The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

TENTACLES: Self-Configuring Robotic Radio Networks in Unknown Environments

Harris Chi Ho Chiu, Bo Ryu, Hua Zhu, Pedro Szekely, Rajiv Maheswaran, Craig Rogers, Aram Galstyan, Behnam Salemi, Mike Rubenstein, and Wei-Min Shen

Abstract— **This paper presents a bio-inspired, distributed control algorithm called *TENTACLES* for a group of radio robots to move, self-configure and maintain communication between some critical entities (such as humans, command centers, or other systems) in an unknown environment. The basic idea is to direct robots' explorative movements to grow "tentacles" from entities and establish links when tentacles meet. This approach can self-heal failures of robots and improve communication coverage and quality over time. Experiments in simulations and real robots have shown positive results.**

I. Introduction

Establishment and maintenance of communication between entities in an unknown and dynamic environment is a very challenging problem. This is due to the unknown locations and distances between entities, obstacles that may block or deflect signals, unexpected changes in the environment, and movement of or damage to the entities. One possible solution is to deploy a group of intelligent robots to explore the environment and position themselves to provide relays. Such robots must self-configure into an effective network, self-optimize the performance of the network, self-heal changes and damages, and adapt to movements of the critical entities.

Researches have studied the robotic radio network both in open areas and in indoor environments. For open areas, the nodes of the radio network are able to form different patterns using attraction-repulsion fields [1], self-heal broken links [2] and self-optimize connectivity [3]. The assumptions usually include a nearly perfect inverse square radio model. However, the multipath property of indoor radio signal environments causes signal strength to be noisy and unpredictable within nearby locations. Howard et al. [4] incrementally build a sensor network covering indoor building, which relies on another radio network node as a landmark. Stump et al. [5] optimizes the connectivity of the robotic radio network with a known global map. Both approaches require the robots to be initially connected and the relative position between robots to be known. Compared to these and other previous work, the main contribution of this paper is a distributed coordination algorithm for self-organization and self-healing of robotic networks to establish radio links between critical entities despite the unpredictability and noise of radio signals and unknown locations of entities and relay nodes. The ideal resulting network should support high-quality communication between the critical entities yet use as few robots as possible.

One example of possible applications is urban search and rescue where rescuers have to enter unknown buildings and communications to the outside world is vital. In a complex environment, walls, doors, obstacles and other materials make radio propagation hard to predict. At the same time, relay nodes may be damaged and unpredictable changes of environment, such as changes in signal propagation due to fire damage, may occur. This often results in a situation where it is essentially impossible to determine the accurate relative positions of other nodes, and how the signal strengths may change when transmitters and receivers move. In these scenarios, pre-deployment of relay nodes can be infeasible. Radio nodes must to self-organize the communication links between entities to be effective.

In this paper, we address a subset of the problem of self-organization and self-healing of robotic radio networks by assuming that the critical entities (non-robots) are *stationary* even though their locations and the map of the environment are unknown to the robots.

The paper is organized as follows. Section II describes the problem in detail. Section III presents the *TENTACLES* algorithm with illustrations of how a tentacle is built and removed, how robots explore the environment, and how they locally optimize network performance. Section IV and V discusses the implementation of the algorithm and the experimental results in simulation and on real robots. Section VI concludes the paper with future research directions.

II. The Problem Description

Figure 1 illustrates an overview of the problem to be solved. The nodes S_i and G_i are the critical entities to be connected (they are assumed stationary in this paper) and the nodes N_i are the mobile robots with relay radios. We assume the distance between the G_i and S_i nodes are too large for them to communicate directly. There are unknown obstacles in the environment (not shown) that may prevent robots from going freely to wherever they want. The goal of the robots is to move to positions where they can relay the communication between the G_i and S_i nodes with sufficient

H. Chiu, P. Szekely, R. Maheswaran, C. Rogers, A. Galstyan, M. Rubenstein, B. Salemi and W.M. Shen are with Information Sciences Institute, The University of Southern California, Marina Del Rey, CA 90292, USA
Email: chichiu@usc.edu, shen.usc.edu
Hua Zhu and Bo Ryu are with ArgonST. 6696 Mesa Ridge Road, Suite A, San Diego, CA 92121.
Email: Hua.Zhu@argonst.com, Bo.Ryu@argonst.com

