Improvements on Ant Routing for Sensor Networks

Ying Zhang[1], Lukas D. Kuhn[2], and Markus P.J. Fromherz[1]

[1] Palo Alto Research Center, Palo Alto, CA, USA,
`yzhang,fromherz@parc.com`
[2] Ludwig Maximilian University, Munich, Germany

Abstract. Ad-hoc wireless sensor networks have been an active research topic for the last several years. Sensor networks are distinguished from traditional networks by characteristics such as deeply embedded routers, highly dynamic networks, resource-constrained nodes, and unreliable and asymmetric links. Ant routing has shown good performance for communication networks; in this paper, we show why the existing ant-routing algorithms do not work well for sensor networks. Three new ant-routing algorithms are proposed and performance evaluations for these algorithms on a real application are conducted on a routing simulator for sensor networks.

1 Motivation

Large-scale ad-hoc networks of wireless sensors have become an active topic of research [6]. Such networks share the following properties:

– *embedded routers* – each sensor node acts as a router in addition to sensing the environment;
– *dynamic networks* – nodes in the network may turn on or off during operation due to unexpected failure, battery life, or power management [2]; attributes associated with those nodes (locations, sensor readings, load, etc.) may also vary over time;
– *resource constrained nodes* – each sensor node tends to have small memory and limited computational power;
– *dense connectivity* – the sensing range in general is much smaller than the radio range, and thus the density required for sensing coverage results in a dense network;
– *asymmetric links* – the communication links are not reversible in general.

Applications of sensor networks include environment monitoring, traffic control, building management, and object tracking. Routing in sensor networks, however, has very different characteristics than that in traditional communication networks. First of all, address-based destination specification is replaced by a more general feature-based specification, such as geographic location [7] or information gain [3]. Secondly, routing metrics are not just shortest delay, but also energy usage and information density. Thirdly, in addition to peer-to-peer communication, multicast (one-to-many) and converge-cast (many-to-one) are major traffic patterns in sensor networks. Even for peer-to-peer communication, routing is more likely to be source or destination driven than table-based [9], and source/destination pairs often are dynamic (changing from time to time) or mobile (moving during routing).

The characteristics of the *Ant System* [5] — positive feedback, distributed computation, and constructive greediness — seem to fit the domain of sensor networks very well. Various ant-routing algorithms have also been proposed for telecommunication networks [1, 4, 10], which demonstrated good performance for those networks [1]. We started by implementing an ant-routing algorithm with the basic ideas from AntNet [1], which we'll call the *basic* ant routing. Due to the properties of highly dynamic nodes and asymmetric links in sensor networks, the basic ant routing did not perform well. We then developed three improved versions of ant routing based on the message-initiated constraint-based routing framework [12]:

– *Sensor-driven and cost-aware ant routing (SC)*: One of the problems of the basic ant-routing algorithm is that the forward ants normally take a long time to find the destination, even when a tabu list is used (i.e., no repeating nodes if possible). That happens because ants initially have no idea where the destination is. Only after one ant finds the destination and traverses back along the links will the link probabilities of those links change. In SC, we assume that ants have sensors so that they can smell where the food is even at the beginning. That is not an unrealistic assumption for sensor networks, since feature-based routing dominates address-based routing in that space. Some features, such as geographic location, have a natural *potential field*. If the destination does not have a clear hint, pre-building the feature potential is sometimes still efficient. Cost awareness generalizes the objective of shortest path length so that ants can apply other routing metrics as well, e.g., energy-aware routing.
– *Flooded forward ant routing (FF)*: Even augmented with sensors, forward ants can be missguided due to obstacles or moving destinations. Flooded forward ant routing exploits the broadcast channel in wireless sensor networks. When a forward ant starts at the source, it tells all its neighbors to look for the food, and neighbors tell neighbors and so on, until the destination is found. Ants then traverse backward to the source and leave pheromone trails on those links. To control the flooding, only those ants who are "closer" to the food will join the food searching process.
– *Flooded piggybacked ant routing (FP)*: Single path routing tends to have high loss rates, due to dynamic and asymmetric properties of sensor networks. Multi-path routing such as flooding is very robust and has high success rate. In FP, we combine forward ants and data ants, using constrained flooding in the previous algorithm to route the data and to discover good paths at the same time.

