1-1-2010

Desire: A Socially Inspired Algorithm for Autonomous Emergent Routing in Ad-hoc Senso Networks

Pranab K. Banerjee
Utah State University

Banerjee, Pranab K. 2010. "DESIRE: A Socially Inspired Algorithm for Autonomous Emergent Routing in Ad Hoc Sensor Networks." Paper presented at Seventh International Conference on Information Technology: New Generations, Las Vegas, NV, April 12-14. doi:10.1109/ITNG.2010.29

DESIRE: A Socially Inspired Algorithm for Autonomous Emergent Routing in Ad-hoc Sensor Networks

Pranab K. Banerjee
Space Dynamics Laboratory
1695 North Research Parkway, North Logan, Utah 84341, USA

Abstract

This paper presents a socially inspired dynamic emergent routing algorithm for ad-hoc sensor networks called DESIRE (Dynamic Emergent Socially Inspired Routing Enabler). The network is composed of nodes with no goe-locating capabilities, and uniformly spatially distributed over a rectangular area. It consists of two types of nodes - (i) large number of sensor nodes with relatively limited storage, power, and radio range, and (ii) the relatively sparse transmitter nodes with higher storage, power, and radio communication range. A sensor node is responsible for sensing the environment in its immediate vicinity. A transmitter node collects data from a set of sensor nodes and transmits this information to a collector station. The network is assumed to be autonomous with no centralized control. This paper proposes an algorthm that dynamically constructs the network communication topology based on available resources through emergent properties resulting from local inter-node communication. The mechanism is analogous to emergence of social structures in human communities through primarily short-range local communication among community members.

Key Words: adaptive, routing, topology, self organization

1. Introduction

Sensor networks refer to a collection of low cost sensing nodes where each node has small form factor, limited computational capacity, limited power, and short range wireless communication capability. These nodes are usually deployed in a spatially dense way for *in situ* sesnsing of various environmental, biological, nuclear and other parameters of interest. Normally there is no pre-planned deployment topology for such a sensor network. Instead, the nodes are randomly placed over the spatial domain of interest, and there is no centralized control over the network operations. Hence, it is paramount that these nodes have the capability to self-organize in order to autonomously develop dynamic communication topologies for information transfer to the data collection agents. Thus, these collection of sensing nodes form an *ad-hoc* network. The concept of *ad-hoc* networks has been around for quite a long time. One of the earliest research in this area was the DARPA sponsored project on packet radio network [1]. However, these networks were mostly of theoretical interest until the recent advances in the areas of miniature electro-mechanical systems, highly compact computing hardware, embedded computing software, wireless communication, and compact batteries which have made sensor netoworks practical, cost effective and operationally viable. Berkeley motes [2] are probably one of the most prominent examples of such deployable sensors.

The motivation behind this research was the deployment of a sensor network for battlefield environmental situational awareness for the safety of the war fighters. It is conceivable that the enemy might have engaged in environmental (chemical, bilogical, radioactive etc.) contamination in strategic locations to make those sites dangeours for human war fighters. In this scenario, a cluster of sensor nodes could be sprinkled over these areas from an airborne platform, and the sensed information collected from an aircraft during a follow-on flyover. The sensor nentwork consists of two kinds of nodes - (i) large number of chemical/biological/radioactive agents sensing nodes with relatively limited storage, power, and radio range, and (ii) a sparse set of transmitter nodes with higher storage, power, and radio communication range. These transmitter nodes collect data from a set of sensing nodes and then transmit the collected information to the data collecting aircraft during flyover within range.

This paper describes an algorithm for emergent dynamic route formation in such an *ad-hoc* network based on short range local information exchange. The algorithm is based on analogies found in human society for self-organized structure formation based on localized communication among community members.

2. Algorithm

The algorithm described in this paper is inspired by observations about how some long range information pathways are formed in human society through local short range communication among neighbors and acquaintances. A simulation testbed has been developed using the Python scripting language for implementation and study of the algorithm. This testbed has a built-in discrete clock that advances time. A clock cycle is defined as the time required to send or receive one *message unit*. The sensors and the transmitters are assumed to be uniformly distributed over a rectangular region. This is a realistic assumption since [3] has shown that sensors dropped from aircrafts follow such a distribution.