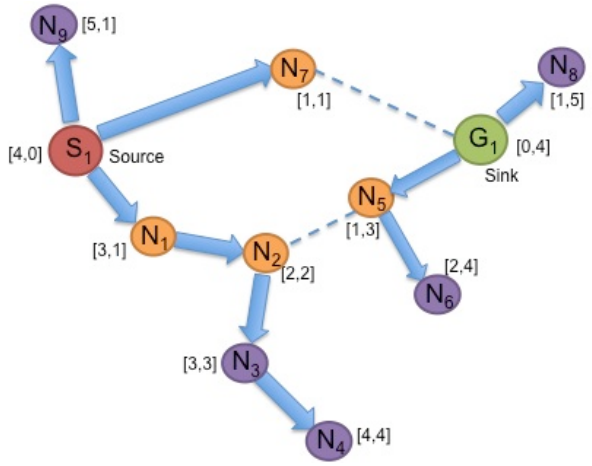

Fig. 1. A tentacle building and connection map with distance vector. Node P => node Q means P is a parent of Q in a tentacle with *Good* signal strength. Dotted lines represent radio links between nodes in *Normal* range of signal strength or better. Distance vectors *[Sink, Source]* are used for differentiation between open nodes and closed nodes.

bandwidth and throughput. They accomplish that by growing "tentacles" (the thick links between nodes) from the G_i and S_i nodes and hope the tentacles will meet (shown by the dashed lines). The robots may notice that they are *non-critical* in certain tentacles such as N_3, N_4, N_6, N_8 and N_9 and such nodes in the tentacles should be removed to establish more promising tentacles somewhere else. The *TENTACLES* algorithm to be described below will run on each robot and together the robots will self-configure, self-improve, and self-maintain the network even in case of node damages.

To ease our following descriptions of radio signal strength, the range of the strength will be in 4 regions: *Strong*, *Good*, *Normal*, and *Weak* separated by three signal strength threshold values α, β and γ ($\alpha > \beta > \gamma$). A connection is considered as disconnected if its strength signal is in the *Weak* range.

III. Tentacle Building Algorithm

Without global map information and accurate relative physical positions of other radio nodes, connecting the entities and even the robotic radio nodes themselves does not come easily. At the same time, robotic radio nodes must balance the tradeoff between moving/exploring unknown areas for possible connections and staying/maintaining the existing links (i.e., not moving). Intuitively, a radio node should stop moving once it realizes it has already been relaying network traffic between the critical entities. If relaying is not possible without multiple agents, a *"tentacle"* should be grown. Technically, a tentacle consists of a series of stationary robotic radio nodes stretching out from the entities. These entities can communicate when the tentacles meet one another. If a tentacle is deemed to be useless (based on its lifetime and whether it carries communication traffic or not), that tentacle will be rebuilt and nodes become

TABLE I
Data Structure in Tentacle Building Algorithm

Local Data Structure:

```
Node myNode
    int nodeID; int tentacle[]; int distanceVector[]; bool isLeaf;

NeighborNodeEntry nbrNode
    Node nbr; int timestamp;

NeighborNodeEntry NeighborTable[];
Node Msg;
```

"free". A free node continues to explore candidate positions to join a tentacle by detecting the gradual change of radio signal strength of its neighboring nodes in the tentacle. Nodes that are part of a tentacle or are relaying network traffic remain relatively stationary, following a local optimization policy to gradually adjust their position to increase their relay bandwidth and throughput and hence overall network performance.

The TENTACLES algorithm consists of four parts: *Tentacle Building, Tentacle Rebuild, Radio Guided Exploration and Local Flow Optimization*. Table I shows the data structure used in the algorithm. Each node N stores its node data including node identifier $nodeID_N$, a tentacle array $tentacle_N[]$ storing the identifiers of all its ancestors up to a entity node and ending with its own $nodeID_N$ e.g. $[G_1,N_5, N_6]$, a distance vector array $distanceVector_N[]$ determining the criticality of the node in current network topology and a boolean flag specifying whether it is the end of a tentacle or not. This node data structure is sent periodically as a probe message to all its one-hop neighbors. All receiving nodes store or renew corresponding node information in *NeighborTable* with a new timestamp; expired information will be ignored. Once a node ceases to receive updated messages from its immediate parent node, it becomes a free node.

