Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks

Devika Subramanian Peter Druschel Johnny Chen
Department of Computer Science
Rice University
Houston, Texas 77005

Abstract

We investigate two new distributed routing algorithms for data networks based on simple biological "ants" that explore the network and rapidly learn good routes, using a novel variation of reinforcement learning. These two algorithms are fully adaptive to topology changes and changes in link costs in the network, and have space and computational overheads that are competitive with traditional packet routing algorithms: although they can generate more routing traffic when the rate of failures in a network is low, they perform much better under higher failure rates. Both algorithms are more resilient than traditional algorithms, in the sense that random corruption of routing state has limited impact on the computation of paths. We present convergence theorems for both of our algorithms drawing on the theory of non-stationary and stationary discrete-time Markov chains over the reals. We present an extensive empirical evaluation of our algorithms on a simulator that is widely used in the computer networks community for validating and testing protocols. We present comparative results on data delivery performance, aggregate routing traffic (algorithm overhead), as well as the degree of resilience for our new algorithms and two traditional routing algorithms in current use. We also show that the performance of our algorithms scale well with increase in network size using a realistic topology.

Efficiently routing data in a dynamic network is a difficult and important problem in computer networking. By dynamic we mean a network subject to frequent and unpredictable changes in topology (e.g, due to link failure) and link costs (e.g., due to congestion). One of the most widely used solutions for routing on such networks [2] places substantial space and time requirements on all the routers to guarantee effective performance, thus limiting its scalability.

In this paper, we investigate two algorithms for this problem inspired by the dynamics of how ant colonies learn the shortest routes to food sources, using very little state and computation [1]. The first algorithm, which we call the *regular ant algorithm*, is based on earlier work by Holland et. al. [5] for call routing in telephone networks. The regular ant algorithm is a single shortest path algorithm and is only applicable to networks with symmetric path costs. We have developed a second algorithm, called the *uniform ant algorithm*, which is a natural multi-path routing algorithm that is applicable to data networks with or without path cost symmetry. We provide a comprehensive theoretical analysis of both ant algorithms and provide extensive empirical evaluation of the algorithms on a low-level network simulator.

A major reliability issue in data networks is the *resilience* of the routing algorithms to corruption of router state. Current routing algorithms are more susceptible to such failures since they maintain more network state at each router. We experimentally demonstrate that the ant-based algorithms are far more resilient. Even though the routing traffic generated by our ant algorithms exceeds that of existing algorithms, ants can be readily piggybacked onto data packets on a hop-by-hop basis, since they are small and are of fixed size. This piggybacking helps to significantly defray the routing overhead. Finally, the routing overhead of ant algorithms is independent of the rate of change (link/recoveries and failures and link cost changes) in the network, making these algorithms very attractive for highly dynamic networks.

The paper is organized as follows. In Section 2, we describe the routing problem and present two current solutions to it. In Section 3, we describe the two ant algorithms, the parameters that characterize them, and present their theoretical convergence properties. In Section 4, we present an detailed empirical evaluation of the ant algorithms. We provide comparisons to the two algorithms currently in use on an example network as well as more realistic topologies. We compare speed of convergence both during initial route discovery and in response to topology changes (link failures and subsequent recovery) for all the algorithms. We present data on average delays in delivery of data packets and the aggregate routing traffic (routing algorithm overhead). We then show the results of an experiment in which one of the routers in a network enters a corrupted state and we study the impact on end-to-end delivery delay on all four algorithms. We demonstrate how the algorithms scale with size of the network by measuring routing traffic they generate. We conclude with a discussion of related work, and open problems in the design of ant-based algorithms for routing on data communication networks.

1 The Routing Problem

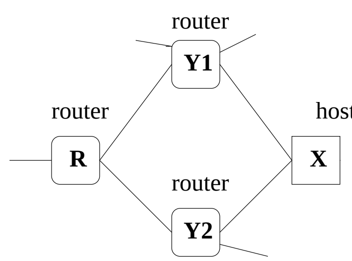
The challenge of routing in packet-switched communication network is the need for a fully distributed algorithm that is able, without central coordination to disseminate knowledge about the network, to find shortest paths robustly and efficiently in the face of changing network topologies as well as changing link costs. There are two major classes of adaptive, distributed packet routing algorithms in the literature: distance-vector algorithms and link-state algorithms. Distance-vector algorithms are dynamic programming algorithms which only propagate cost information. No node, at any time, has complete knowledge of the topology of the whole network. Link state algorithms require all nodes to know the topology of the network before computing shortest paths. Link state methods have each node periodically broadcast its local topology and costs to the entire network. The Internet Open Shortest First (OSPF) Protocol [2] uses a link state algorithm.

