPIN*, each node advertises to its set of neighbors whenever it has some interesting information. These protocols optimize on the power consumption in the nodes. However, they do not build a routing infrastructure in the network and only target broadcast applications. They also assume that every message in the network must eventually be broadcasted. In the case when a critical piece of information is necessary for a specific subset of nodes in the network(eg., sensors in one's home, traffic sensors in a highway), none of the intermediary nodes may be interested in the information but the information must be transmitted to the final subset of nodes. In such a scenario, [7] does not guarantee delivery.

[3] is a paper that comments on the requirements of scalable coordination in sensor networks. They hint at building a data-centric model in which applications focus on data generated by sensors. They also stress on lowering power consumption and performing localized operations.

[2] argues for an address free architecture for dynamic sensor networks but we have showed some applications which require addressing of nodes. [4], [8], [19] describe some energy efficient adaptations for sensor networks and mobile applications. [9] presents a new communication paradigm for sensor networks. Though their model is very interesting, it may not be applicable to a wide range of sensor applications.

The concept of hierarchy formation is not new. We borrowed some ideas on hierarchy formation from Large Packet Radio Networks [17] and PNNI hierarchy [1]. There has been a lot of work in the area of ad-hoc routing protocols. These include [5], [13], [14], [15], [16], [12]. Some of the problems of sensor networks can be solved using ad-hoc routing protocols. But, in principle the power constrained problems and the data-centric model make sensor networks completely different. *Source* and *Distributed* routing ad-hoc protocols require the maintenance of a lot of state information in the nodes and incur huge communication overhead for dynamic link-costs. Therefore, these algorithms are not well suited for power constrained networks and perform poorly in data-centric models.

VI. Conclusions and Future Work

This paper describes a taxonomy of sensor applications. We believe that such a classification has not been reported in previous works. The paper also describes a generic architecture for building a special class of sensor applications called self-configurable systems. We describe the reasons behind the requirement of every architectural and infra-structural component. We feel that these components would be necessary and sufficient to build self-configurable systems.

In this paper, we also have described a self-organizing algorithm that develops an addressing, routing and broadcasting infrastructure in the backbone of the network. The hierarchy formation ensures that the height of the tree is $\log$ the number of nodes in the network. The routing table maintained at every node is also of reduced size. The algorithm mainly targets power constraints and attempts to minimize the power consumed at various stages of the algorithm. The paths and the tree structures are made fault tolerant by constantly making failing nodes as leaf nodes.

In our architecture, we assume that specialized sensors are mobile within the region of router sensors. The architecture for applications where sensors may move with no area restrictions will be very different from this one. We are working towards evaluating the practicality and the goodness of the algorithm through simulations and also by implementing over some of the sensor motes designed by Berkeley MEMS department.

Acknowledgements

Ashwin A.Seshia cleared a lot of our doubts regarding the sensor aspects of the architecture. Some of the communications aspects of the problem were discussed in detail with Kiran. We also thank Prof.K.S.J.Pister and Sharad Agarwal for their valuable comments.

References

[1] ATM Forum. http://www.atmforum.com/

[2] Jeremy Elson and Deborah Estrin. An address free architecture for dynamic sensor networks. submitted for publication. http://www.isi.edu/ estrin/papers/

[3] Deborah Estrin et.al. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of MOBICOM'99*, Seattle, pp 263-270.

[4] Jason Flinn and M.Satyanarayanan. Energy aware adaptation for mobile applications. *ACM Symposium on Operating System Principles (SOSP 1999)*, December 1999.

[5] Piyush Gupta and P.R.Kumar. A system and traffic dependent adaptive routing algorithm for ad-hoc networks. *In Proc. IEEE* 36^{th} *Conference on Decision and Control*, pp 2375-2380, Sandiego 1997.

[6] E.Gafni and D.D.Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions in Communications*, Jan 1981.

[7] Wendi R.Heinzelman, Joanna Kulik and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of MOBICOM'99*, Seattle, pp 174-185.

[8] Wendi R.Heinzelman, Anantha Chandrakasan and Hari Balakrishnan. Energy efficient communication protocol for wireless microsensor networks. *Hawaii International Conference on System Sciences*, january 4-7, 2000.

[9] C.Intanagonwiwat et.al. Directed diffusion: A scalable and robust communication paradigm for sensor networks. to appear in *ACM Mobicom 2000*.

[10] J.M.Kahn, Randy H.Katz and K.S.J.Pister. Next century challenges: Mobile networking for "Smart Dust". In *Proceedings of Mobicom'99*, Seattle, pp270-278.

[11] Satish Kumar et.al. Scalable Object tracking through unattended techniques(SCOUT). submitted for publication. http://www.isi.edu/ estrin/papers/

[12] Charles E.Perkins and Elizabeth Royer. Ad-hoc on demand distance vector routing. Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-05.txt

[13] Charles E.Perkins and P.Bhagwat. Highly dynamic destination sequenced distance vector routing for mobile computers. In *Proceedings of SIGCOMM'94*, October 1994.

[14] Vincent D.Park and M.Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In*proceedings of INFOCOM'97*, 1997.

[15] Sinha P., Sivakumar R. and Bhargavan,V. CEDAR: A core extraction distributed ad-hoc routing algorithm. In *Proceedings of INFOCOM'99*.

[16] R.Sivakumar, B.Das and Bhargavan V., An improved spine based infrastructure for routing in ad-hoc networks. In *Proceedings of IEEE Symposium of Computers and Communications*, 1998.

[17] Martha Steenstrup. *Routing in Communication Networks*. Prentice Hall, 1995.

[18] William A.Winoto, Elliot Schwartz et.al. The design and implementation of an intentional naming system. 17^{th} *ACM Symposium on Operating System Principles(SOSP 1999)*. 34(5); 186-201, Dec 1999.

[19] Ya Xu et.al. Adaptive Energy Conserving Routing for multihop ad hoc networks. submitted for publication. http://www.isi.edu/ estrin/papers/