A. Tentacle Building

A tentacle always starts with an entity (root) node and ends with a robotic (leaf) node. The idea of tentacle building is to incrementally connect free nodes to the leaf node of a tentacle with a proper connection link with *Good* signal strength. Initially, a root node is also a leaf node. In summary, a free node considers joining a tentacle based on four criteria. (1) *Good Signal Rule* -To build a tentacle good for communication, the signal strength between the free node and the leaf node should be in the *Good* range (not in the *Strong* range). (2) *No Branching Rule* - To reach out as long as possible, tentacles may not branch. However, multiple tentacles are allowed to grow from the same entity node as long as the children of this entity node are not connected in *Good* signal range to each other. (3) *Weak Grandparent Rule* - To prevent "folding" of the tentacle, a free node joins a tentacle only when the signal strength to the parent of the leaf node is much weaker than to the leaf. (4) *Avoid Previous Parent Rule* - To avoid repeatedly

TABLE II
PSEUDO CODE FOR UPDATING DISTANCE VECTOR

```
dv[i] = infinity //for all node i is sink/source node
UpdateDistanceVector()
 FORALL (node i is a source or sink)
   dVMin= infinity
   IF (node i is the root of current path)
      dv[i] = path.size -1
   ELSE
      FORALL    (Neighbor n in NbrTable)
        IF    (SignalStrength(n) is better than Normal)
        THEN
             IF (n is a sink/source node)
                dv[n] = 1;
             ELSE
                 dVMin = min(dVMin, n.distanceVector[i])
                  IF     dv[i] < dVMin +1
                 THEN    dv[i] = infinity
                 ELSE    dv[i] = dVMin +1
            ENDIF
      ENDIF
    ENDFORALL
   ENDIF
 ENDFORALL
```

forming similar tentacles searching the same area, the free node will avoid joining previous parents within a certain time period.

Whenever a free node N detects the fulfillment of all four tentacle-building criteria, it stops its exploration and extends a tentacle from leaf node L. A *Join-Tentacle* request message is sent to node L and an acknowledgement message is received with node information. The data variable $isLeaf_N$ is set to *true* and $tentacle_N[]$ is updated by concatenating $tentacle_L[]$ with $nodeID_N$. For missing acknowledgements, node N resumes exploration. There are two mechanisms to prevent *"dead tentacle"* and *branching* effects. To prevent a *"dead tentacle"* with $isLeaf_L$ being *false*, node L only confirms node N received the acknowledgement from probe messages with the updated $tentacle_N[]$ from N. To prevent *branching* due to simultaneous *Join-Tentacle* requests from free nodes, a *Rebuild* message will be propagated from the branching node to a randomly selected branch to free the radio nodes.

B. Tentacle Rebuild

Tentacle rebuild aims at releasing non-critical radio nodes from tentacles. This involves evaluating the criticality of radio nodes and freeing up stationary nodes in a distributed way. Critical nodes are the ones (1) not connected to other entities or (2) not essential for carrying traffic between any two entities.

To identify the connectivity of a node between 2 entities, we define two node types, *closed* and *open* and assume network traffic always goes through *Normal* links between relay nodes and a shortest route from one entity to another entity with smallest hop count. *Closed* nodes are on the shortest path to another entity for every link from one entity. For example, in Figure 1 node S_1 has three links to node N_1, N_7 and N_9. Only links to N_1 and N_7 have a route to another entities. and therefore N_1, N_2, N_5, *and* N_7 are closed nodes. An *open* node is simply a non-closed node. It is either not on the shortest path to another entity, whether ot not it belongs to a tentacle. *Open* and *Closed* nodes are determined by a distance vector based on the *"distance"* (number of hops) from each entity initialized with infinity. A distance vector m is said to *dominate* another distance vector n when all entries in m is smaller or equal to the corresponding entries in n with at least one entry in m strictly smaller than that in n. Therefore, a closed node has its distance vector with at least 2 finite entries and not domninated by the one of any neighboring nodes. Table II shows the function for updating distance vectors distributedly. Each entry of a distance vector is updated by taking the minimum among the neighbors and adding one. If the corresponding node is in a tentacle, the entry will be based on the hop count distance from the root node.