The rest of the paper is organized as follows. Section 2 presents the basic ant-routing algorithm. Section 3 develops three improved ant-routing algorithms. Section 4 evaluates all four ant-routing algorithms in a routing simulator for sensor networks using a real application scenario. Section 5 concludes the paper.

2 Basic Ant Routing

Informally, the basic ant routing can be described as follows (cf. [1]):

- At some intervals, which may vary with time, a forward ant is launched from the source node toward the destination node.
- Each forward ant searches for the destination by selecting the next hop node according to the link probability distribution. Initially all the links have equal probability.
- While moving forward, each forward ant remembers the list of nodes it has visited and tries to avoid traversing the same node.
- Once a forward ant finds the destination, a backward ant is created, which moves back along the links that the forward ant had traversed.
- During the backward travel, the cost from the destination to each node in the path is recorded; rewards are then given according to the relative goodness of the path. Probabilities of the nodes in the path are updated according to the rewards.
- Once a backward ant arrives at the source, the next launch interval is calculated according to the relative goodness of the whole path.

At each node, like AntNet [1], the link probability distribution p_n is maintained for each neighbor node n, with $\sum_{n \in N} p_n = 1$, where N is the set of neighbor nodes. Initially, $p_n = 1/|N|$. In addition, the average cost (e.g., the number of hops) μ and the variance σ^2 from the current node to the destination is updated by the backward ants as follows. Let C be the current cost of the path from the destination to the current node:

$$\mu \leftarrow \mu + \eta(C - \mu), \sigma^2 \leftarrow \sigma^2 + \eta((C - \mu)^2 - \sigma^2) \tag{1}$$

An observation window W of size M is kept for storing the cost of past M paths, so that the minimum cost within the window W can be obtained. In this case, we set $\eta = \min(5/M, 1)$ as in [1].

Given a reward $r \in [0, 1]$, the probability distribution on the links is updated as follows. Assuming the backward ant is coming from node $m \in N$, then [3]:

$$p_m \leftarrow p_m + r(1 - p_m), p_n \leftarrow p_n - rp_n, n \in N, n \neq m \tag{2}$$

The reward r can be simply a constant, e.g., 0.5; however, it works better when r is cost-sensitive. For example, we use

$$r \leftarrow k_1\left(\frac{C_{inf}}{C}\right) + k_2\left(\frac{C_{sup} - C_{inf}}{(C_{sup} - C_{inf}) + (C - C_{inf})}\right) \tag{3}$$

[3] Another update rule, $p_m \leftarrow \frac{p_m + r}{1+r}$ and $p_n \leftarrow \frac{p_n}{1+r}$, can be used as well. According to our experiments, there is no significant difference from (2).

where $C_{inf} = \min(W)$, $C_{sup} = \mu + z(\sigma/\sqrt{M})$, $k_1 + k_2 = 1$, and $z \in R^+$. This formula guarantees that $r \in (0, 1]$. To slow down the learning effect in (2), one may use $r \leftarrow \alpha r$, where $\alpha \in (0, 1)$ is a learning rate.

When a backward ant arrives at the source, the forward ant launch interval I is updated by $I \leftarrow e^{r-0.5} I$, where r is obtained from (3). The intuition is to increase I if the path has relatively lower cost (or higher reward with $r > 0.5$), and to reduce I otherwise.

The pseudo code for the basic ant routing is presented in Figure 1. Each forward ant carries a tabu list to avoid loops if possible. If the loop size is larger than half of the list size, the forward ant dies; otherwise the loop is removed from the list and the ant continues. Each backward ant follows the tabu list and updates the statistics and the probability distribution on the way back to the source. The cost is the number of hops in this basic algorithm. The forward ant interval is updated according to the cost of the whole path at the source. Like AntNet [1], the data ants are prevented from choosing links with very low probability by remapping p to p^β for $\beta > 1$.

The algorithm assumes that a neighborhood structure is established initially. To establish a neighborhood structure, each node sends out "hello" packets a few times at the beginning. A node u is a neighbor of node w if w can hear packets from u. For asymmetric links, u being a neighbor of w does not imply that w is a neighbor of u. When the ant routing starts, the probabilities are uniformly distributed among the neighbors.

The problem with the basic ant routing is that the forward ants initially have a hard time to find the destination. It is equivalent to random walks in a maze with no hint. Because of the density property (large neighborhood), the problem becomes more severe. The probability distributions are not modified until the first forward ant arrives at the destination and traverses back. Due to asymmetric links, ants who successfully reach the destination may not be able to move back to the source, causing the updates to happen even more slowly. Collisions and failure nodes contribute to the low performance as well.

3 Improved Ant-routing Algorithms

3.1 Sensor-driven Cost-aware Ant Routing (SC)

To improve the performance of the forward ants, we equip ants with sensors to sense the best direction to go even initially. In addition to storing the probability distribution, each node estimates and stores the cost to the destination from each of its neighbors. Assume that the cost estimation is Q_n for neighbor n. The cost from the current node to the destination is 0 if it is the destination, otherwise $C = \min_{n \in N} (c_n + Q_n)$, where c_n is the local cost function. The initial probability distribution is calculated according to

$$p_n \leftarrow \frac{e^{(C-Q_n)^\beta}}{\Sigma_{n \in N} e^{(C-Q_n)^\beta}} \tag{4}$$

```
**received** forward ant $f$ at node $w$ from node $m$ **do**
  **if** destination($w$) **then**
    $b.cost \leftarrow 0$; $b.list \leftarrow f.list$;
    **send** $b$ to $b.list[1]$;
  **else**
    $f.list \leftarrow [w, f.list]$; % insert $w$ at the head of the list
    $N^* \leftarrow N \backslash f.list$;
    **if** $N^* = \emptyset$ **then** $N^* \leftarrow N$; **end**
    choose $n' \in N^*$ according to $\frac{p_{n'}}{\Sigma_{n \in N^*} p_n}$
    **if** $n' \in f.list$ **then**
      $i \leftarrow find(f.list, n')$; $L \leftarrow |f.list|$;
      **if** $L < 2i$ **then return**; **end**
      $f.list \leftarrow f.list[i+1 : L]$; % remove the loop
    **end**
    **send** $f$ to $n'$;
  **end**
**end**

**received** backward ant $b$ at $w$ **do**
  $b.cost \leftarrow b.cost + c_w$;
  $\mu \leftarrow \mu + \eta(b.cost - \mu)$; $\sigma^2 \leftarrow \sigma^2 + \eta((b.cost - \mu)^2 - \sigma^2)$;
  $W \leftarrow [b.cost, W]$;
  **if** $|W| > M$ **then** $W \leftarrow W[1 : M]$; **end** % keep only $M$ costs
  calculate $r$ according to (3);
  **if** source($w$) **then** $I \leftarrow e^{r-0.5} I$;
  **else**
    $r \leftarrow \alpha r$;
    calculate $p$ according to (2);
    $L \leftarrow |b.list|$; $n' \leftarrow b.list[1]$; $b.list \leftarrow b.list[2 : L]$;
    **send** $b$ to $n'$;
  **end**
**end**

**received** data ant $d$ **do**
  choose $n' \in N^*$ according to $\frac{p^{\beta}_{n'}}{\Sigma_{n \in N^*} p^{\beta}_n}$;
  **send** $b$ to $n'$;
**end**

**timeout** release forward ant $f$ at node $w$ **do**
  **if** source($w$) **then**
    $f.list \leftarrow [w]$;
    choose $n'$ according to $p_n$;
    **send** $f$ to $n'$;
  **end**
  set next timeout after $I$;
**end**
```

Fig. 1. Basic ant-routing algorithm

The exponential term in this formular makes the probability distribution differ more, thus even more favoring the good links.

There are two ways in which Q_n can be obtained. First, for the feature-based destination specification, one may estimate the "distance" from the current node to the destination. For example, for geographic routing, one may estimate the node's cost by $k\sqrt{(x_d - x)^2 + (y_d - y)^2}$, where (x, y) is the location of the current node and (x_d, y_d) is the location of the destination. During initialization, each "hello" packet also includes the cost of a node. After initialization, each node has the costs of its neighbors. Alternatively, if the destination is known initially, which is the case for many sensor network applications, one can get the cost by flooding from the destination. The pseudo code for cost estimation and ant initialization is shown in Figure 2.

```
received initialization ant $i$ from $u$ do
  $Q_u \leftarrow i.cost$;
  $i.cost \leftarrow \min_{n \in N} (c_n + Q_n)$;
  broadcast $i$;
end

initialization at node $w$ do
  if destination($w$) then
    $i.cost \leftarrow 0$;
    broadcast $i$;
  end
end

ant-start at node $w$ do
  $p_n \leftarrow \frac{e^{(C-Q_n)^\beta}}{\Sigma_{n \in N} e^{(C-Q_n)^\beta}}$;
  if source($w$) then
    release forward ant;
  end
end
```

Fig. 2. Initialization for cost estimation

3.2 Flooded Forward Ant Routing (FF)

If the destination is unknown initially or the cost estimation cannot be derived from the destination specification (e.g., address-based destination), SC reduces to the basic ant routing, and the problem of wandering around to find the destination still exists. A solution to that problem is to exploit the broadcast channel of wireless sensor networks. The idea is to flood forward ants to the destination;

as before, successful forward ants will create backward ants to traverse back to the source. Multiple paths are updated by one flooding phase. Probabilities are updated in the same way as in the basic ant routing. The flooding can be stopped if the probability distribution is good enough for the data ants to the destination. In our case, we reduce the rate for releasing the flooding ants when a shorter path is traversed.

Two strategies are used to control the forward flooding. First, a neighbor node will broadcast a forward ant to join the forward search only if it is "closer" to the destination than the node that broadcasted at an earlier time. Link probabilities are used for the estimation, i.e., a forward ant is to broadcast only if $p_n < 1/|N|$, where n is the neighbor the ant is coming from and N is the set of neighbors. If initially there is no hint, i.e., $p_n = 1/|N|$ for all n, each node will broadcast once. Secondly, delayed transmission is used in that a random delay is added to each transmission, and if a node hears the same ant from other nodes, it will stop broadcasting. The pseudo code for the flooded forward ant routing is shown in Figure 3. The problem with FF is collisions. The flooded forward ants

```
received forward ant $f$ at node $w$ from node $m$ do
  if destination($w$) then
    $b.cost \leftarrow 0$; $b.list \leftarrow f.list$;
    send $b$ to $b.list[1]$;
  elseif $new(f)$ and $p_m < 1/|N|$ then
    $f.list \leftarrow [w, f.list]$;
    broadcast $f$ after random delay $t$ if not heard $f$ within $t$;
  end
end

timeout release forward ant $f$ at node $w$ do
  if source($w$) then
    $f.list \leftarrow [w]$;
    broadcast $f$;
  end
  set next timeout after $I$;
end
```

Fig. 3. Flooded forward ant routing

create a significant amount of traffic, which interferes with the data ants and the backward ants. Controlling the frequency of the forward flooding becomes the key for this method to succeed.

3.3 Flooded Piggybacked Ant Routing (FP)

The flooding mechanism in wireless networks is very robust and works extremely well when the network is highly dynamic. In FP, we combine forward ants with data ants, with the data ants carrying the forward list. The same strategy to control the flooded forward ants as in FF is used to control the flooded data ants. In this case, the data ants not only pass the data to the destination, but also remember the paths which can be used by the backward ants to reinforce the probability on these links. The probablity distribution constrains the flooding towards the destination for future data ants. This method has a very high success rate with a relatively high energy consumption. Figure 4 shows the fragment of the code.

```
received data ant $d$ at node $w$ from node $m$ do
  if destination($w$) then
    $b.cost \leftarrow 0$; $b.list \leftarrow f.list$;
    send $b$ to $b.list[1]$;
  elseif $new(d)$ and $p_m < 1/|N|$ then
    $d.list \leftarrow [w, d.list]$;
    broadcast $d$ after random delay $t$ if not heard $d$ within $t$;
  end
end
```

Fig. 4. Flooded piggybacked ant routing

4 Performance Evaluations

We have simulated these routing strategies using Prowler [11], a probabilistic wireless network simulator designed for the Berkeley Motes.

4.1 Network Model

Prowler [11], written in Matlab, is an event-driven simulator that can be set to operate in either deterministic mode (to produce replicable results while testing an algorithm) or in probabilistic mode (to simulate the nondeterministic nature of the communication channel). Prowler consists of a *radio propagation model* and a *MAC-layer model.*

The radio propagation model determines the strength of a transmitted signal at a particular point of the space for all transmitters in the system. Based on this information the signal reception conditions for the receivers can be evaluated and collisions can be detected. The transmission model is given by:

$$P_{rec,ideal}(d) \leftarrow P_{transmit}\frac{1}{1+d^{\gamma}}, \quad where\ 2 \leq \gamma \leq 4 \tag{5}$$

$$P_{rec}(i,j) \leftarrow P_{rec,ideal}(d_{i,j})(1+\alpha(i,j))(1+\beta(t)) \tag{6}$$

where $P_{transmit}$ is the signal strength at the transmission and $P_{rec,ideal}(d)$ is the *ideal* received signal strength at distance d, α and β are random variables with normal distributions $N(0, \sigma_\alpha)$ and $N(0, \sigma_\beta)$, respectively. A network is asymmetric if $\sigma_\alpha > 0$ or $\sigma_\beta > 0$. In (6), α is static depending on locations i and j only, but β changes over time. Furthermore, an additional parameter p_{error} models the probability of a transmission error by any unmodeled effects. The MAC-layer simulates the Berkeley motes' CSMA protocol, including random waiting and back-offs.

4.2 Performance Metrics

Various performance metrics are used for comparing different routing strategies in sensor networks. We have used the following:

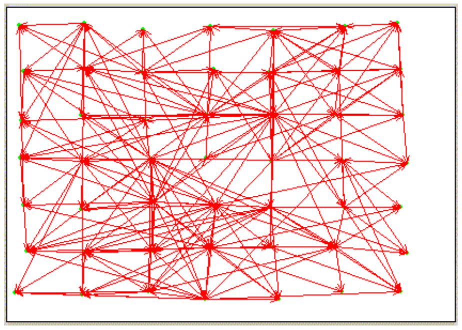
– *latency* – the time delay of a packet from the source to the destination;
– *success rate* – the total number of packets received at the destinations vs. the total number of packets sent from the source;
– *energy consumption* – assuming each transmission consumes an energy unit, the total energy consumption is equivalent to the total number of packets sent in the network;
– *energy efficiency* – the ratio between the number of packets received at the destination vs. the total energy consumption in the network;

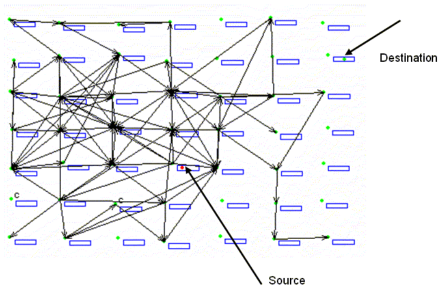
Some of these metrics are correlated and some conflict with each other. For different applications, different metrics may be relevant.