2.1. A Social Scenario

Consider a new comer **X** to a foreign city **C** who wants to send a letter in an envelope to another part of the country through a courier service. However, as someone unfamiliar with the city and the country, **X** has no idea as to the availability of courier services. It so happens that the city **C** has no such service available but the courier services from the neighboring larger city **D** periodically advertise their services in the local newspaper **N** whose circulation is mainly limited to that city. However there are some people in **C** who know about these services through friends who live in **D** and have access to the newspaper **N**. The person **X** eventually comes to know a set of neighbors **P**, **Q**, and **R** who know about the services in the next city and are willing to accept the envelope from **X** to forward it to one of the courier services either directly or through their friends. Because of limited transportation access in an unfamiliar city, **X** decides to accept the offer from his neighbors. **P**, **Q**, and **R** each give **X** an estimate of how long it would take them to forward the envelope to one of the couriers in the city **D**. The person **X** saves the information about all the possibilities of forwarding the letter to a courier, and hands it to the neighbor who can deliver it the fastest. It eventually reaches a courier for cross-country delivery. Consequently, **X** receives a delivery confirmation message originating from the courier service that received the packet. If **X** does not receive a confirmation within a specified time since the envelope was handed to a neighbor, **X** prepares a duplicate envelope, and hands it to another neighbor for forwarding to a courier that he knows about. The algorithm implemented here is based on such a scenario.

2.2. Analogy With A Sensor Network

In the sensor network under consideration, the transmitting nodes are analogous to the courier services, and the sensing nodes are analogous to members of a community who need to send information to a destination (in this case, the data collecting aircraft) using the courier services. At the very beginning after being deployed, these nodes broadcast their availability through "Hello" messages, and the sensing nodes come to know about the availability of those transmitting nodes within communication range. Such a broadcasting by a transmitter node is analogous to the advertising in the newspaper **N** in the human social scenario. The spatial footprint of the broadcast signal range of a transmitter T_i is analogous to the area of the city **D**, and the set of sensing nodes which directly receive the broadcast from a transmitter T_i are analogous to the members of population of the city **D** who come to know about the courier service through the advertisements in **N**. The transmitter nodes keep broadcasting these "Hello"s at monotonically increasing intervals. This helps those sensor nodes which could not receive the earlier messages either because they were busy communicating with other nodes at the time or because they were deployed at a later time, become cognizant of these transmitters. A sensor node with a transmitter within its direct communication range is called a *DC* (Direct Connect) node. Other nodes are called *ND* (Non Direct) nodes.

2.3. Neighbor Discovery

In this algorithm, each sensing node also broadcasts a "Hello" message at monotonically increasing time intervals, and the other nodes that receive this message send an acknowledgment signal back to the transmitting node. Note that the acknowledgments are not broadcasts but directed responses to specific transmitting nodes. Thus, this algorithm is a hybrid of broadcasts (flooding) as well as peer-to-peer communication modes. This "Hello"/acknowledgment mechanism is meant for discovering neighbors within range. When a node n_i receives a Hello from another node n_j, it adds n_j to its sensor neighbors list. Each sensor node also maintains a separate list of transmitter neighbors, which are the transmitters that it received a Hello broadcast from. A transmitter node also maintains a list of neighboring sensors that it received acknowledgments from in response to a Hello message.

It is assumed that a sensor can receive data from one source at a time, i.e., it has a single receiver channel. However, it can transmit while it is receiving. So, a receiver within range of multiple transmitters broadcasting Hello messages simultaneously can receive such a message from only one transmitter during one simulation clock cycle. This is one of the reasons why it is important for a transmitter to keep broadcasting these "Hello"s so that they can be discovered by more and more sensor nodes over time. However, the interval between these broadcasts get longer with time. This is because a transmitter gets discovered by most of its neighbors during the first few Hello broadcasts. When a sensor broadcasts a Hello, its next Hello broadcast time is set to a base interval plus a random offset, thus reducing the chances of multiple transmitters broadcasting at the same time causing message collision.

A transmitter is assumed to have the ability to receive data on more than one channel at a time. The number of such receiver channels T_R for a transmitter is a configuration parameter in this algorithm. However, it is assumed to have a single transmitter. This is to make the transmitters satisfy low power consumption requirement. Transmissions are usually more power hungry than receptions. Having multiple receiver channels makes it more efficient for a transmitter node to receive the Hello acknowledgments from multiple sensor nodes in one simulation clock cycle. This enhances the process of neighborhood discovery.