Sometimes, *closed* nodes are not very useful for overall network performance with some adhoc network protocols, like OLSR[6]. Route selection is usually based on the cost of communication relating to the link quality and hop count. For example, in Figure 1, if the network protocol chooses hop count as its cost function, the route with node N_7 is sufficient for the traffic and the route with node N_1, N_2 and N_5 becomes redundant. These nodes can set themselves free if there is no network traffic going through for a period of time.

To release *open* nodes in a tentacle after a certain waiting period, a *Rebuild* message is initiated from an *open* leaf node and propagates to all the ancestors. A *closed* node will stop the propagation while an *open* node will forward the message and resume exploration upon receiving *Rebuild* message. For example, in Figure 1, N_4, N_6, N_8 and N_9 initiates *Rebuild* messages and together with N_3 free themselves from the tentacle. Each node remembers the identifier of its previous parent which it uses to avoid forming the same tentacle repeatedly.

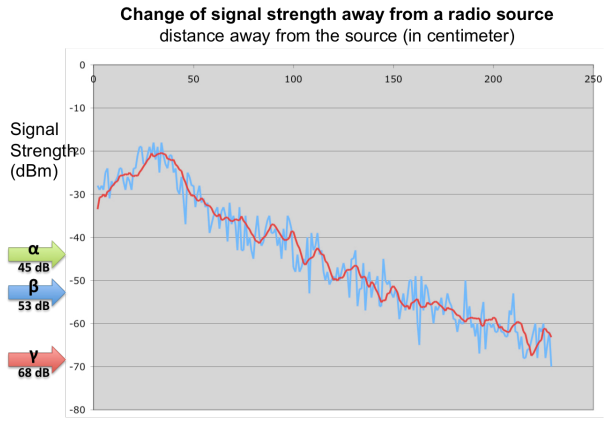

Fig. 2. Trend of signal strength signal. Raw data is in blue. Averaged data over last 10 raw data points is in red with α, β, γ threshold specified.

C. Radio Guided Exploration

To speed up tentacle creation, the gradient of radio signal strength may be a good source of localization information through triangulation. Figure 2 shows an experimental

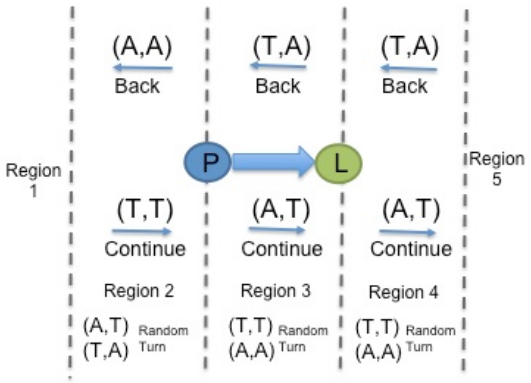

Fig. 3. Breakdown of decision making of radio guided exploration in a 2-node tentacle scenario.

measurement of a radio node approaching, passing by and leaving another radio node. Values are obtained through averaged smoothing of collected signal strength values. In our experimental setup, we obtained only ~2 dB difference over 1 meter, which made it hard to triangulate the position of other nodes in our indoor office-like environment.

To guide the exploration without triangulation, a radio node can hypothesize whether it is going *Towards (T)* a node or it is going *Away (A)* from another radio node by comparing its increased or decreased averaged values respectively to those at a previous location over a fixed travel distance, such as 1 meter. A decision on the continuation of current action can then be made based on the observed signal strength changes. A decision to "*Turn_Back"* means the node should reverse previous behaviors and search backwards. "*Continue"* means the node should continue with its previous behavior. "*Random Turn"* means the node should turn to a random direction and resume its exploration behavior.

To travel along a multi-node tentacle, we start with a simple case of two-node tentacle. Figure 3 shows five regions defined according to the relative signal strength between 2 nodes – node *P* (parent) and node *L* (leaf). Our goal is to move a free node from any region to the boundary of region 4 and 5 and join the tentacle.