2 Ant Routing Algorithms

To facilitate the description of our algorithms we distinguish between two types of nodes on the network: hosts and routers. Hosts are communication end-points, that is, they can initiate and terminate data messages[1]. Routers are nodes which forward data messages and can send and receive routing messages. There are three key conceptual ideas underlying our two ant algorithms. *Ants that explore the network*: The hosts take a very active role in gathering information about the costs of various paths in the network. Periodically, each host h_d in the network generates a message to another randomly chosen host h_s. This message is of the form (h_d, h_s, c) where c is a cost that starts out at zero and is the cost to get to h_d. The message is sent out on the network where it is forwarded to h_s by routers it encounters, which also increment the cost c to reflect the

[1] In general, a host represents a network destination for which the ant algorithms compute routes; a destination could be a single computer attached to the network, a subnetwork within an internetwork, or a routing domain within a hierarchical routing architecture.

cost in reverse[2] of links it has traveled on thus far. When the message reaches its destination h_s, the cost c is the end-to-end cost of sending a message from h_d to h_s. The message[3] is destroyed at host h_s. These messages are small (size O(1), 6 to 10 bytes), and we refer to them as *ants*. Ants explore paths backward from destination nodes in the network to source nodes. How ants are routed through the network, and the rate at which they are generated (the generation rate is the same for all hosts in the network) are key parameters in the definition of the ant routing algorithms.
Probabilistic routing tables: Unlike link state and distance-vector algorithms which compute deterministic forwarding tables for each router, our forwarding tables are probabilistic. The router r maintains for each destination node x in the network, an entry of the form $(x, (y_1, p_1), (y_2, p_2), \ldots, (y_n, p_n))$, where $\forall i, (r, y_i)$ is a point-to-point link and $\sum_{i=1}^{n} p_i = 1$. When r receives a message destined for host x it forwards it to its neighbor y_i with probability p_i. The probabilistic tables are a mechanism for exploring alternate paths in the network and keeping estimates of their lengths relative to the current best paths.



dest	Y1	Y2
X	0.2	0.8

Figure 1: R sends messages destined for X to Y1 20% of the time, and to Y2 80% of the time.

Probabilistic updates of routing tables by ants: An ant (h_d, h_s, c) generated by host h_d to host h_s probabilistically updates the routing tables of all the routers it encounters along its path. The update rules are similar in spirit to those in traditional reinforcement learning algorithms [3]; the key technical difference is that the update rules are non-linear. If router r receives the ant (h_d, h_s, c) on link l_i from node y_i, it updates c by adding the cost of traversing l_i in reverse, and then updates its entry for h_d which is $(h_d, (y_1, p_1), \ldots, (y_n, p_n))$ as follows:

$$p_i = \frac{p_i + \Delta p}{1 + \Delta p}, p_j = \frac{p_j}{1 + \Delta p}, 1 \le j \le n, i \ne j$$

where $\Delta p = \frac{k}{f(c)}, k > 0$, and $f(c)$ is a non-decreasing function of c.

These update rules are drawn from [5]. The constant k is called the *learning rate* of the algorithm. It is generally less than 0.1. The learning rate needs to be set high enough that each ant has some effect on p_i, and low enough so we can guarantee convergence of the routing probabilities. Note that the router's probabilities to the host which generated the ant are updated. Ants perform a form of backward learning.

Unlike traditional adaptive algorithms in this context, including [4, 3] the extent of the reinforcement is not the direct cost c of the ant, rather it is a decreasing non-linear function[4] of

[2] When a message traverses a link from node a to node b, c is incremented by the cost of the link from b to a.

[3] In fact, we do not require that the message get to h_s since h_s could be an inaccessible host. Any other host in the network can absorb the message, and the message is destroyed once c increases beyond a pre-set threshold.

[4] We normalize actual path costs, so that the Δp's are not affected by linear scaling of link costs in the network.

c.

A router r forwards the ant (h_d, h_s, c) after updating its table as discussed above. It uses the routing probabilities for ant forwarding as follows. Let the routing entry for destination h_s be $(h_s, (y_1, p_1), \ldots, (y_n, p_n))$. The *regular ant* algorithm uses $Pr(\text{ant sent to } y_i) = p_i$ and the *uniform ant* algorithm uses $Pr(\text{ant sent to } y_i) = \frac{1}{n}$.

Regular ants use the learned forwarding tables to route ants. If a good, non-congested route is found, new ants will be more likely to be forwarded on that route. Exploration of less desirable routes is curtailed by this policy, so eventually the ants converge to a single shortest route. Regular ants require path costs in the network to be symmetric: i.e., the cost of getting from node a to node b has to be the same as the cost of getting from node b to node a.

Uniform ants are unbiased by the forwarding probabilities and explore all paths with equal probability. Uniform ants are natural multi-path routers, they increase available bandwidth and are not prone to route oscillation problems that single shortest path routing algorithms have. Uniform ants have a much simpler structure than regular ants. A uniform ant is of the form (h_d, c) where h_d is the host that generated it and c, initialized to zero at the source, is the cost to get to h_d on the links it has traveled on so far. Uniform ants have a time to live, after which they are terminated. Uniform ants do not require a destination, a fact of great practical importance because not every host on the network knows about all the other hosts.

2.1 A simple example and its analysis

We present the example in Figure 2 to illustrate the ant algorithms in more detail. We will, in the context of this example, also demonstrate convergence properties of the update rules for both regular and uniform ants.

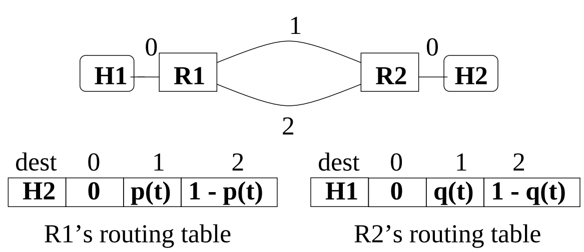


Figure 2: A simple network with two hosts and two routers. The routing tables for the two routers are also shown. Link 1 has cost C_1 and Link 2 has cost C_2.

2.1.1 Regular Ants

For regular ants, the probabilities $p(t)$ and $q(t)$ evolve simultaneously according to the coupled stochastic discrete difference equations shown below when $C_1 = C_2$.

$$p(t+1) = \begin{cases} \frac{p(t)+\Delta p_1}{1+\Delta p_1} & \text{with probability } q(t) \\ \frac{p(t)}{1+\Delta p_2} & \text{with probability } 1 - q(t) \end{cases}$$

$$q(t+1) = \begin{cases} \frac{q(t)+\Delta p_2}{1+\Delta p_2} & \text{with probability } p(t) \\ \frac{q(t)}{1+\Delta p_1} & \text{with probability } 1 - p(t) \end{cases}$$

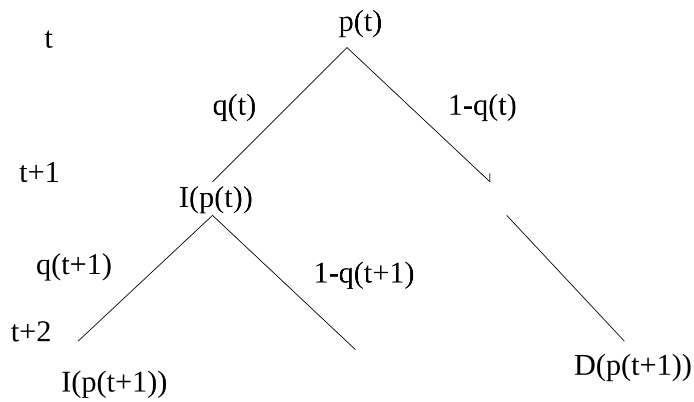


Figure 3: Updates to $p(t)$ and $q(t)$ for the network in Figure 1 for asymmetric costs. Cost ratio between paths is 2.

When the costs are different, the updates occur as shown in the computation tree in Figure 2.1.1. At time t with probability $q(t)$, $p(t)$ is bumped up; however because ants on the slower route take longer to reach destination 2, the lowering of $p(t)$ occurs only at time $t+2$. So $p(t)$ is bumped up twice (at $t+1$ and $t+2$) for each decrementation (at $t+2$), when the cost ratio is 2.

where $\Delta p_1 = \frac{k}{f(C_1)}$ and $\Delta p_2 = \frac{k}{f(C_2)}$.

Proposition 1 *For regular ants, if $C_1 < C_2$ in the network in Figure 2,*

$$lim_{t\to\infty} p(t) = lim_{t\to\infty} q(t) = 1$$

The system of equations above form a non-stationary discrete-time Markov chain on the reals. We first observe that $p(t)$ and $q(t)$ co-evolve; and that the first part of the update rule (I_1) increases $p(t)$, and the second part (D_1) decreases $p(t)$. If $p(t)$ were updated by an infinite sequence of I_1's, $p(t)$ will tend to 1, and $p(t)$ will tend to 0 if it were updated by an infinite sequence of D_1's. Since the result of updating $p(t)$ by the sequence $I_1 D_1$ yields a different result if we update with the sequence $D_1 I_1$, the updates are order dependent. Thus, when $\Delta p_1 = \Delta p_2$, the final values of $p(t)$ and $q(t)$ are determined entirely by the transient behaviour of the system, and the system exhibits deterministic chaos. What this means is that traffic will be divided between the two equal cost links in some arbitrary ratio. If on the other hand, we have $\Delta p_1 > \Delta p_2$ (which arises when $C_1 < C_2$), we have an asymmetric system that converges in the limit to the values provided in the statement of this proposition. The reason for the asymmetry is that $I_1(x) > I_2(x)$ and $D_1(x) < D_2(x)$ for $x \in [0,1]$, where I_2 and D_2 are functions for updating $q(t)$, and I_1 and D_1 are functions for updating $p(t)$. Note that $I_1(1) = I_2(1) = 1$ and $D_1(0) = D_2(0) = 0$. If this system converges, so that $p(t)$ and $q(t)$ achieve their fixed point values $\hat{p}$ and $\hat{q}$, then

$$\hat{p} = \hat{q} I_1(\hat{p}) + (1-\hat{q}) D_1(\hat{p})$$

$$\hat{q} = \hat{p} I_2(\hat{q}) + (1-\hat{p}) D_2(\hat{q})$$

This can be achieved only if $\hat{p} = \hat{q}$ and they take on extreme values of 0 or 1. To eliminate the convergence to 0, we need to understand the dynamics of the network when the costs are different. Following Figure 2.1.1, we can see that the sequence of updates to $p(t)$ has (twice) more I_1's than D_1's, and so asymptotically $p(t)$ is driven to 1.

This proposition can be generalized to any symmetric network to show that regular ants will converge to the shortest path, as long as there is a unique shortest path in the network.

2.1.2 Uniform Ants

For uniform ants, the probabilities $p(t)$ and $q(t)$ evolve simultaneously according to the stochastic discrete difference equations shown below. Note however that the equations for $p(t)$ and $q(t)$ are no longer coupled.

$$p(t+1) = \begin{cases} \frac{p(t)+\Delta p_1}{1+\Delta p_1} & \text{with probability } 0.5 \\ \frac{p(t)}{1+\Delta p_2} & \text{with probability } 0.5 \end{cases}$$

$$q(t+1) = \begin{cases} \frac{q(t)+\Delta p_2}{1+\Delta p_2} & \text{with probability } 0.5 \\ \frac{q(t)}{1+\Delta p_1} & \text{with probability } 0.5 \end{cases}$$

This allows us to establish a convergence theorem much more readily, because $p(t)$ and $q(t)$ are stationary Markov chains over the reals.

Proposition 2 *For uniform ants, in the network of Figure 2,*

$$lim_{t\to\infty} p(t) = \frac{C_2}{C_1+C_2}, lim_{t\to\infty} q(t) = \frac{C_1}{C_1+C_2}$$

provided Δp_1 and Δp_2 are positive constants close to 0, and $f(c) = c$.

Proof:

We note that the expected value $Ep(t)$ of $p(t)$ evolves according to the following recurrence if the $\Delta p_i \to 0$:

$$Ep(t+1) = \frac{1}{2}\left[\frac{Ep(t)+\Delta p_1}{1+\Delta p_1} + \frac{Ep(t)}{1+\Delta p_2}\right]$$

We can solve this recurrence to obtain:

$$Ep(n) = p(0)(\frac{1}{2})^n \left[\frac{1}{(1+\Delta p_1)} + \frac{1}{(1+\Delta p_2)}\right]^n + \left[\frac{1}{2}\frac{\Delta p_1}{(1+\Delta p_1)} \sum_{i=0}^{n-1} \left(\frac{1}{2}\right)^i \left[\frac{1}{(1+\Delta p_1)} + \frac{1}{(1+\Delta p_2)}\right]^i\right]$$

As $n \to \infty$, the $p(0)$ term goes to zero. The second term is an infinite series that sums to the quantity shown below. Therefore,

$$lim_{t\to\infty} Ep(t) = \frac{1}{1+\frac{\Delta p_2(1+\Delta p_1)}{\Delta p_1(1+\Delta p_2)}}$$

Note that if $\Delta p_1 = \frac{k}{C_1}$ and $\Delta p_2 = \frac{k}{C_2}$, where the positive constant $k << C_1$; then

$$lim_{t\to\infty} Ep(t) = \frac{C_2+k}{C_1+C_2+2k} = \frac{C_1}{C_1+C_2}$$

This result can be generalized for the case when there are two loop-free, link disjoint paths between any two nodes in a general network. The probabilities p and q on these nodes evolve as shown above with the probability 0.5 replaced by a, where $0 < a < 1$. a is determined by network topology and can be interpreted as the ratio of the fraction of ants arriving from one node to the other along the two paths. The limiting values of the probabilities are $\frac{aC_2}{aC_2+(1-a)C_1}$ and $\frac{(1-a)C_1}{aC_2+(1-a)C_1}$, where C_1 and C_2 are the end-to-end costs along the two paths. Note that when $a = 0.5$, uniform

Property	LS	DV	Ants
Space	$O(nrl) + O(rh)$	$O(rh)$	$O(rh)$
Time	$O(rnlogn)$	$O(ld)$	$O(ldh)$
# of msgs	$O(rl)$	$O(ld)$	$O(ldh)$
Size of msg	O(e)	$O(h)$	O(1)

Table 1: A comparison of resource requirements for the entire network. n is the sum of the number h of hosts in the network and the number r of routers in the network. l is the number of point-to-point links in the network. d is the diameter of the network and e is the average number of links per router.

ants divide traffic along the two paths in inverse ratio of the path costs; a highly desirable traffic split that can optimize available bandwidth.

Both Proposition 1 and Proposition 2 are statements of asymptotic convergence; the rates[5] of convergence (which depend on the ant generation rate as well as the topology of the network) are determined empirically in Section 4.

2.2 Improving Adaptiveness

Once the regular ants reach steady state, the forwarding probabilities converge to 0's and 1's as shown in Proposition 1. If there are topology changes or link cost changes after convergence, the regular ant policy of forwarding ants according to the learned probabilities will prevent them from adapting to the new situation. Uniform ants are immune to this problem because all paths in the network are explored at all times with equal probability. So changes in the network will be detected and the probabilities will adapt at a pace dependent on the learning rate k.

We use the solution in [5] to allow regular ants to perform a certain degree of random exploration. A certain percentage of the time ($f\%$), a router forwards ants uniformly randomly (i.e., using the uniform ant routing policy), and the rest of the time ($1 - f\%$) the router forwards ants according to the forwarding probability tables (i.e., using the regular ant routing policy). This noise *only* affects ant forwarding and not the data packets. With $f > 0$, the probabilities of a single path never reach 0 or 1. This guarantees that if a primary route becomes no longer available, the update rules reinforce the second best paths fairly quickly. Only the ants traveling on the second best path get to reinforce the forwarding probabilities to the source and there are no ants to reinforce the former best path. We summarize this phenomenon by the slogan: bad news travels fast. However, if a better path suddenly becomes available, the only ants available to reinforce it are the ones routed through it by the noise; and so the system takes longer to switch to the better path. We characterize this as: good news travels slower than bad news. To make good news travel faster, we can crank up the noise levels or increase the learning rate.

Table 1 summarizes the key resource requirements of the ant family of algorithms, the distance-vector and the link state algorithms. This table makes it clear that the link state algorithm pays a significant overhead in terms of state on each router and in terms of time to compute shortest paths. This investment in state in the routers allows it to reduce the number of routing messages sent during the computation of initial routes. The distance-vector and the ant algorithms have significantly lower router state, but this comes at the expense of increased routing message traffic. While the distance-vector and ants use roughly the same order of bytes of routing traffic, the ant algorithms use a factor of h (where h is the number of hosts) more messages than the distance-

[5]It is very difficult to obtain general results on convergence rates for these systems of equations.

vector method. It is important to note that ant messages, unlike the distance-vector messages, are of constant size, and can be piggybacked on to data packets. On high bandwidth networks this virtually gives us ant routing traffic for free.

3 Experimental Results

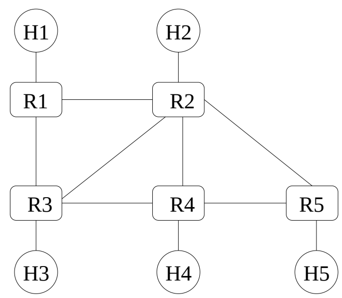


Figure 4: Simple Network Topology

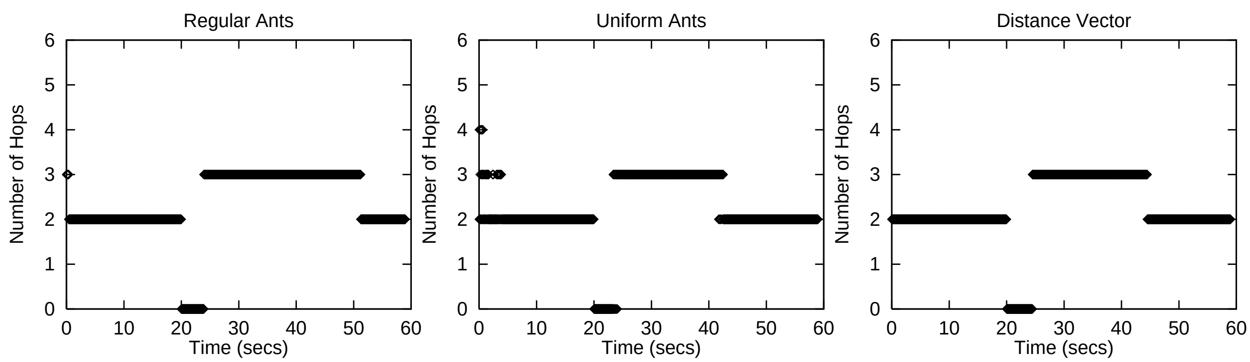


Figure 5: End-to-end delay plots for regular ants, uniform ants, and the distance vector algorithm for the simple network topology in Figure 2. Host H1 is the sender and host H5 is the receiver. The delay plot for the link state protocol is identical to the one for the distance vector algorithm.

In this section, we present the results of extensive simulations that were performed to evaluate the proposed algorithms, and to compare their behavior with that of the distance-vector (DV) and link-state (LS) routing algorithms. The algorithms are evaluated on networks with simple topology, and more realistic topologies derived from a subset of the Internet inter-domain topology.

3.1 Simple Topology

The first, simple network used in our simulations is depicted in Figure 4. In this network, all links are of equal, unit cost. Thus, for instance, the shortest path from R1 to R5 has cost two[6]. Simulations were run for 60 seconds of simulated time. This time was chosen such that each routing protocol can reach steady state conditions between successive changes in the state of the network. Each host generates ants at a rate of $8/second$. At time $t = 0s$, the network is started with initial

[6] The link between a host and its associated router is not included in any path cost calculations. This is purely a matter of implementation convenience. It has no effect on the routing algorithm.

Unit	UA	RA	DV	LS
Msgs per link per second	18.17	9.22	2.20	2.05
Bytes per link per second	109	92	11	9

Table 2: Routing traffic per link per second for Uniform Ant (UA), Regular Ant (RA), Distance Vector (DV), and Link State (LS) algorithms.

conditions. The link connecting R2 and R5 fails at time $t = 20s$, and it recovers at time $t = 40s$. The learning rate is .08 and the noise level is 10%.

Our first experiment quantifies end-to-end delivery latency, which is an important aspect of a routing algorithm's performance as perceived by a network application. Figure 3 shows the end-to-end delivery delay, in number of network hops, measured during the simulation described above. Host H1 periodically sends messages at a rate of $670 bytes/second$ to Host H5. The figures show the results for regular ants, uniform ants, and the distance-vector algorithm respectively. We omit the link-state plot because it is identical to the distance-vector plot. A reported delay of zero hops means that the packet was lost, i.e., it was never delivered to its destination.

To a first approximation, all algorithms achieve the minimal delivery delay of 2 hops during the interval between $t = 20s$ and $t = 40s$, and a delay of 3 hops otherwise. Uniform ants split traffic among multiple paths. This results in some jitter (i.e., variance in delay) when traffic is split over paths with different costs, as can be seen in the graph up to $t = 4s$ and between $t = 40$ and $t = 42s$. Traffic is also split between $t = 30s$ and $t = 40s$, but here the traffic is split over equal cost path, with no impact on jitter. Splitting traffic in this way also leads to an increase in available bandwidth, but this is not evident from the delay plot.

In response to the failure of the shortest path link at $t = 20s$, packets are lost for a period of three to five seconds, depending on the algorithm used. With the ant algorithms, this time is determined by the transition time of the forwarding probabilities, caused by the reinforcement of alternate paths.

Table 2 shows the amount of routing traffic generated by each of the algorithms in our simulation. Whenever a routing message traverses one link of the network, one message transmission is recorded, and the size of the message is added to the number of bytes of routing traffic.

In this experiment, the amount of routing traffic generated by the ant algorithms exceeds that of LS and DV by roughly a factor of 10, both in terms of messages and data. Note, however, that all ant routing messages are small and of fixed size. Therefore, these messages can be piggy-backed onto data packets, substantially defraying their cost to the network. The messages generated by the LS and DV algorithms are larger and not of fixed size; therefore, they do not lend themselves easily to piggy-backing. In the subsequent section, we will present data that quantifies the dependence of routing traffic on network size and rate of change in the link costs.

3.1.1 Resilience to State Corruption

The next set of experiments attempt to quantify the behavior of our routing algorithms in the event of failures. Specifically, we are interested in failure modes where a router's internal state is corrupted. This case is of great practical importance. Router state corruptions can occur in practice due to intermittent software and hardware faults, and configuration errors. Unlike communication errors affecting routing messages, such errors are difficult to detect and/or correct. At $t = 30s$ in the simulation, router R2 suffers an intermittent fault that causes its routing state to be overwritten with random values. Since we are interested in evaluating the robustness of the various algorithms,

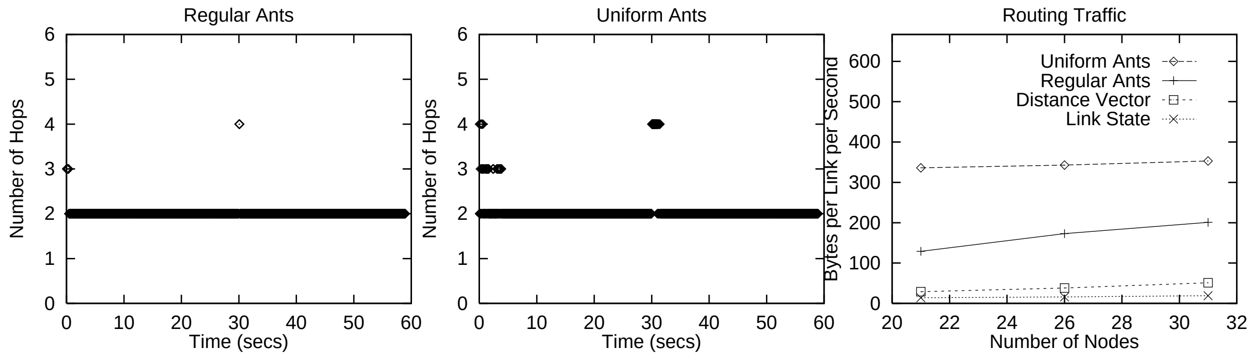


Figure 6: The left two plots show the end-to-end delay, from H1 to H5 in the network of Figure 2 for regular and uniform ants *with router corruption*. Router R2's probability tables are randomly overwritten at time of 30s. The plot on the right side shows the routing traffic for the 21, 28, and 31 node star-like topology.

rather than that of their implementation, the routing state is corrupted with random but plausible values.

With LS, the recovery period is bounded by the periodic link-state broadcast interval. In practice, the broadcast interval tends to be quite long, for instance 30 minutes in the OSPF protocol. During the recovery period, the router is unable to calculate correct routes, which may result in routing loops and the associated loss of connectivity. With DV, once the router has learned its actual local topology, it will eventually converge to the actual shortest paths after a number of message exchanges that is proportional to the diameter of the network, and the range of the path cost metric[7]. In practice, the process can be expected to take time on the order of seconds. During the recovery period, routing loops may exist and cause loss of connectivity.

Figure 6 shows the end-to-end packet delivery delay from H1 to H5. It is evident that both uniform and regular ants perform extremely well in the event of a router corruption. With uniform ants, only a small number of packets suffer an increased delay of four hops for approximately 2 seconds following the router failure. No packets are lost with either algorithm. In comparison, with LS and DV protocols, one expects loss of connectivity (packet loss) after the router failure for a period of time ranging from seconds to minutes.

3.2 Scalable Topology

The second topology we use for our experimental evaluation more closely reflects the structure of realistic data networks. The topology is based loosely on the Internet inter-domain routing topology. The network is roughly a partial 6-ary tree where one of the six leaf nodes of each parent is connected to one leaf node of the adjacent parent. We use graphs with 3, 4, and 5 levels with a total of 21, 26, and 31 nodes. For each size of the network, we measure the routing traffic generated by each of the algorithms necessary to achieve a given speed of convergence of 5 seconds.

Figure 6 shows the number of routing messages generated by each algorithm for networks with 21, 26, and 31 nodes, during a 60 second simulation with parameters equal to those used in the earlier simulations. Thus, for realistic wide-area network topologies that are characterized by low degrees of connectivity, all these algorithms scale well.

In our final simulation, we measure the generated routing traffic as a function of the rate of

[7]A count-to-infinity may occur in a loop of routers involving the failed router.

topological change for each of the algorithms. Topological change is modeled here as the number of links that fail and recover over the simulation period of 60 seconds. Figure 5 shows the amount of routing traffic generated by each algorithm in the 5 level network, for various rates of link failures and recoveries. As predicted, the routing traffic generated by the ant algorithms remains nearly unaffected by the amount of topological change in the network. With link-state and DV, the routing traffic increases more than linearly with the amount of change. This demonstrates one of the key advantages of ant algorithms over LS and DV.

4 Related Work

4.1 Q-Routing and Ants

In Q-Routing [4], each node keeps Q-values of the form $Q_x(d, y)$, representing node x's cost estimate to d via neighbor y. Whenever x forwards data to d, x consults its Q table and forwards to the neighbor with the least cost estimates to d. On receiving a packet from x, neighboring node y immediately sends its least cost estimate to d back to x. x then adjusts its cost estimate $Q_x(d, y)$ based on y's message.

Q-Routing uses a dynamic, i.e., traffic-dependent, routing metric (delay). Therefore, it can adapt to changes in topology as well as traffic conditions. Ants, as well as LS and DV, can be made to do the same simply by choosing an appropriate link cost metric[8].

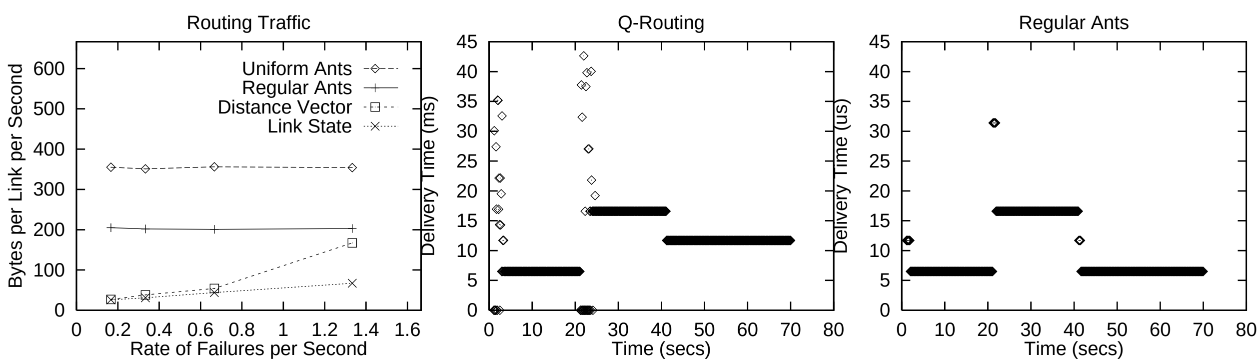


Figure 7: The plot on the left side shows routing traffic as a function of link failures for the 31 node topology. The two plots on the right show an end-to-end delay comparison between Regular ants and Q-Routing. The network load increased to 1.5 at time 20s and decreased to 0 at time 40s.

Q-Routing differs from regular ants in two key aspects. First, unlike ants, Q-routing is not guaranteed to find the shortest path. Second, the amount of routing traffic with Q routing is directly proportional to the amount of data traffic in the network, while routing traffic is constant with ants. Uniform ants have the additional capability of routing packets along multiple paths.

Q routing can discover changes in route cost (delay in this case) only on the *currently used* route. Since the currently used route is the best known route, only an *increase* in the cost of the currently used route can cause Q routing to select a different route. A *decrease* in the cost of a route other than the currently used route cannot be discovered, causing Q routing to potentially use a suboptimal route. The *full echo* variant of Q routing overcomes this deficiency for the special case where a better route other than the currently used route is known to a neighbor of the message source. Full

[8]For instance, to implement a delay-based metric, neighboring routers can periodically exchange ping-pong messages, and assign the measured delay as the cost of the associated link.

echo comes at the cost of increased routing traffic, but can still not guarantee convergence to the shortest path. The regular ant algorithm can be shown to always converge to the single shortest path.

To illustrate this point, consider Figure 7, which shows the packet delivery delay for Q-Routing and regular ants in the 6x6 irregular network used in [4]. For both algorithms, the routing cost metric is end-to-end delivery delay. At $t = 20s$, the background network load increases, causing a change in the optimal path between source and destination. At $t = 40s$, the network is restored to its original load. It is evident that at $t = 20s$, Q-Routing adapts to increasing load and discovers the new optimal path, but fail to "rediscover" the original path when the background traffic subsides at $t = 40s$. As a result, the delivery delay never returns to its value before $t = 20s$. The ant algorithms, on the other hand, is able to converge to the optimal path in both cases. Data packets that are forwarded more than 30 times are considered undeliverable and are shown as having a delivery time of zero. As evident from the graph, a substantial number of packets are not delivered with Q routing during the transition phases following $t = 0s$ and $t = 20s$.

With ant algorithms, the routing traffic is independent of the data traffic. The rate of ant generation determines a fixed tradeoff between amount of routing traffic and rate of convergence. In Q-Routing, the number of routing messages is directly proportional to the number of transmitted data messages. This means that at low network load, Q-routing is slow to adapt to network changes. At high network load, Q-Routing overhead increases, thus limiting the aggregate capacity of the network.

4.2 Link-state and Distance-vector Algorithms

Ant routing algorithms differ from LS and DV algorithms in various aspects. Raw ant routing traffic exceeds that of LS and DV by a small factor, but this must be seen in light of the following facts. First, the absolute amount of routing traffic for a 31 node network is still modest (201 Bytes/sec and 20 msgs/sec of traffic are generated per link). The small, fixed sized (6 bytes for uniform, 10 bytes for regular) ant messages can be piggy-backed onto data packets on a hop-by-hop basis, which largely defrays their cost to the network. Unlike LS and DV, ant routing traffic does not increase with the rate of change in the network, making these algorithms suitable for highly dynamic networks. Ant algorithms maintain minimal state in the routers, which meets the needs of networks with scarce router resources. Thus they are particularly suitable, for instance, in mobile, wireless networks used for personal communication networks where topology and load changes frequently, and where resources in the routers are scarce. Finally, the ant algorithms have additional capabilities (multi-path and improved resilience to router state corruption) not present in LS and DV.

References

[1] R. Beckers, J. L. Deneuborg, and S. Goss. Trails and U turns in the selection of a path by the Ant *lasius niger*. *Journal of Theoretical Biology*, 159:397–415, 1992.

[2] R. Coltun. OSPF: an Internet routing protocol. *ConneXions*, 3(8):19–25, 1989.

[3] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of AI Research*, 4:237–285, 1996.

[4] M. Littman and J. Boyan. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Proceedings of NIPS-94*, 1994.

[5] R. Schooenderwoerd, O. Holland, J. Bruten, and L. Rosenkrantz. Ants for load balancing in telecommunication networks. Technical Report HPL-96-35, HP Labs, Bristol, 1996.