2.4. Dynamic Self-Organized Routing

The goal of the route development mechanism is to enable the sensor nodes to deliver their sensing results to one of the transmitter nodes for eventual delivery to the data collecting aircraft for spatio-temporal situational awareness. In the context of this algorithm, a route is a directed graph from a sensor node to a transmitter. A route is represented as an ordered list of nodes such as $[n_1, n_2, \cdots, n_k, t_j]$ where the sensor node n_1 uses the intermediate nodes $n_2 ... n_k$ to relay its message to the transmitter node t_j. Note that the final node in a route is always a transmitter. Routes develop in a self-organized manner and propagate from *DC* (Direct Connect) nodes to *ND* (Non Direct) sensor nodes. The route formation is not initiated until the network has had the chance to form a set of DC sensor nodes, each of these DC nodes have had the chance to discover a preset minimum number of neighboring sensor nodes through the Hello/acknowledgment message transfer, as described in section 2.3, and the DC nodes have broadcast a preset minimum number of Hello messages (this ensures that a node has made sufficient attempt to develop a neighborhood).

Once the route generation phase starts, a DC sensor with enough neighboring sensors and enough Hello broadcasts under its belt broadcasts its routes for its sensor neighbors. All the DC as well as ND neighbors who are not engaged in any communication during that clock cycle receive these routes, add their own node ID to the head of the ordered lists, and saves them as their route lists. As is evident, a sensor node can have multiple routes in its route list as it can receive routes from more than one neighbor. During broadcast, a node only tranasmits the shortest route. If there are more than one route in the list with the same number of hops, it picks one at random from this set. Depending on the size of the message containing a route, it may take more than one clock cycle to transmit and receive. During this period, the transmitting channel on the broadcasting sensor and the receiver channel on each of the receiving nodes are marked busy. Subsequently, these neighbors broadcast their route lists for their neighbors, and the process continues. When a node broadcasts a route, its next route broadcast time is set to a fixed interval plus a random offset. This reduces message collisions, and increases the chance of a sensor to receive route broadcasts from multiple neighbors, since a sensor node has only one receiver channel. When a sensor is not broadcasting a "Hello" or a route, it attempts to deliver its sensing results (referred to as the *payload*) to a transmitter node for delivery to the data collector. If a node is a ND sensor, its payload has to be relayed via other nodes. In this case, a path information is attached to the payload. When a node relays this payload, it attaches its ID to the payload's path. When the payload reaches the final transmitter, the path has the trace of its journey. This enables a delivery confirmation message issued by the transmitter to be relayed back to the originator of the payload. If a node does not receive a delivery confirmation within a specified time limit, it assumes that the payload delivery was not successful (possibly because one of the nodes in the route failed in the mean time) and sends the payload again along a different route, if available. If no other route is available, it waits for at least a predefined resend interval, and then sends it again, hoping that one of the nodes in the path figured out a new route to bypass any failed node in the mean time.

2.5. Payload Delivery

A node n_P that has a payload L to deliver (either's its own or one's being relayed on behalf of neighbors) picks the shortest route from its stored list of routes. If there are multiple routes with the same number of hops, it picks one at random from this subset. It then attempts to a setup a dedicated, connection-oriented communication link with the next forwarding node n_F and requests information about its available power p_F and storage capacity s_F. If n_F is not busy, it replies with the requested status. n_P then determines if s_F is sufficient to store the payload L and if p_F is sufficient to receive and then transmit L to the next hop along the route. It is important for this connection to be a dedicated one because otherwise p_F and s_F could be obsolete by the time n_P forwards the payload to n_F.

It is worth noting that the payload delivery is based entirely on direct node-to-node communications and does not use broadcast or flooding. This results in a power efficient data transfer mechanism.

2.6. Fault Tolerance

In sensor networks, reliability is achieved through aggregation and dynamic self-organization. Because of the compact footprint and limited resources, it is likely that some of the nodes would fail to operate before the expected life span of the entire network. It can be due to drained battery, failure of the radio electronics, storage failure etc. Thus it is important for a sensor network to autonomously reconfigure itself to minimize the impact of failed nodes. This algorithm achieves this through periodic verification by each node of its neighbors' health. This is achieved through a directed *status request* from a specific neighbor. If an OK reply is not received, this could mean the neighbor was busy with other communication task and failed to receive the status request, or the node is dead. So the status request is repeated as necessary a preset number of times at randomized intervals. If no OK is received at the end, this node is assumed to be dead, and the node requesting the status deletes any route from its route list that includes the dead node. It also repeatedly broadcasts a predefined number of times at randomized intervals the fact that a node is dead. This helps its neighbors to pick up this information and update their route list as well.