Region 1: SignalStrength(L) is *Weak*
Region 2: SignalStrength(P) > SignalStrength(L)
Region 3: SignalStrength(P) > β and SignalStrength(L) >β
Region 4: SignalStrength(P) < Signal Strength (L)
Region 5: SignalStrength(P) is *Weak*

We also define signal trend tuple (T_P, T_L) with $T_P = \{A,T\}$ and $T_L = \{A,T\}$ indicating the movement trend of node *P* and *L* respectively. A decision can be made based on the region. For example, in Region 2, trend *(T,T)* means the exploration is moving towards node *P* and *L*. This behavior should be *Continued*. A *Random Turn* is taken for uncertain trends resulting from noise and the randomness can help the robotic node to leave the uncertainty area and local minima. By always considering two nodes closest to the leaf of the tentacle, a free node is able to traverse along a tentacle to its leaf and extend the tentacle.

D. *Local Flow Optimization*

Even though a node should remain stationary to maintain connectivity whenever it carries network traffic, a better bandwidth might result if a node just slightly explores the nearby area. For example, a node has joined a tentacle and just missed line-of-sight to some other node as it moved around a corner. In this case, a local search is essential to improve network performance.

Local Flow Optimization mode (LocalOpt) is activated for a node with a low traffic flow. The node goes forward for a fixed distance (1 meter) with one random direction out of eight 45-degree separated directions. The node stops whenever the new location is reached or hits an obstacle. It stays if it receives increased traffic, else it returns to its original position. If the node gets blocked while returning, the node quits LocalOpt and moves again only if there is insufficient traffic flow to anchor it in place.

To avoid destroying the link due to simultaneous movement, nodes are coordinated through the periodic probe messages to inform neighbors about current *LocalOpt* modes. Once neighboring nodes both enter *LocalOpt* mode, they go backward towards their original positions and quit the mode. They restart the mode only if they still carry low traffic flow after a random period of time.

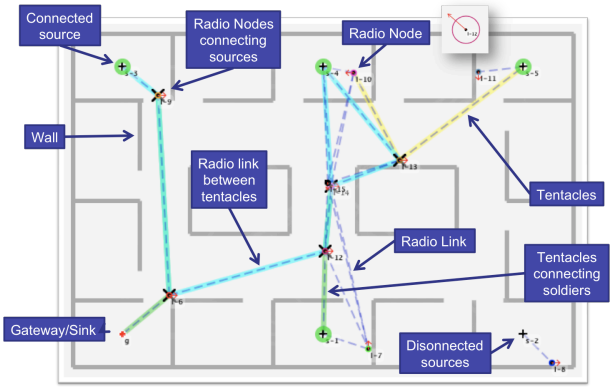

Fig. 4. A snapshot of simulation for *TENTACLE* algorithm and 5 source nodes have connected to the gateway.

IV. Simulation

Figure 4 shows a snapshot of a simulated scenario having five sources connected to a gateway (sink). Radio nodes (x) move around with wall-following behavior in an office-like environment to search for connections between source entities (+) and the gateway entity (.). Other legends are detailed in the figure.

In the simulation, radio signal strength decreases over distance based on an inverse square model. The strength of a radio signal is fractioned when it penetrates a wall. Each node is able to communicate and detect the signal strength of its connected neighbors. The TENTACLES algorithm is implemented without *Local Flow Optimization*.

Experiments have been carried out with three initial connections.

Fig.5. A radio node built by iRobot Create with Ubiquiti radio module and Gumstix Verdex.

TABLE III
PERFORMANCE OF TENTACLES IN SIMULATION

Condition	Signal Strength Noise stdv (dB)	Coverage (%)
Excellent	2.2	97.64
Excellent	7	97.07
Fair	2.2	94.26
Fair	7	93.15
Poor	2.2	83.26
Poor	7	80.58

Excellent – radio nodes connects source to gateway entity with *Good* link quality.

Fair – source and gateway are initially disconnected and only one radio node next to each entity.

Poor – same *as Fair* condition, but radio nodes are far away from gateway entity.

The performance *Coverage* measures the percentage of total sources that carried traffic to the sink over a period of time. Table III shows that the resulting network is able to cover at least 4 out of 5 sources most of the time, despite the noisy signal in the simulation.

Table IV gives the self-healing result of the simulation. The goal is to test the recovery of the network after nodes are turned off in steps. Results show at least 60% of the sources can still be connected all the time.

