

# Evaluation of a Large Scale Pervasive Embedded Network for Robot Path Planning

Keith J. O'Hara, Victor Bigio, Shaun Whitt, Daniel Walker and Tucker Balch

College of Computing  
Georgia Institute of Technology  
Atlanta, Georgia 30332-0250  
Email: kjohara@cc.gatech.edu

**Abstract**— We investigate a technique that uses an embedded network deployed pervasively throughout an environment to aid robots in navigation. First, we show that the path computed by the network is useful for a simple mobile robot. The robot uses a network of 156 nodes to navigate through a complex, dynamic, environment. This is the largest embedded network used for navigation we are aware of. In our approach, the network nodes do not need to know their absolute or relative positions and the mobile robots do not build any kind of map. Second, the impact of specific network deployments on path quality is examined. Two types of arrangements, hexagonal and rectangular, in two different environments are considered. We present quantitative results collected from a real-world embedded network of 60 nodes. Experimentally, we find that on average, the path computed by the network is only 24% longer than the optimal path. Also, we find a slight advantage for the hexagonal arrangement.

## I. INTRODUCTION

Previous researchers [1], [2], [4], [6] have investigated using embedded networks (e.g., a sensor network) to aid in robot navigation. The main idea is to use a pervasive network of computing, communicating, and possibly sensing devices to plan paths for mobile agents. In this type of *physical path planning*, the embedded nodes act as vertices of the path planning graph. In our approach, communication links represent unit-cost edges of the path planning graph. This assumption is valid as long as the communication graph approximates the path planning graph. Previous research [6], [7] has shown, qualitatively, that devices relying on infrared communication systems meet this requirement. In this work we look to further access this assumption.

First, we show how a simple mobile robot can use an embedded network of 156 nodes to navigate through a complex, dynamic, maze. This is the largest embedded network used for navigation we are aware of. We contend that the effectiveness of this and other approaches are best exhibited in real-world experiments with very large numbers of devices since this is where they draw their power. Logistical issues often prevent experiments with hundreds of devices, but to fully understand the behavior of swarm-like systems these issues must be overcome to enable real experimentation. Second, we examine the impact of specific network deployments on path quality. Two types of deployments, hexagonal and rectangular, in two different environments are considered.

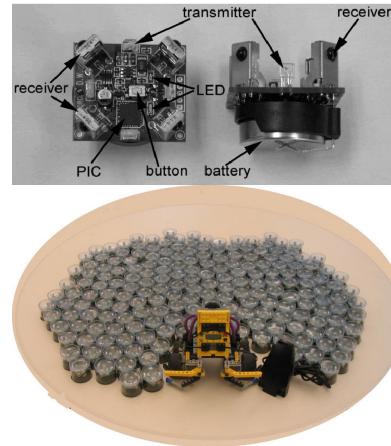


Fig. 1. The Gnats embedded network platform.

In our approach to physical path planning, neither the network nodes or the mobile robots need to know their positions or build any kind of map. The very nature of the network being pervasive and embedded in the environment provides, “World Embedded Computation” [7]. The idea is simple: rather than placing a map of the world in a centralized agent, many decentralized agents are placed into the map.

Although having estimates of the map or node positions might increase navigation performance, it is indeed possible for a robot to navigate through complex environments without any map building or localization. The fact that complex behavior such as global navigation can arise in such a simple system makes the approach an interesting candidate for study. Moreover, even compared to map-based navigation, using a pervasive network is particularly beneficial in two situations – dynamic environments and planning for multiple robots. In the first case, the network can detect environmental changes and reconfigure quickly, since a communication message can traverse the network faster than a robot. In the case of multiple robots, instead of the robots building their own maps and somehow merging them, the robots share a common map and common plan. The network could also perform robot coordination, for instance, directing coverage patterns, path de-confliction, or traffic control. We do not address multiple mobile robots in this paper, but plan that for future work.

## II. THE HARDWARE PLATFORM

The Gnats embedded network platform is used to perform physical path planning. The Gnats platform is flexible and inexpensive, but very resource constrained. The platform is pictured in Figure 1. The devices have 4 IR emitters, 4 IR receivers, a temperature sensor, a button, and two visible light LEDs. The Gnats rely on the PIC16F87 microcontroller for computation. The PIC’s oscillator is configurable from software, with a frequency range of 125kHz to 8MHz. It has 4K of programmable flash program memory and 368 bytes of RAM. There is an external 32K serial EEPROM for storage, and also an additional 256 bytes of backup EEPROM. A variety of sleep modes result in very long lifetimes and always-on functionality. The Gnats can be put to sleep and woken up using infrared messages, the button, or by timer.

The IR emitters have a maximum rating of 100mA and with this we have measured reliable communication over ranges of up to 5m in best-case configurations. The Gnats have been programmed to communicate with the LEGO Mindstorm/RXC robots allowing them to communicate with both RCX robots as well as the LEGO infrared PC dongle. This enables interaction with a PC, for instance, for downloading new programs, putting the devices to sleep, and reading the EEPROM.

The devices can be programmed individually via the MPLAB integrated development environment through a PIC-START PLUS device programmer. However, the best way to program many devices at once is using the LEGO infrared dongle. The PIC is self-programmable, meaning it can read and write its own program memory while running. This capability enables us to program the Gnats at runtime using the infrared communication system. We have programmed over 100 Gnats in under 3 minutes in this fashion. A system for dynamic code propagation is currently being developed.

## III. PHYSICAL PATH PLANNING

The distributed Bellman-Ford algorithm is used as our physical path planning algorithm. This algorithm was shown to result in physical path planning in previous work [6]. The distributed Bellman-Ford algorithm is a simple network routing algorithm [3] to find the shortest path to a destination from all nodes. With this method, one node in the network becomes the goal location. The Bellman-Ford equation for finding the shortest path from  $i$  to  $j$  is:

$$D(i, j) = \min_{k \in \text{neighbors}} d(i, k) + D(k, j)$$

where  $D(i, j)$  is the path cost from  $i$  to  $j$ , and  $d(i, k)$  is the distance between  $i$  and  $k$ . This algorithm is used to create a directed graph of shortest paths from every node to the goal. Note, in our approach the navigation distance is measured in hops. This only makes sense if our communication medium is local and blocked by obstacles to navigation. The hardware platform, the Gnats, meets these requirements. As we shall see in Section V, hops provides a good approximation to the true navigation distance in many environments.

```

structure PathMsg
  int hops
  int sequencenumber
  int gnatid
procedure INIT()
  hops ← MaxHops
  sequencenumber ← 0
  neighbors ← HashTable
on event MSGRECEIVED(PathMsg msg)
  ADD(neighbors, msg)
  for  $n$  in neighbors do
    if  $n.timestamp < ValidTimeout$  then
      if  $n.hops + 1 < hops$  then
        hops ←  $n.hops + 1$ 
      end if
    else
      DELETE(neighbors,  $n$ )
    end if
  end for
  if  $msg.sequencenumber > sequencenumber$  then
    sequencenumber ←  $msg.sequencenumber$ 
    TRANSMITMSG(hops, sequencenumber, gnatid)
  end if

```

Fig. 2. The Gnats path planning algorithm.

We can think of the network as routing robots to their destination. However, currently, the nodes do not know the global, or even local, position of their neighbors, so they do not direct the robot in any direction. Rather, the mobile robot greedily approaches the lowest-valued node. It will then come within communication range of that node’s parent and continue on toward that node, eventually reaching the goal.

The Bellman-Ford algorithm was implemented successfully on the Gnats platform. A brief outline of the algorithm follows. First, a device is designated as the goal. Next, the goal node begins to send messages to its neighbors, with its distance to goal (its hop-count) equal to zero, and with increasing sequence numbers. The rest of the network continues to propagate this “goal” information. The Gnats keep a neighbor list with information such as the neighbor’s id, the last time the device talked to its neighbor, and the neighbor’s hop-count. A neighbor is removed from the neighbor-list if a period of time has passed without any interaction. Then, each device looks through their neighbor-list for the neighbor with the minimum hop-count. This minimum hop-count is incremented by one and is propagated as the node’s hop-count. The gnats only propagate information when they receive a message with a higher sequence number than they have seen before, thus preventing cycles. The algorithm is sketched in Figure 2.

Typically, propagation of the goal information takes on the order of a couple of minutes in networks of around 100 nodes. Of course, this time is a function of environmental complexity, network deployment, ambient interference, etc. During the experiments the devices log their local connectivity information (neighbor-list) and their distance to the goal (hop-count). After the experiment, we uploaded the devices’ EEPROMs to a PC for post-processing. Using the data we create a graphic

```

procedure INIT()
  hops  $\leftarrow$  MaxHops
  gnatid  $\leftarrow$  0
  SETTIMER(MaxTimeout)
on event MSGRECEIVED(PathMsg msg)
  if msg.hops < hops or msg.gnatid = gnatid then
    SETTIMER(0)
    hops  $\leftarrow$  msg.hops
    gnatid  $\leftarrow$  msg.gnatid
  end if
procedure MAIN()
  INIT()
  loop
    if BUMPLEFT() then
      BACKUPANDTURNRIGHT()
    else if BUMPRIGHT() then
      BACKUPANDTURNLEFT()
    else if GETTIMER() > LargeTimeout then
      INIT()
      SPIN360DEGREES()
    else if GETTIMER() > SmallTimeout then
      SCANTURN()
    else
      GOSTRAIGHT()
    end if
    TRANSMITREQUESTMSG(gnatid, hops)
  end loop

```

Fig. 4. The mobile robot’s algorithm for navigating through the Gnats.

describing the experiment. Experiments using 60 Gnats are visualized in Figure 6.

For this approach to be successful, two criteria must be met. The area being traversed must be covered by Gnats (including the start and goal locations) and the network must be connected. The authors in [5] showed that these two criteria were the primary factors impacting embedded network navigation in a simulation study. In this work we assume the network has been deployed such that it is both connected, and it also covers the area to be navigated. We are interested in understanding how well the network can aid navigation given these two criteria are met.

#### IV. MOBILE ROBOT RESULTS

In addition to propagating messages from the goal, the network of Gnats are programmed to interact with a LEGO Mindstorm/RXC robot. The mobile robot is chiefly composed of a LEGO base, two motors, two bump sensors, an Hitachi H8 processor with 32K of RAM, and an infrared transceiver. The base of the robot is constructed such that Gnats can pass through the middle of its body, or they will be slid aside as it passes. The mobile robot was programmed with the BrickOS [9] software environment. BrickOS is a complete operating system for the RCX, allowing us to modify the communication protocol stack to be compatible with the Gnats. The Gnats and the RCX robots can exchange arbitrary messages over a 2400 baud infrared communication channel at ranges up to 3ft.

As described above, the Gnats communicate with each other to compute their distances to the goal. Then, the robot uses

a very simple algorithm to navigate through the environment. The robot continually polls nearby Gnats (approximately 3 times a second) for their hop-count. When the robot has not talked to any Gnats (either initially or after a time-out), the robot will spin in a complete circle keeping track of the lowest hop-count it sees. The robot then drives forward as long as it hears its current destination node or another node of lower hop-count. Note, once the robot has chosen a destination node, it will not consider other nodes of the same hop-count until it times out, thus preventing oscillations between nodes of the same hop-count. If the robot loses contact with its destination node, it turns until its realigned with the current destination node or another node of lower hop-count. The robot will time out completely after some long idle period. This method of spinning and constant communication provides the robot very rough azimuth information about nearby Gnats. The algorithm is sketched in Figure 4.

The global navigation information provided by the Gnats is complemented by a simple reactive response to its bump sensors. If the robot bumps into something, it backs up and turns a small amount away from the collision. This method allows the robot to traverse small obstacles not accounted for by the Gnats. The global path planning provided by the Gnats combined by the robot’s local reactive navigation results in the robot successfully navigating through complicated environments. The robot was able to use a network of 156 Gnats to navigate through a dynamic, complex, maze in under 10 minutes. Figure 3 shows the connectivity network of 156 Gnats aiding a robot in navigation, and a path the robot took.

Because of the mobile robot’s simple navigation algorithm, and even simpler hardware, its trajectory through the network is not as straight as we would hope. The robot occasionally loses communication with its destination node causing it to turn in circles until it regains contact. The hardware mismatch between the infrared systems of the robot and the gnats is the chief cause of this unreliability in communication.

Additionally, when the environment changes, the Gnats are able to detect this, replan, and offer the robot a new path. The speed of this response is a function of the timeout values of both the Gnats and the robot. Given a very reliable communication channel these timeouts values can be arbitrarily small. Sadly, our communication channel is not so reliable and we must have modest values for these timeouts. Despite the latency in reconfiguration (on the order of minutes), the time spent reconfiguring is much faster than having the robot perceive, map, and re-plan.

#### V. EXPERIMENTS

In motivating our technique, we assumed the communication graph approximates the path-planning graph, but how good is this approximation? After all, the Gnats communication is affected by ambient infra-red interference, unexpected interference by nearby nodes, reflections, hardware failure, etc. In order to better understand how the communication graph approximates the path-planning graph we empirically analyze two types of deployments, rectangular and hexagonal

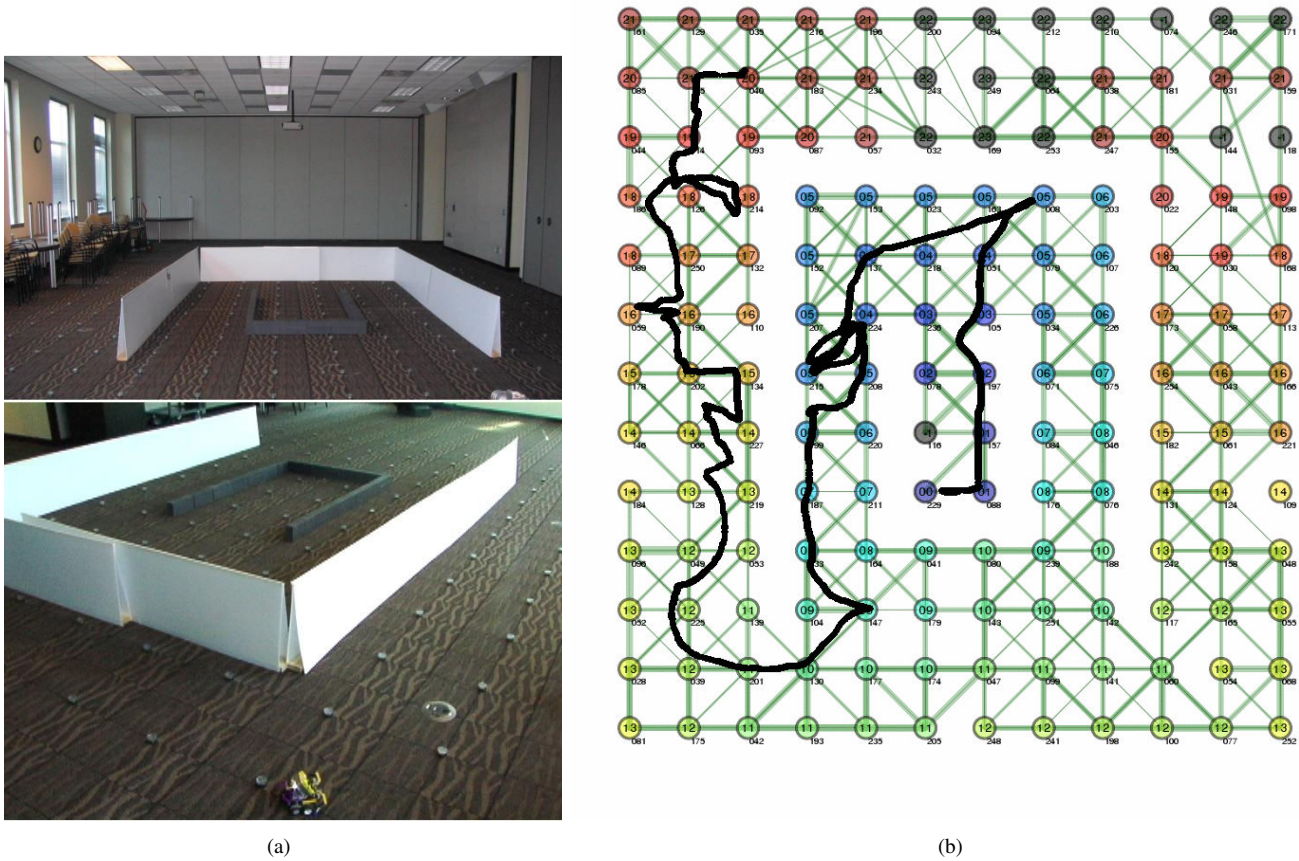


Fig. 3. (a) Two views of the maze. (b) The communication graph of 156 Gnats when node 229 is assigned to be the goal. The path of the robot is drawn in black.

	Rectangular		Hexagonal	
	No Obstacles	Box-Canyon	No Obstacles	Box-Canyon
Optimal Path Length (mean(ft.))	8.62	9.40	8.56	9.29
Expected Gnats Path Length (mean(ft.))	10.01	11.70	9.46	10.74
Actual Gnats Path Length(mean(ft.))	11.98	12.86	9.87	12.29
Average Difference from Optimal(mean(ft.)/stddev(ft.))	1.71/4.66	3.52/9.51	1.33/1.12	3.06/8.76
<b>Percentage Longer than Optimal (mean/stddev)</b>	<b>18.60/49.96</b>	<b>34.35/85.78</b>	<b>13.89/10.50</b>	<b>28.62/73.46</b>

Fig. 5. The average path lengths of the optimal path, the expected Gnats path, and the actual Gnats path.

arrangements, in two different environments. Although these arrangements seem somewhat ideal and unrealistic, topology control techniques (e.g. [8]) can be used to create similar uniform network overlays for navigation from real (i.e. non-uniform) sensor network deployments.

In all configurations, the environment is covered by a connected network of 60 Gnats. The first environment has no obstacles and the second environment has a box-canyon obstacle. In all cases the nominal communication range is set to 2-3 ft. In the rectangular configuration, the Gnats are separated by 2 ft, and the hexagonal arrangement is the rectangular with every other row shifted. The environments are pictured in Figure 6.

For each configuration we ran 10 different experiments. For

each experiment, a different node was designated as the goal and we logged the communication connectivity and hop-counts to EEPROM. Graphics depicting the network connectivity and hop-counts for four particular experiments are shown in Figure 6. The data collection in one experiment (rectangular-obstacle-254) failed so we do not include that in the analysis.

To better understand the quality of the computed paths, we compare three different path lengths. First, we compute *Optimal Path Length*, the length of the direct path a robot could take if it used a map of the environment instead of the Gnats. Second, *Expected Gnats Path Length*, the length of the path along the waypoints prescribed by the Gnats, assuming a perfect 2.5 ft. communication range. Finally, *Actual Gnats Path Length*, the length of the actual path in light

of imperfect or unexpected communication between Gnats during our experiments.

The last two path-lengths model a robot moving from node to node. In other words, the *Actual (Expected) Gnats Path Length* is the length of the path a robot would use to navigate through a field of Gnats. To compute the path-length in cases where a node has multiple parents, we average the parent's path-lengths. At any node the robot moves to one of the node's parents with equal probability. Thus, the path-length for a node is the average of the quantity - the distance to each parent plus the parent's path-length.

For all 40 experiments we compute the optimal path-length from each node to the goal. Also, we compute actual Gnats path-length, i.e., if a robot were to go from node to node following the computed hop-counts as described above. Using these two numbers we calculate, on average, how much longer the Gnats path is than the optimal path. The results are presented in Figure 5. We see that the hexagonal deployment is on average, closer to the optimal than the rectangular. More importantly, the computed path-lengths are on average (over all configurations), only 24% longer than optimal.

## VI. RELATED WORK

A number of researchers [1], [2], [4], [6] have investigated using embedded networks (e.g. a sensor network) to aid in robot navigation. Batalin et al. [2] use a network of Motes to aid in coverage, navigation, and task allocation. Specifically, they employ a technique called "Distributed Value Iteration". In their approach, transition probabilities between nodes for each navigation action are determined during deployment. Then, the network uses these probabilities and a cost function to compute a policy of action for the robot. They demonstrated their approach using a Pioneer mobile robot in an office environment using a network of 9 Motes.

Li et al. [4] use a network of 49 Motes to compute a potential field for navigation. The embedded nodes are able to sense and propagate "danger-ness" allowing them to plan around areas that would hinder navigation. In a manner similar to [6], [7] we don't need to explicitly model obstacles, but rather use properties of the communication network to plan around obstacles. They evaluated their technique on various network topologies and found the shortest and safest paths. Due to the differences between the Gnats and Motes communication systems, their communication network is much less local than ours, often resulting in long-range unexpected connections which can be detrimental in supporting navigation.

We used the algorithm of O'Hara et al. [6] in this work. They showed their network of 19 nodes was able to perform physical path planning in a dynamic environment. Additionally, simulation studies [5] showed how the embedded network could support effective cooperative foraging. Quantitative simulation results illustrated the sensitivity of the approach to different network sizes, environmental complexities, and deployment configuration. Alankus et al. [1] develop an approach where a sensor network is used in conjunction with a probabilistic roadmap method to help in mobile robot navigation.

They demonstrated their system with a Pioneer mobile robot and 7 Motes. Payton et al. developed ideas like "World Embedded Computation" in their approach to controlling a 20 robot, mobile swarm. In their approach, many mobile robots use short-range infrared communication to navigate and sense in a distributed fashion.

## VII. DISCUSSION

First, a simple mobile robot used a network of 156 nodes to navigate through a complex, dynamic, environment. This is the largest embedded network used for navigation we are aware of. We contend that the effectiveness of this and other approaches are best exhibited in real-world experiments with very large numbers of devices since this is where they demonstrate their power. Logistical issues often prevent experiments with hundreds of devices, but to fully understand the behavior of swarm-like systems these issues must be overcome to enable real experimentation. Second, we presented results for two different network deployments, in two different environments. We found that the embedded network provides a path that is on average 24% longer than optimal.

In future work, we plan to compare these results to random deployments of different densities. Random deployments are better models for networks dropped from planes or ships, and networks that have degraded over time. In addition, random deployments offer no guarantees on connectivity or coverage. We also plan on investigating ways the network can coordinate the activities of multiple mobile robots, for instance, providing path de-confliction and traffic management services.

**Acknowledgments.** This work was supported by the National Science Foundation under award #0326396. The authors would like to thank Ben Axelrod, Can Envarli, and Arya Irani for help with the experiments and useful discussion as well as Shaleen Harlalka for help building the robot base.

## REFERENCES

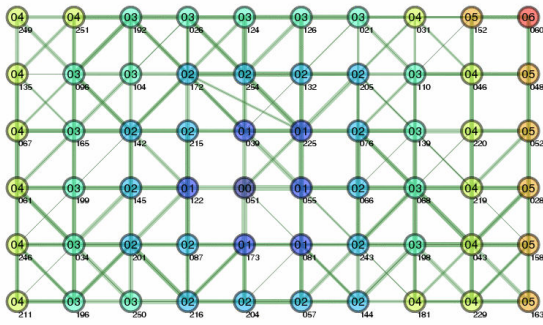
- [1] G. Alankus, N. Atay, C. Lu, O.B. Bayazit, "Spatiotemporal Query Strategies for Navigation in Dynamic Sensor Network Environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, Edmonton, Canada, August 2005.
- [2] M. Batalin, G. S. Sukhatme, and M. Hattig, "Mobile Robot Navigation using a Sensor Network," *IEEE International Conference on Robotics and Automation (ICRA '04)*, pp. 636-642, New Orleans, Louisiana, April 2004.
- [3] L. Ford, D. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.
- [4] Q. Li, M. DeRosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," *9th International Conference on Mobile Computing and Networking*, pp. 313-325, September 2003.
- [5] K. O'Hara and T. Balch, "Pervasive Sensor-less Networks for Cooperative Multi-Robot tasks," *7th International Symposium on Distributed Autonomous Robotic Systems*, June 2004.
- [6] K. O'Hara, V. Bigio, E. Dodson, A. Irani, D. Walker, T. Balch, "Physical Path Planning Using the GNATS," *IEEE International Conference on Robotics and Automation (ICRA '05)*, Barcelona, Spain, March 2005.
- [7] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone Robotics", *Autonomous Robots*, vol. 11, pp. 319-324, 2001.
- [8] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing," *ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001.
- [9] <http://brickos.sourceforge.net/>



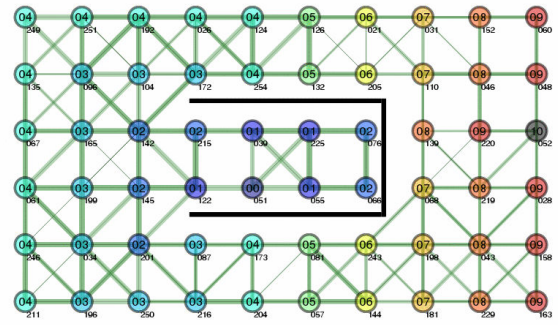
(a)



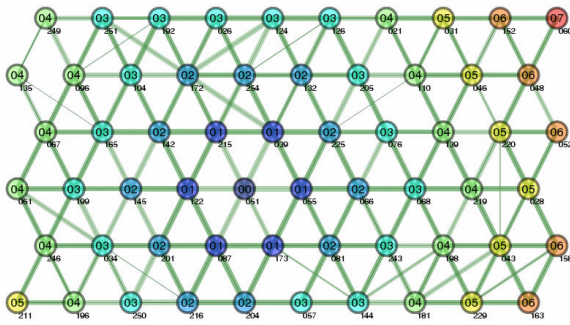
(b)



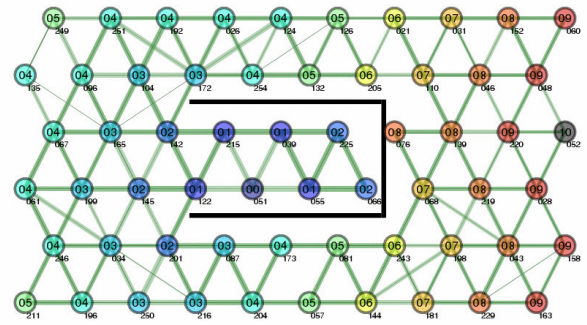
(c)



(d)



(e)



(f)

Fig. 6. (a) The rectangular arrangement with no obstacles. (b) The hexagonal arrangement with a box canyon. (c-f) Visualizations of the data from a network of 60 Gnats showing the hop-counts and connectivity when node 51 acts as the goal node. Inner number indicates hop count, outer number indicates device id, thicker lines represent more connectivity.