3. Implementation

A simulation testbed has been developed to study this algorithm. It has been implemented using the Python scripting language on an Intel CPU based PC running the Ubuntu 8.04 [4] version of the Linux operating system. This is an agent based simulation where each node acts as an agent and the routes emerge through local inter-agent interactions. The behaviors of the agents are dictated by a set of attributes. Some of the key attributes are total battery power at startup, total storage capacity, initial Hello transmission interval, rate of increment of this interval with time, power depletion per message unit transmission/reception, idle state power depletion rate, initial route broadcast interval, rate of increment of this interval with time, communication range for sensor nodes, minimum number of "Hello"s to be transmitted before engaging in route formation, and the minimum number of neighbors to find before engaging in route formation.

4. Results

Results from running the algorithm with 300 sensor nodes and 20 transmitter nodes sprinkled over an area of dimensions 100x100 square units are presented here. The following parameter values have been chosen empirically: initial charge for sensors = 1000 units, initial charge for transmitters = 10,000 units, storage capacity of sensor nodes = 100 units, storage capacity of transmitter nodes = 1,000 units, initial Hello transmission interval = 2 clock cycles (all times are in clock cycles and will be implicit from here on), Hello transmission interval increment size = 1, maximum Hello transmission interval = 6, minimum route broadcast interval = 2, route broadcast interval increment size = 1, maximum route broadcast interval = 6, power depletion per message unit transmission = 2 units, power depletion per message unit reception = 1 unit, idle power depletion rate = 0.01 units/clock cycle, communication range for sensors = 15 units, minimum number of "Hello"s to be transmitted before engaging in route formation = 15, and minimum number of neighbors to find before engaging in route formation = 4.

Figure 1 shows the shortest routes formed with 300 sensor nodes and 20 transmitter nodes. The red lines represent the routes, the black circles represent the transmitters and the blue "x" markers are the sensor nodes.

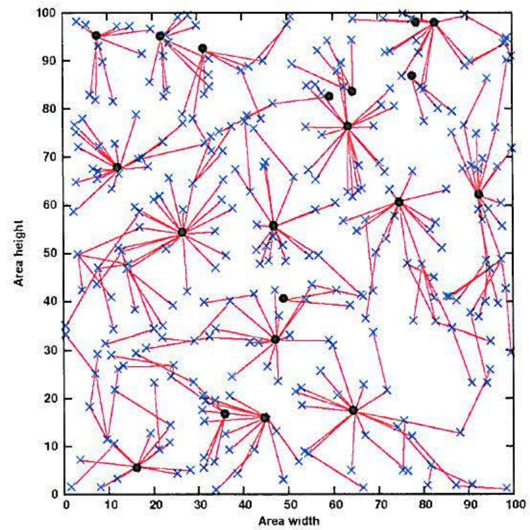

Figure 1. Routes in a network of 300 sensors and 20 transmitters

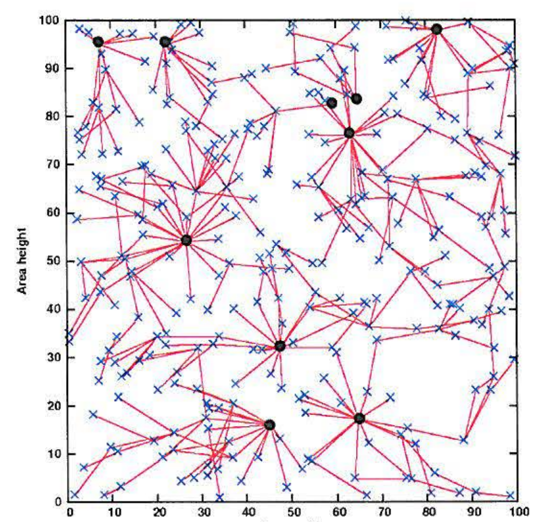

Figure 2. Reorganized routes in the network when ten of the transmitters failed

To illustrate the fault tolerance of the algorithm, Figure 2 shows how the network self-organized when ten of the original 20 transmitter nodes were disabled (to represent failure). Note that all the sensor nodes still have routing ability to at least one transmitter node.