V. EXPERIMENTS ON REAL ROBOTS

Similar experiments were carried on the iCreate[7] platform. Figure 5 shows a picture of the robotic radio node. Each iCreate robot featured bumper sensors, an odometer, and wall following infrared sensors on its right. Movement commands were issued from a Gumstix Verdex Microprocessor. Each iCreate was equipped with an Ubiquiti 5 GHz XR5 module for radio communication with OLSR [6] as the routing protocol. In our experiment, all stationary entities had the same Ubiquiti radio unit connected to a Linux PC, which sent and received traffic flows to and from the network, but did not make any intelligent decisions about tentacle building. iCreate built-in wall following behavior was used as the primary exploration behavior in our experimental office environment.

A tough challenge for indoor environments is the unpredictable signal spikes due to multipath radio signals. For the rest of the section, the algorithm is able to guide a 7-robot radio network into a self-organizing and self-healing network. We have also experimented on how fast *Radio Guided Exploration* is able to speed up a free node joining a tentacle. The experiments were performed on a floor of Information Sciences Institute (ISI), about *30m x 40m*. The doors of the rooms were closed during experiments. Figure 6 shows the floor plan of this floor.

TABLE IV
PERFORMANCE OF SELF-HEALING IN SIMULATION

Time(sec)	Actions	Results
t=600	Powered off 4 nodes (Remain 6)	Self healed – no coverage loss
t=1200	Powered off 3 nodes (Remain 3)	Self healed – no coverage loss
t=1800	Powered off 1 nodes (Remain 2)	Lost one source (Supporting 4 sources)
t=2400	Powered off 1 nodes (Remain 1)	Lost one source (Supporting 3 sources)

t – time after starting up the nodes.

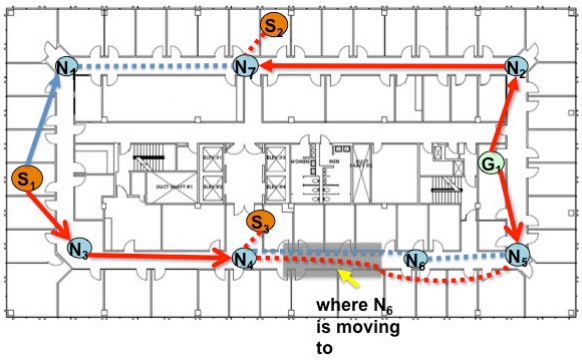

Fig. 6. A snapshot of the position of 7 robots during 3-source and 1-gateway experiment. Red lines are radio links with traffic.

A. Experiments on Self-Organization and Self-Healing

Seven robots were placed randomly on the hallways of the environment. Robots were started in pairs, with 45 seconds between pairs, to simulate the deployment delay of the robots in a real environment. Figure 6 shows a converged instance of one of the three runs where every source entity (S_1, S_2 and S_3) has a connected route (in red) to the gateway entity G_1 represented in green. In this instance, node N_1 was a closed node having distance vector $[G_1, S_1, S_2, S_3] = [3, 1, 2, \infty]$ while N_7 had [2,2,1, ∞] and S_1 had [4, 0, 3, 3]. N_1 was a *closed* node as it was not dominated by any other nodes. However, since there was no traffic going through for a certain time (two minutes in the experiment), node N_1 became a free node again.

We have also observed that the algorithm allows free nodes to improve the performance of the network. When node N_6 moved to the gray shaded area, it began to carry traffic. The underlying ad-hoc routing protocol in the network module (OLSR) switched network traffic from source S_1 and S_3 to use N_4 instead of N_6. Since N_1, N_4 did not carry any further traffic, they became free nodes after a two-minute timeout.

During the experiment, we also noticed that node N_2 had entered LocalOpt mode with traffic at half of the maximum throughput. After testing new locations for several times, the

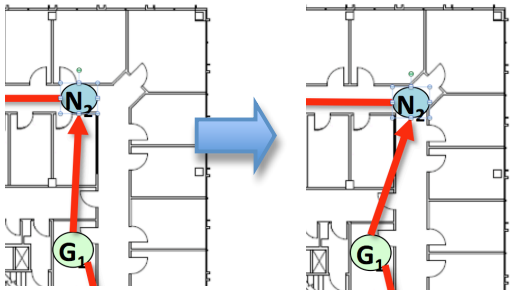

Fig. 7. Node N_2 moves to a new position and stays for better traffic flow

radio node ended up with a location with better line of sight as shown in Figure 7.