4.3 Example: Pursuer Evader Game

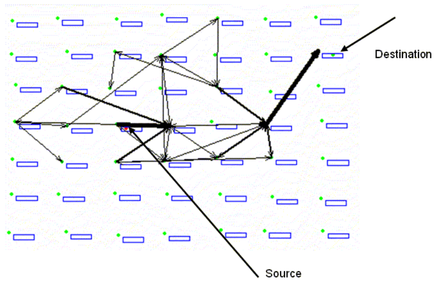
We use a real application to test the performance of these ant-routing algorithms. The Pursuer Evader Game (PEG) has been a standard testbed for sensor networks. In this testbed, sensors are deployed in a regular grid with random offsets. An evader is a tank or a car whose movement can be detected by the sensor nodes. A pursuer is another vehicle that is going to track the evader down. The communication problem in this task is to route packets sent out by sensor nodes who detected the evader to the mobile pursuer.

In our test, the network is a 7×7 sensor grid with small random offsets. For Berkeley motes, the maximum packet size is 960 bits and the radio transmission rate is 4Kbps. The default settings of Prowler are used, i.e., $\sigma_\alpha = 0.45$, $\sigma_\beta = 0.02$, and $p_{error} = 0.05$. The transmit signal strength is set to 1, and the maximum radio range is about $3d$ (a dense network), where d is the standard distance between two neighbor nodes in the grid. Figure 5 (a) shows an instance of the connectivity of such a network. Figures 5 (b) (c) and (d) show snapshots of ant traces during the first 100 seconds assuming both the evader and the pursuer are stationary. Figure 5 shows that the forward ants easily get lost in the basic ant routing, whereas SC can find destinations easily without wandering too much, and FF has shorter paths overall.

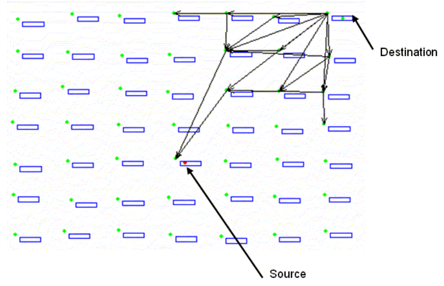

(a)



(b)



(c)



(d)

Fig. 5. (a) Instance of radio connectivity (b) Traces of the forward ants in the basic ant routing (c) Traces of forward ants in SC (line thickness corresponding to the probability of the link) (c) Traces of backward ants in FF

In the performance simulation, the source is changing from node to node, following the movement of the evader, and the destination is mobile. The average speed of both the evader and the pursuer is set to $0.01d/s$. The source rate is about 1 packet per second. For the basic ant routing, SC and FF, the ratio between data ants and forward ants is set to be 2 initially. The forward ant interval is changing according to the quality of the discovered path, but the data rate is fixed. There is an initialization phase of 3 seconds for establishing the connectivity of the network. In the basic ant routing, 3 "hello" packets are sent out from each node; for SC, FF, and FP, 3 backward floodings are used to establish the potential field and the initial probability distribution of links. Both initializations use about the same amount of energy, i.e., $3N$, where N is the number of nodes in the network. A total of 200 seconds are simulated for the four ant-routing algorithms, with 10 random runs for each.

Performances are compared among the four ant-routing algorithms for this application. Figure 6 shows the average success rate, latency, total energy consumption, and energy efficiency. The basic ant routing has extremely poor success rates. FP has the highest success rates but is not very energy efficient. FF has in general the shortest delays, and SC is most energy efficient. We have

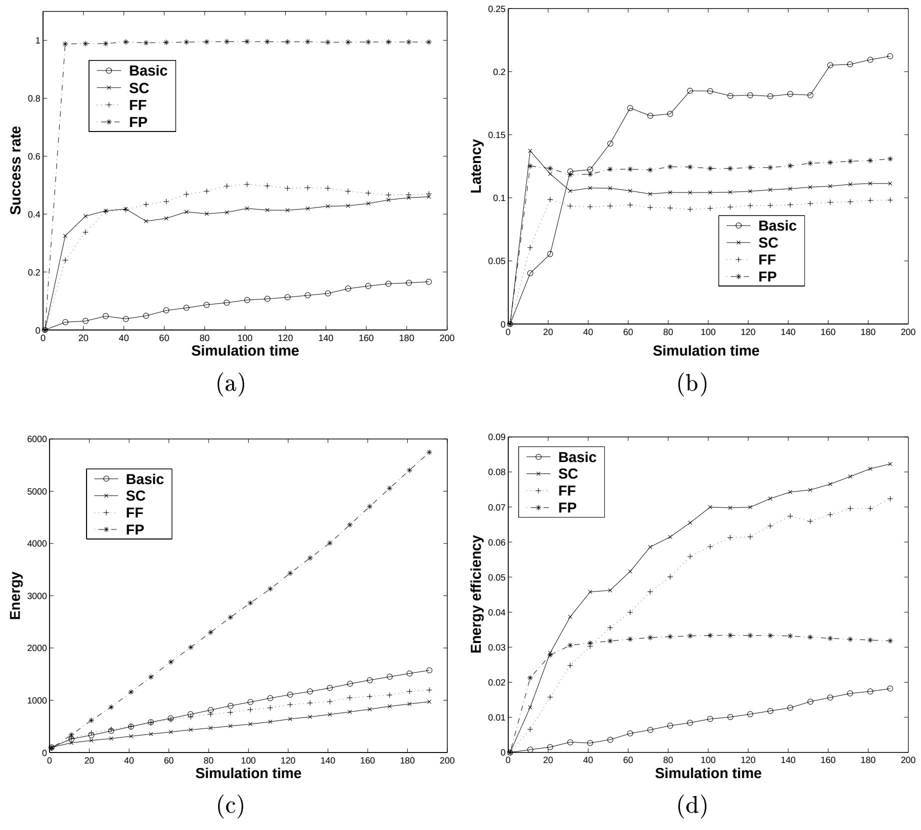


Fig. 6. Performance evaluation among four ant routings: (a) Success rates (b) Latency (c) Energy consumption (c) Energy efficiency

developed other routing strategies for sensor networks [12]. Comparisons with these and other routing strategies will be discussed in future papers. The biggest problem with these ant-routing algorithms is the use of forward lists, which introduces limitations for small packet sizes in large-scale networks. Furthermore, even though ant-routing algorithms work for asymmetric networks, they all reinforce the best symmetric links. If a network does not have a symmetric path, these algorithms do not work as well.

5 Conclusions

In this paper, we have shown that the basic ant-routing algorithm does not perform well for dense, asymmetric, and dynamic networks. We then developed three new ant-routing algorithms for sensor networks. The new ant algorithms work quite well overall, with each having some advantages in terms of performance: SC is energy efficient, FF has shorter delays, and FP has the highest success rates. In the near future, we will compare ant routing with other reinforcement learning based algorithms and with traditional ad-hoc routing algorithms such as AODV [8].

6 Acknowledgments

This work is funded in part by Defense Advanced Research Project Agency contract # F33615-01-C-1904.

References

[1] G. D. Caro and M. Dorigo. Antnet: Distributed stigmergetic control for communication networks. *Journal of Artificial Research*, 9:317 – 365, 1998.

[2] A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Proc. IEEE InfoComm*, New York, NY, June 2002.

[3] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *Int. Journal on High Performance Computing Applications*, June 2002.

[4] J. Chen D. Subramanian, P. Druschel. Ants and reinforcement learning: A case study in routing in dynamic networks. Technical report tr96-259, Rice University, July 1998.

[5] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):1–13, 1996.

[6] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scale coordinations in sensor networks. In *Proc. ACM MobiComm*, pages 263–270, Seattle, WA, August 17-19 1999.

[7] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. 6th Int'l Conf. on Mobile Computing and Networks (ACM Mobicom)*, Boston, MA, 2000.

[8] C. E. Perkins and E. M Royer. Ad hoc on-demand distance vector routing. In *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[9] E. Royer and C. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, April 1999.

[10] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169 – 207, 1997.

[11] G. Simon. Probabilistic wireless network simulator. http://www.isis.vanderbilt.edu/projects/nest/prowler/.

[12] Y. Zhang and M. Fromherz. Message-initiated constraint-based routing for wireless ad-hoc sensor networks. In *Proc. IEEE Consumer Communication and Networking Conference*, 2004.