Figure 3 shows how the neighborhoods evolve in this algorithm. It shows the number of isolated nodes without any knowledge of neighbors with simulation time. Initially (during time steps 0 annd 1), all the nodes are isolated. Time 0 corresponds to the deployment of the sensors and time 1 corresponds to the start of algorithm. However, neighborhoods form rather rapidly, and every node is part of a neighborhood by the 16th time step, indicated by zero isolated nodes in this figure.

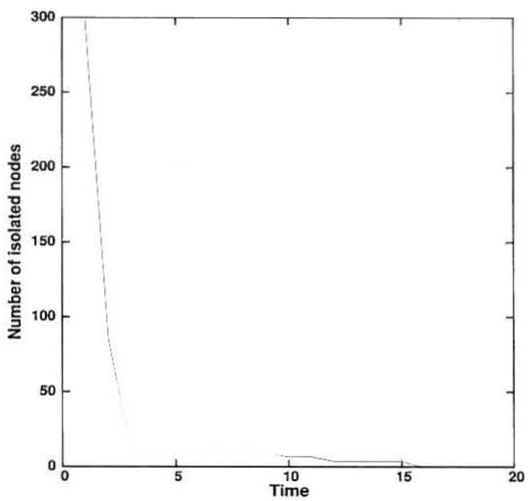

Figure 3. Evolution of neighborhoods with time

Figure 4 shows the effect of the number of transmitters in a network on payload delivery rate from the sensors to the transmitters. It appears that addition of transmitters are beneficial up to a point. But further addition beyond that level does not improve performance by any significant amount. In figure 4, the red curve corresponds to 15 transmitters, and it shows 17 nodes yet to deliver their payload at the end of 200 simulation time steps. The green curve corresponds to 20 transmitters and in this case all the nodes delivered their payloads by 179 time steps. The dashed blue curve corresponds to 40 transmitters and now all the nodes are able to deliver payloads by 133 time steps. So going from 15 transmitters to 40 transmitters show significant improvement in delivery rate. However, increasing the number of transmitters from 40 to 50, as indicated by the purple curve shows almost no improvement at all.

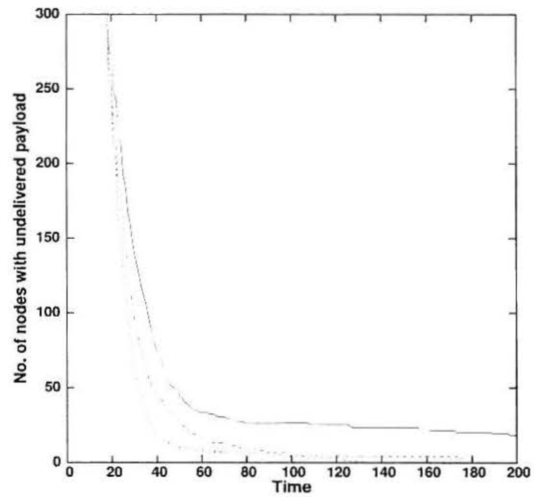

Figure 4. Effect of number of transmitter nodes on payload delivery rate

5. Future Research

This is an ongoing research and has scope for further development. One of the areas that needs more study is the determination of optimal parameters. These are chosen empirically at present. However, careful sensitivity analyses are planned in the future. Also, the algorithm needs to be addressed through rigorous statistical and mathematical framework for deeper understanding of the interaction of the different controling parameters. Another area of interest is the implementation of the algorithm on a GPU for significant performance enhancement and scalability.

References

[1] R. E. Kahn, S.A. Gronemeyer, J. Burchfiel, and R.C. Kunzelman, "Adances in Packet Radio Technology", *Proceedings of the IEEE*, 1978, pp. 1468-1496.

[2] R. Bose, "Sensor Networks - Motes, Smart Spaces, and Beyond", *IEEE Pervasive Computing*, July-Sept. 2009, vol. 8, no. 3, pp. 84-90.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey", *Computer Networks*, 2002, vol. 38, no. 4, pp. 393-422.

[4] Ubuntu, *http://www.ubuntu.com* (accessed on November 06, 2009)