To test the capability of self-healing, node N_5 was turned off while N_6 was stationary and N_4 was a free node. Traffic from S_1 and S_3 stopped immediately. Nodes N_3, N_4 and N_6 resumed exploration and built tentacles in the lower hallway in approximately 10 minutes. These three nodes spread out evenly in the hallway and resumed carrying the traffic from S_1 and S_3 to G_1.

B. *Experiment of Radio Guided Exploration*

The convergence speed highly depends on how fast a radio node can explore to the leaf of a tentacle with *Radio Guided Exploration*. As shown in Figure 8, a source node was placed in an office in the middle of the hallway. A free radio node was switched on next to it. The node selected its exploration behavior every 15 seconds. With five runs using *Radio Guided Exploration*, the node picked whether to *Continue*, *Turn_Back*, or do a *Random Turn* every 15 seconds with the history of signal strengths. For five runs without Radio Guided Exploration, it made a *Random Turn* every 15 seconds before resuming wall following behavior. With a maximum of 15 minutes for each search, results showed random exploration around an average of 10.5 minutes with 4.5 minutes standard deviation to explore the shaded area. For the test using Radio Guided Exploration, it took only an average of 2.5 minutes with 30 seconds standard deviation to get to the right position and start a tentacle with the source entity. The decision made every 15 seconds had about 90% accuracy. This shows that *Radio Guided Exploration* is at least 3 times faster than random for a radio node to look for the leaf of a tentacle to join in hallway scenario.

VI. Conclusions and Future Work

This paper addresses the problem of connection establishment and maintenance between entities in an indoor unknown environment. A bio-inspired TENTACLES algorithm is introduced to stretch "tentacle" from stationary entities to connect another entity by incrementally connecting radio nodes to the leaf of tentacle. Tentacle rebuild frees up non-useful tentacle resources to build new connections. Local flow optimizations help to maximize traffic based on local view of network traffic condition.

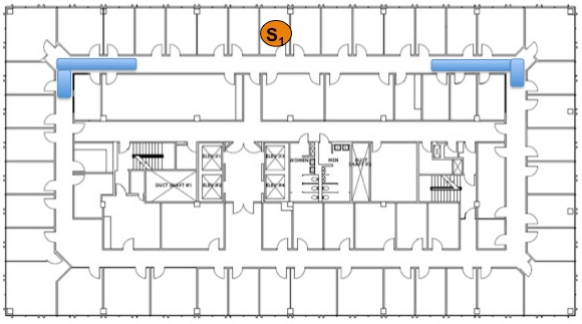

Fig. 8. Blue shaded areas are the candidate position for a radio node to start a tentacle with node S_1 as parent.

Experiments in simulation and practical robots show the solution converges with good coverage. In spite of the unknown environment, a noisy radio signal is able to give coarse direction, allowing free nodes to build tentacles more quickly.

Future work includes algorithm development with the removal of the stationary assumption of sources to allow dynamic movement of entities, exploration of environments with frequent branches (such as open rooms), and theoretical analysis on the relationship between convergence time and different parameters such as number of robotic nodes and tentacle rebuild time.

Acknowledgements

We are very grateful that robots and radio units were sponsored by the DARPA LANDroids program. We would like to thank the Information Sciences Institute of the University of Southern California and ArgonST for providing office areas for experiments.

References

1. Spears, William, M., Spears, Diana, F., Hamann, Jerry, C., & Heil, Rodney (2004) Distributed, Physics-Based Control of Swarms of Vehicles. *Auton. Robots* 17, 137--162.

2. Zhang, F., & Chen, W. (2007) Self-healing for mobile robot networks with motion synchronization. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on* , 3107-3112.

3. Zavlanos, M. M., & Pappas, G. J. (2005) Controlling Connectivity of Dynamic Graphs. *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on* , 6388-6393.

4. Howard, A., Matarić, M. J., & Sukhatme, G. S. (2002) An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots, Special Issue on Intelligent Embedded Systems* 13, 113--126.

5. Stump, E., Jadbabaie, A., & Kumar, V. (2008) Connectivity management in mobile robot teams. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* , 1525-1530.

6. Jacquet, P., Clausen, T., Laouiti, A., Qayyum, A., & Viennot, L. (2001) Optimized link state routing protocol for ad hoc networks. , 62--68.

7. iRobot. iRobot Create Owner's Guide. http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf.