Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

The Markovian Termite: A Soft Routing Framework

Martin Roth
Deutsche Telekom Laboratories
Martin.Roth02@telekom.de

Abstract— **An analytical framework is presented to study the self-adaptive behavior of probabilistic routing protocols for computer networks. Such soft routing protocols have attracted attention for delivering packets more reliably, robustly, and efficiently than conventional deterministic approaches. Efficient global operating parameters can be estimated without resorting to expensive Monte-Carlo simulation of the whole system. Key model parameters are routing sensitivity and routing threshold/noise, which control the "randomness" of packet routes between source and destination, and a metric estimator. Global network characteristics are estimated, including steady state routing probabilities, average path length, and path robustness.**

The framework is based on a Markov chain analysis. Individual network nodes are represented as states. Standard techniques are used to find primary statistics of the steady state global routing pattern, given a set of link costs. The use of packets to collect information about, or "sample," the network for new path information is also reviewed. How the network sample rate influences performance is investigated.

I. Introduction

A. Overview

Adaptive behavior is one of the fundamental requirements of modern network routing protocols [1]. Deterministic approaches are able to perform well in relatively static networks by relying on traditional link state or distance vector shortest path algorithms. Difficulty arises under dynamic conditions when route costs are highly variable or the topology itself is unstable. Multipath routing is often cited as a solution, however its implementation is complicated by the need to explicitly manage additional routes; multipath routing does not fit elegantly into a traditional routing framework. Such a scenario is especially relevant in wireless mesh and ad-hoc networks, where the communications characteristics of wireless links are subject to large and frequent variations. Probabilistic routing (here also referred to as soft routing or p.routing) is able to maintain route utility estimates on all routes simultaneously in one unified framework. Good paths are used proportionally more than bad ones, and all paths are successively refined based on simple local interactions between hosts. Adaptive end-to-end routing is an emergent behavior based on the interactions of each node in the network. Soft routing is ultimately able to outperform deterministic routing (see Section I-B for details).

Probabilistic routing algorithms contain many parameters that influence their performance. Though there are several possible formulations of p.routing, each approach generally has three primary parameters. These are a sensitivity parameter, which controls the bias towards well performing paths, a noise parameter, which sets a lower bound on the "randomness" of paths, and an estimation parameter, which determines how different measurements of the network are combined to determine a single estimate of its state. Related parameters include how often metric estimates are updated and how fast the network itself changes.

These parameters define interactions at a local level, at each network node, but it is unclear how they affect global behavior such as routing performance. System properties of interest might include the expected per-packet path cost between a source and destination, or a measure of the robustness of the equilibrium routing solution. Extensive simulation can be used to evaluate these characteristics, but a model which can directly determine (possibly optimal) local parameter values is missing. The framework presented here is the first step towards such a model.

The current work finds only the equilibrium routing distribution to a network with static costs, though the network correlation time is taken into account. Nothing is said of the convergence properties of the modeled p.routing algorithms.

B. Previous Work

Little work has been done to directly model the performance of soft routing algorithms. The vast majority of the effort in the area has been simulation based. A variety of protocols has been proposed including Q-Routing [2], Ant-Based Control (ABC) [3], AntNet [4], Cooperative Asymmetric Forwarding (CAF) [5], Probabilistic Emergent Routing Algorithm (PERA) [6], Ant-based Routing Algorithm (ARA) [7], Mobile Ant-Based Routing (MABR) [8], Multiple Ant Colony Optimization (MACO) [9], Termite [10], Ad-hoc Networking with Swarm Intelligence (ANSI) [11], AntHocNet [12], BeeAdHoc [13], and SAMPLE [14]. These protocols span the application space between wired and wireless mesh or ad-hoc networks, usually being compared to a well known deterministic protocol such as Open Shortest Path First routing (OSPF) [15] for wired networks or Ad-hoc On-demand Distance Vector routing (AODV) [16] for ad-hoc networks.

Strictly analytical work is scant, and focuses on characterizing global behavior. [17] considers two different packet forwarding equations and shows that one type will end up using only the better of two links, regardless of their relative difference. The other method will split traffic across the two links proportionally to their utility. [18] develops a model for a node's estimate of the path utility to a destination given a metric update equation and its parameters. The model is used to explain differences in global routing performance when different metric updates are used, including a normalized exponential filter (also known as the pheromone update rule), an exponential filter, and a Dijkstra-inspired update.

An independent parameter is discovered which determines the maximum link estimate. [10] develops a heuristic for a good selection of the estimation parameter in the Termite p.routing algorithm based on the network rate-of-change; the correlation time. The results in this paper continue the analytical characterization effort by accounting for all fundamental aspects of soft routing algorithms. This paper is a direct extension of [19] with updated mathematics, figures, additional results and discussion, and an appendix of mathematical derivations.

The framework presented in this paper is the first effort of its kind to model the performance of an entire probabilistic routing algorithm, rather than just specific aspects. For this reason, no direct comparison to other frameworks is possible.

C. Structure of Paper

Section II gives an overview of how a generic probabilistic routing protocol works and introduces the p.routing framework in this light. The Termite protocol and an ACO based approach are used as examples. Section III develops a Markov chain analysis of the framework, which is used to reveal various aspects of the performance of soft routing algorithms. Section IV gives numerical results based on the model, followed by analysis and discussion. Section V concludes the paper, in addition to providing avenues for future work. An appendix gives a more thorough mathematical treatment to some aspects of the framework.

II. Probabilistic Routing Framework

A. Framework Overview

A framework for probabilistic routing is presented. The Termite p.routing algorithm and a generic ACO based approach (hereafter referred to simply as ACO) are used as illustrative examples. Several proposed soft routing algorithms are based on ACO, including AntHocNet, ANSI, and ARA. The soft routing protocols proposed to date are essentially probabilistic distance vector protocols. There are two key components to any p.routing algorithm, the packet *forwarding equation* and the *metric estimator*. Each node estimates the average path cost (or inversely, the utility) to each destination through each neighbor based on routing data collected from received or overheard traffic. This information can be piggybacked on data packets, or found in control packets such as forward/backward ants [4]. The network is continuously sampled for changes by each node in order to maintain up-to-date information. The estimated route utilities are used to generate a probability distribution from which a next hop to a specific neighbor can be selected. Multipath routing is easily implemented since the utility of each neighbor to arrive at each destination is maintained by the metric estimator, and each neighbor is considered as a viable next hop by the forwarding equation. Soft routing is an application of dynamic optimization.

B. Packet Forwarding

1) Forwarding Equation: The forwarding equation determines the next hop probability distribution for each packet, given its destination. The next hop is selected according to this distribution; this is per-packet probabilistic routing. The pmf, p, is a normalization of the current route utility estimate, P, for each neighbor to deliver a message to the destination. Normalization is an intuitive mechanism used to send more packets over better neighbors and fewer over worse ones. Two parameters influence the forwarding equation, including the *sensitivity*, $F \geq 0$, and *threshold*, $K \geq 0$. The sensitivity modulates the differences between link utilities, making the resulting probabilities more or less dependant on them. It controls how much better paths are used more than worse ones. The threshold parameter determines how good a path through a neighbor must be before it has a substantial impact on the routing distribution. This parameter balances the sensitivity by pushing the routing distribution closer towards uniform for large K. The threshold is alternatively known as *noise*, $0 \leq q \leq 1$, which takes a different form in the forwarding equation but serves the same purpose. The balance between sensitivity and threshold/noise determines the tradeoff between network exploitation and exploration. The forwarding equation will be denoted as the function $\mathsf{p} = W(\mathsf{P}, K, F)$ or $\mathsf{p} = W(\mathsf{P}, q, F)$, whichever is appropriate. The matrix variables, p and P, correspond to routing probabilities and route utilities, respectively, and are detailed in Section III-A.

2) Estimate Uncertainty: Routing metric estimates become stale as the network changes, requiring them to be constantly updated. The next hop distribution should reflect the uncertainty of the underlying routing metric estimates at the current time. If an estimate is stale, then the corresponding link should be less probable. This intuition is included in the forwarding equation by multiplying the correlation between successive metric estimates, R, against the current estimate. Longer times between estimate updates yield a lower correlation and ultimately a smaller routing probability for that link. The correlation function depends on the type of metric estimator used. Section II-D reviews estimate correlation in more detail.

This method of accounting for estimate uncertainty formalizes efforts in previous work to account for continuous-time pheromone decay. Based on the biological inspiration of these algorithms, pheromone is used to represent average path quality. Pheromone continuously evaporates, establishing the need to replenish it with new deposits (and thus requiring new trials of the system). Algorithmically, this was accomplished by multiplying the current estimate by an exponential decay term based on the elapsed time since the last estimate was generated. This approach emulated exponential pheromone decay. Sample correlation is able to formalize this intuition and produce more general results for any type of estimation filter.

3) Termite: Equation 1 shows the Termite forwarding equation. $P^n_{i,d}$ is node n's estimate of the utility of using a path through neighbor i to deliver a packet to destination d. The normalization is with respect to all neighbors of n, $\mathcal{N}^n \subseteq \mathcal{V}$, where $\mathcal{V}$ is the set of all nodes in the network. The threshold must be set according to the expected range of the metric. $R(t - t^n_{i,d})$ is the correlation between the current time, t, and the current estimate (the last sample of which was generated

at $t^n_{i,d}$).

$$p^n_{i,d} = \frac{\left[P^n_{i,d} R\left(t - t^n_{i,d}\right) + K\right]^F}{\sum_{j \in \mathcal{N}^n} \left[P^n_{j,d} R\left(t - t^n_{j,d}\right) + K\right]^F} \quad (1)$$

4) ACO: A typical ACO type forwarding equation is shown in Equation 2. The noise parameter balances the routing distribution between a normalization of the link utilities and a uniform distribution across all outgoing links. If an instance of a random variable $v \sim U(0,1)$ is greater than the noise parameter $q \in [0,1]$, then the routing distribution is determined by a normalization of the link metrics. If the instance is less than the noise parameter, then a uniform routing distribution is chosen over all outgoing links.

$$p^n_{i,d} = \begin{cases} \frac{\left[P^n_{i,d} R\left(t - t^n_{i,d}\right)\right]^F}{\sum_{j \in \mathcal{N}^n} \left[P^n_{j,d} R\left(t - t^n_{j,d}\right)\right]^F} & , \quad v > q \\ |\mathcal{N}^n|^{-1} & , \quad v \leq q \end{cases} \quad (2)$$

Equation 3 simplifies this two part distribution into one. This equation will be used to represent ACO.

$$p^n_{i,d} = (1-q) \cdot \frac{\left[P^n_{i,d} R\left(t - t^n_{i,d}\right)\right]^F}{\sum_{j \in \mathcal{N}^n} \left[P^n_{j,d} R\left(t - t^n_{j,d}\right)\right]^F} + q \cdot \frac{1}{|\mathcal{N}^n|} \quad (3)$$

C. Metric Update

The metric update equation generates an estimate of the path utility from each node to each destination through each neighbor. Routing information is gathered either by proactively probing the network with control packets, or by passively collecting data from received and overheard packets. For the purposes of this analysis of the framework, how the information is collected is irrelevant, only that it is, and that the probe packets follow a routing rule defined by the forwarding equation. The returning information is treated as samples of a non-stationary stochastic process describing the change in path utilities. The samples are filtered to track the mean of the process, which is used as the estimate. Relevant parameters include the rate at which packets (those carrying routing information) are received, λ, and the network correlation time, T, which is the period of time over which the network statistics are assumed to remain stationary.

There are two commonly used methods for estimating path utility based on samples of the network, both of which are basically low pass filters. Equation 4 shows the traditional exponential filter (also known as a pheromone filter). It requires little state but does not make an optimal estimate of the link utility. It includes information from all received samples in the current estimate (ie., it has an infinite impulse response). The time constant of the filter, τ, is parameterized by the network correlation time, T. Here, $\gamma^n_{r,s}$ is the arrived path utility update at node n from source node s over previous hop r. $P^n_{r,s}$ is the estimate at node n to get to the destination s, which is the source of the arriving packet, through the previous hop, r.

$$P^n_{r,s} \leftarrow P^n_{r,s} \cdot e^{-\left(t - t^n_{r,s}\right)\tau} + \gamma^n_{r,s} \cdot \left[1 - e^{-\left(t - t^n_{r,s}\right)\tau}\right] \quad (4)$$

Equation 5 shows the optimal path utility estimator in the form of a sliding window, or box, filter, with length equivalent to the network correlation time, T. Received utility updates are indexed as $\gamma^n_{r,s}[m]$, and corresponding arrival times as $t^n_{r,s}[m]$. The optimality of this filter is discussed in more detail in [10], however it is so because incoming network samples are assumed to be iid over a time period T, in which case an average is the best estimate of the mean of the process.

$$P^n_{r,s} \leftarrow \left|\{m : t - T \leq t^n_{r,s}[m] \leq t\}\right|^{-1} \sum_{m : t-T \leq t^n_{r,s}[m] \leq t} \gamma^n_{r,s}[m] \quad (5)$$

D. Sample Correlation

The value of the correlation function, R, depends on the sort of filter that is used to estimate the path utility. Because different filters weight received path utilities differently in order to generate their estimates, the correlations between successive estimates also differ. Equation 6 shows the correlation of the exponential filter. It is calculated according to the standard methods of stationary random process and LTI filters. A more detailed derivation is available in the appendix. Δt is the elapsed time between any two estimates.

$$R_{exp}(\Delta t) = e^{-|\Delta t|\tau} \quad (6)$$

Outputs of the exponential filter are exponentially correlated by an amount depending on the inter-estimate time. But because the nonstationary input process has a finite correlation time of T, the correlation of the estimation filter should be also limited to this time. The decay constant, τ, is chosen such that the correlation is less than a chosen threshold, z, by time T. The threshold is chosen to be small, perhaps $z \approx 0.1$.

$$\tau = -\frac{\ln(z)}{T} \quad (7)$$

Equation 8 shows the output correlations of the box filter. The correlation of the box filter naturally drops to zero after time T.

$$R_{\square}(\Delta t) = \begin{cases} 1 - \frac{|\Delta t|}{T} & , \quad |\Delta t| \leq T \\ 0 & , \quad |\Delta t| > T \end{cases} \quad (8)$$

III. Markovian Analysis

A. Steady State Routing Probabilities

With a general framework for probabilistic routing defined, it is now necessary to describe how to find the steady state routing solution in a network, based on a given forwarding equation and metric update scheme. The equilibrium solution can then be used to show how the local parameters affect the global routing pattern and how they may be adjusted in order to achieve a given performance level. Each node is represented as a state in a Markov chain, and standard methods are used to find the statistics of paths from a source to a destination (represented as an absorbing state in the Markov chain) [21].

First the average path cost to a specific destination is calculated from every other node. This determines the path utility based on the current routing probabilities. The per-link packet arrival rate is then determined, which allows the average

sample correlation to be set. Routing probabilities are then recalculated, and at the equilibrium routing solution is found in an iterative manner.

1) Average Cost to Destination: The average number of state transitions from a source to the destination state is calculated. This is then used to determine the average end-to-end path cost, and establishes the path utility estimate for given routing probabilities. Suppose that the transition (or routing) probabilities for a network are given in a matrix p, shown in Equation 9.

$$\mathsf{p} = \begin{bmatrix} \mathsf{S} & \mathsf{T} \\ 0 & \mathsf{I} \end{bmatrix} \tag{9}$$

In this matrix, S is a square $(x-y)\times(x-y)$ matrix representing the transition probabilities between the nonabsorbing states, T is a $(x-y)\times y$ matrix representing the transition probabilities between the nonabsorbing states and the absorbing states, 0 is a $y\times(x-y)$ matrix of zeros, and I is a $y\times y$ identity matrix representing the self-transition probabilities of the absorbing states. In this analysis, $x=|\mathcal{V}|$ and $y=1$, since there are $|\mathcal{V}|$ total nodes and only one destination node. The fundamental matrix of the system is Q, calculated in Equation 10. Here, $\mathsf{Q}_{i,j}$ describes the expected number of visits to state j, starting in state i, before arriving at an absorbing state. The exponent is the matrix inverse operator.

$$\mathsf{Q} = (\mathsf{I}_{(x-y)\times(x-y)} - \mathsf{S})^{-1} \tag{10}$$

The expected number of states visited when starting at state i before being absorbed (at the destination state) is the sum of each row of Q, and is calculated in Equation 11.

$$\mathsf{c}_i = \sum_j \mathsf{Q}_{i,j} \tag{11}$$

If the cost of each link is unity, such as with hop count, then c_i is also the expected path cost from node i to the destination. In general, if the link costs are given in the matrix C such that $\mathsf{C}_{i,j}$ is the cost of the link from node i to j, then the cost from source to destination is calculated as in Equation 12.

$$\begin{aligned} \mathsf{c}_i &= \sum_j \mathsf{Q}_{i,j} \cdot E[\mathsf{C}_j] \\ &= \sum_j \left[\mathsf{Q}_{i,j} \cdot \left(\sum_{k\in\mathcal{N}^j} \mathsf{C}_{j,k} \cdot \mathsf{p}_{j,k} \right) \right] \end{aligned} \tag{12}$$

Equations 10 and 12 will be referred to as the function $\mathsf{c} = avgCostToDestination(\mathsf{C}, \mathsf{p})$.

With the average cost from each node to the destination calculated, it is then possible to recalculate the routing probabilities based on the forwarding equation. By repeating this process of calculating the steady state path costs based on current routing probabilities, and then recalculating the probabilities based on the steady state costs, it is possible to arrive at an equilibrium solution of routing probabilities based on the given link costs. As long as certain parameter settings are avoided, such as $K=0$, $q=0$, or $F=\infty$, trivial routing solutions (i.e., using only one path) will not be generated. The general algorithm is shown in Algorithm 1. A discussion of computational performance is moved to Section III-B.

Algorithm 1 Steady State Routing Probabilities

```
intialize p {random, nonzero, unity row sum}
while p not converged do
  c = avgCostToDestination(C, p)
  for all i, j ∈ (V − d) do
    P_{i,j} ← (C_{i,j} + c_j)^{-1} {link utilities}
  end for
  p ← W(P, K, F) {forwarding equation}
end while
```

2) Per Link Packet Arrival Rates: The next step in calculating the equilibrium routing probabilities is to include the decay factor, R, since there is a finite packet arrival rate on each link. It is assumed that packets with routing information are sent between a single source and destination pair with independently and identically distributed exponential interarrival times, with mean λ^{-1} seconds per packet. It is further assumed that their arrival rate at any given node is also independently distributed exponential with a link dependant mean sample rate, $\lambda_{i,j}$ packets per second from node i to j. These assumptions are a first-order approximation to the true arrival distribution. The packet arrival rate at node j from neighbor i when source s is sending a rate of λ packets per second to the destination, $\lambda^s_{i,j}$, is shown in Equation 13.

$$\lambda^s_{i,j} = \mathsf{Q}_{s,i} \cdot \mathsf{p}_{i,j} \cdot \lambda \tag{13}$$

3) Expected Per Link Sample Correlation: The expected correlation between the current time and the previous estimate can then be calculated according to the metric estimation filter. Equation 14 shows the expected correlation for the exponential filter, assuming exponential inter-arrival time between packets, the previously developed decay rate heuristic, and the τ heuristic of Equation 7.

$$\begin{aligned} E[R_{exp}] &= \int_0^\infty exp_{\Delta t}(\lambda^{-1}) \cdot R_{exp}(\Delta t) d\Delta t \\ &= \int_0^\infty \lambda e^{-\lambda(\Delta t)} \cdot e^{-(\Delta t)\tau} d\Delta t \\ &= \frac{\lambda T}{\lambda T - \ln z} \end{aligned} \tag{14}$$

A similar analysis results in Equation 15 for the expected correlation of box filter.

$$E[R_\square] = \frac{\lambda T - 1 + e^{-\lambda T}}{\lambda T} \tag{15}$$

The critical parameter influencing the change in expected correlation from one sample of the network to the next is the product λT. The units of this term may be thought of as *packets per network correlation time unit*, or *network samples per network correlation time unit*. Such language makes it clear that the framework relies on samples of the network in order to make decisions, and that performance is improved (this term is used qualitatively at the moment) with higher sampling rates.

4) Steady State Algorithm: The original Algorithm 1 can be updated in order to reflect a limited network sampling rate. This is shown in Algorithm 2, where a specific source s is assumed.

Algorithm 2 Steady State Routing Probabilities with R

```
intialize p {random, nonzero, unity row sum}
while p not converged do
  c ← avgCostToDestination(C, p) {Equations 10 and 12}
  for all i, j ∈ (V − d) do
    P_{i,j} ← (C_{i,j} + c_j)^{-1} {link utilities}
    P_{i,j} ← P_{i,j} · E[R] (λ^s_{i,j} T)
    {E[R] from Eqn. 14 or 15 depending on estimator}
    {λ^s_{i,j} from Equation 13}
  end for
  p ← W(P, K, F) {forwarding equation}
end while
```

a) An Optimal Network Sampling Rate: It has now been established that the information available about a network is dependent on only one parameter, namely λT, and that there is a useful lower bound to this value. It is unclear if there exists an optimal rate. If the rate tends towards infinity, then perfect information will be available about the network. But only a limited rate can be supported by the network, especially considering that data traffic must also be supported. There must exist a tradeoff between the quality of routing information, the cost to achieve a given quality, and the routing that can be accomplished with that information. [22] suggests a cross entropy approach which can adapt the sample rate to the current network conditions.

B. Algorithmic Performance

Two critical issues of algorithmic performance are convergence and computational complexity. No formal proof of convergence is given and is left to future work. However, experience shows that the algorithm converges under any circumstances, usually within a dozen iterations. This includes the avoidance of oscillations. The routing probabilities, p, are initialized nonzero and randomly. Convergence is determined based the sum squared difference of expected costs, c, between iterations. Once this difference falls below a given threshold, the algorithm halts.

Regarding computational issues, By far the slowest step is the matrix inversion required by Equation 10 in $avgCostToDestination$, which is executed in every iteration of the algorithm. Matrix inversion is $O(|\mathcal{V}|^3)$; the algorithm slows significantly as the network size grows. Execution time for a network of 50 nodes on an average machine is only 0.05 seconds.

IV. Numerical Analysis

A. Analysis Overview

Using the previously described framework, this section examines the relationships between the sensitivity, threshold/noise, sampling rate, and the equilibrium expected path cost between a source and destination.

B. On Sensitivity and Noise

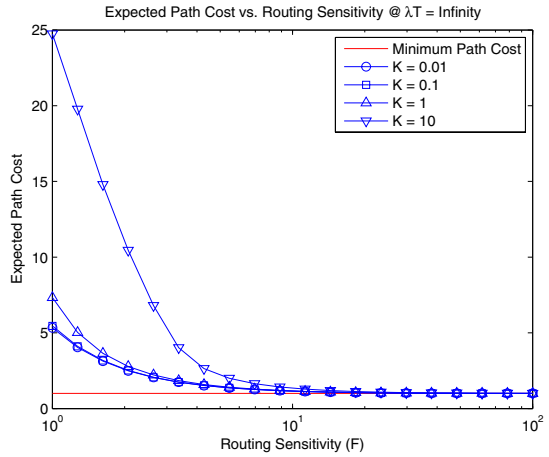
Figure 1 shows how the expected path cost between a source and destination vary with sensitivity and noise, using the Termite and ACO forwarding equations, as calculated by Algorithm 1. A network is randomly generated with 50 nodes, where node connectivity is determined by a uniform circular transceiver distance and link cost is inter-node distance squared, which is similar to an energy conservation metric. Transceiver range is set such that the average number of neighbors per node is eight; the network is connected. All results are the average of twenty such networks. The expected path cost approaches the minimum path cost, which is normalized in all cases, as the routing sensitivity is increased. Because the threshold is under the influence of the sensitivity, the Termite forwarding equation will send all packets on the shortest path if the sensitivity is large enough. This is not the case for the ACO equation, where the noise and sensitivity are independent. For nonzero noise, there is a lower bound on the expected path cost which is greater than the minimum path cost. The routing solution is also dependent on the network topology, associated link costs, and source-destination distance. In the interest of space, no figures are shown how the solution varies with network topology. The variance can be quite substantial, though the general relationship between F and K is the same for any given topology.

C. Effect of Network Sample Rate

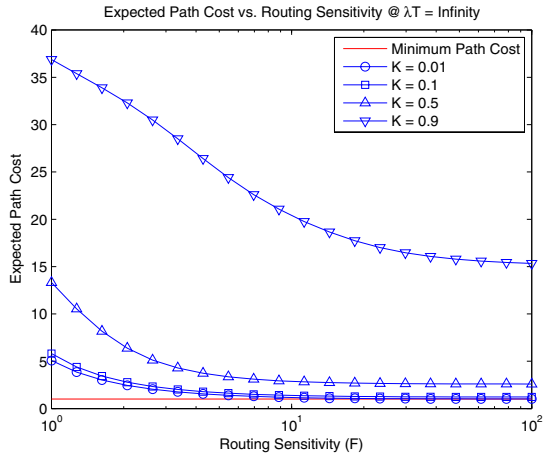
The network sample rate has a substantial effect on the performance of the routing algorithm. The samples are responsible for updating each node's estimate of how good each link is to arrive at each destination. Figure 2 shows the effect of the network sample rate on the expected path cost between source and destination, as calculated by Algorithm 2. For clarity, the results are shown varied with F and λT, where $K = 0.01$ and $q = .01$ are held constant, using the box filter. "λT = Inf" is a comparison with Algorithm 1, as it removes the effect of the sample correlation function.

A lower sample rate causes the algorithm to converge faster towards the shortest path than with a higher sample rate, assuming constant F. As good paths are found and their associated probabilities increase, the number of times that less attractive paths are sampled quickly drops (since only a limited number of samples can be made). The infrequent arrival of new metric information on poor paths makes the currently stored information unreliable. The estimates then have an even smaller effect on the routing probabilities, according to the correlation equation, R. A reasonable estimate for the optimal network sampling rate *may be* on the order of one to ten packets.

The results presented here are only valid for the sampling taking place for a single source-destination pair. This actually represents a pessimistic view of a functional network. Since routing information from several sources can be piggybacked on a single packet, and nodes can eavesdrop on broadcasted packets, the effective network sampling rate is higher than a simple superposition of multiple source-destination pairs may suggest.

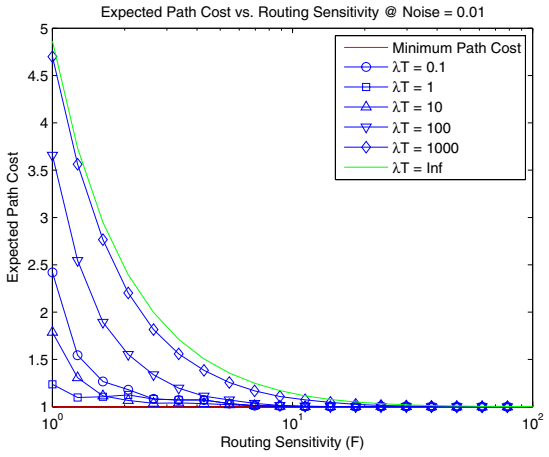


(a) Termite

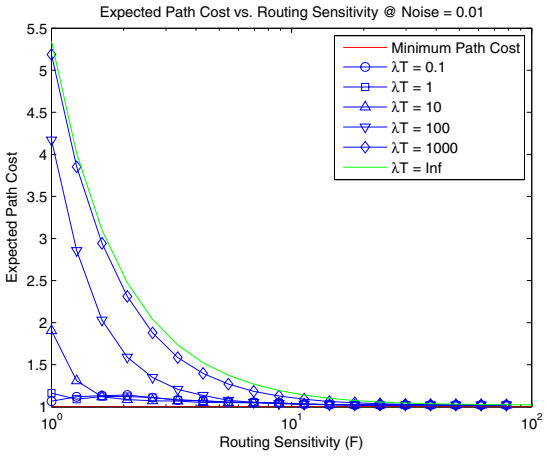


(b) ACO

Fig. 1. Normalized Expected Path Cost vs. Routing Sensitivity



(a) Termite (box filter)



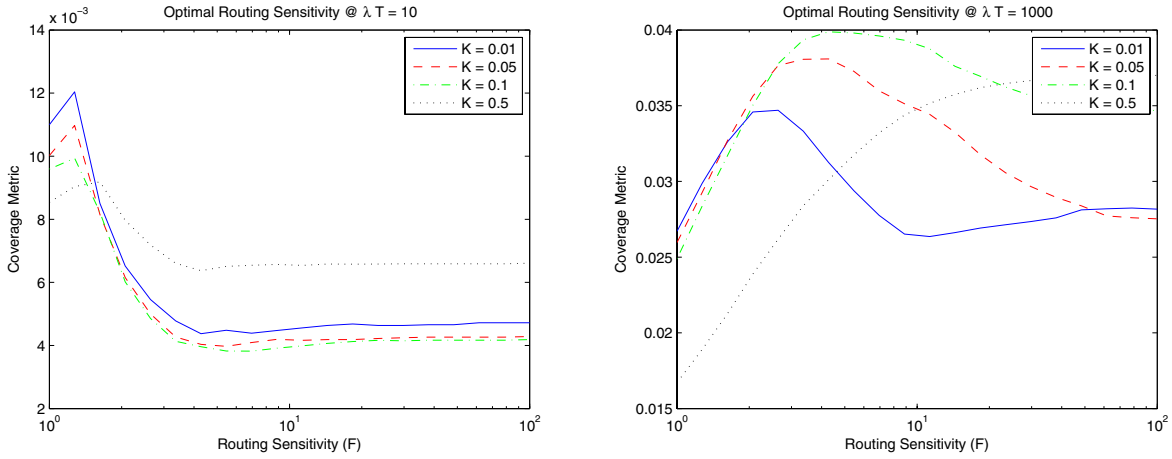
(b) ACO (box filter)

Fig. 2. Normalized Expected Path Cost vs. Routing Sensitivity

D. Towards An Optimal Sensitivity

There exists a sensitivity tradeoff. $F > F^*$, the optimal sensitivity, does not allow for enough network exploration and will not be able to track cost changes of many links over time. Packets are too sensitive to current metric estimates and will not take advantage of all paths to the destination. $F < F^*$ is able to track changes, but allows packets to wander the network. Too many packets follow grossly suboptimal paths. A simple metric is devised in order to measure this tradeoff. This metric consists of the fraction of links in the network that have an average packet rate of $\lambda T \geq 1$, divided by the stretch factor of that parameter setting. The stretch factor is the ratio of the expected path cost to the minimum path cost. The intuition behind this metric is that as many links as possible should be sampled with more than one packet per network correlation time, while minimizing the expected path cost. This metric is not perfect, as it is not comparable between networks of different sizes, or networks with pathological topologies, however it is suitable for current purposes. Figure 3 shows this metric for $\lambda T = 10$ and $\lambda T = 1000$, using only a single network as an example. In both subfigures, ACO is used with the exponential filter. In the low rate case, a low noise has superior performance than a high noise setting, reaching its peak at a relatively low sensitivity of $F^* \approx 1.2$. The high rate case allows more links to be sampled more often. As shown in Section IV-C, a higher sample rate increases the expected path cost. A higher sensitivity is needed to decrease this measure, resulting in a larger routing sensitivity necessary to reach the best coverage metric. In this case, $F^* \approx 4$.

These results can be generalized. The lower the packet rate, fewer links can be tested less often. The larger the topology, the more links are likely to exist between a given pair of nodes. It is expected that performance according to this metric will drop with larger topologies and constant λT. This is a similar



(a) $\lambda T = 10$ (b) $\lambda T = 1000$

Fig. 3. Coverage Metric vs. Routing Sensitivity (ACO, box filter)

effect to the low rate instability condition introduced in Section IV-C. This effect is partly unavoidable. The number of links in the network is simply a property of the graph, and if not enough packets are sent then the links cannot be tested. On the other hand, the stretch factor can generally be pushed as close to unity as desired with F approaching infinity (and $q \rightarrow 0$). Therefore, the sample rate between a single source destination pair must be adjusted to the hop distance between them, depending on how much information about how many paths should be maintained.

V. Conclusion

A framework for the analysis of soft routing algorithms has been presented. Global routing behaviors can be determined based on local parameters without the need for Monte-Carlo simulation. Termite and ACO are used as examples to illustrate a two part framework including a forwarding equation and a metric updating equation. The continuous time metric update is reviewed, including the use of the filter correlation function to properly model the loss of metric information over time. An analysis of the filter correlation shows that the critical parameter for updating the network with new information is the network sampling rate, or the number of route metric updates sent per network correlation time.

A Markov chain approach is used to find the steady state routing probabilities given the routing parameters and network costs. The expected path cost from source to destination can then be calculated. The cost is examined to reveal the tradeoffs between sensitivity, threshold/noise, and network sample rate in the context of average behavior and robustness against cost dynamics. A good parameter choice is a small threshold, a sensitivity proportional to the minimum path utility, and a sample rate proportional to the number of good paths between source and destination, which is dependent on network size.

Future work should compare the results of the model analysis to Monte-Carlo simulation experiments. Steady state convergence time should also be considered, since dynamic networks may not offer enough time for the soft routing protocol to converge. The existence of an optimal sampling rate will also be explored, as the physical effects of a large sampling rate are detrimental to the network at large.

Appendix

A. LTI Filtering of Stationary Random Processes

1) Autocorrelation of LTI Filter Output: The derivation of the filter correlation functions is shown. From the theory of random processes, the autocovariance function, $R_Y(\Delta)$, of the output of a linear time-invariant (LTI) filter, $h(t)$, with input from a random process with autocovariance function, $R_X(\Delta)$, is calculated for the continuous time case in Equation 16. This function is then normalized in order to produce the autocorrelation function, which is used in the forwarding equation.

$$R_Y(\Delta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(w)h(v)R_X(\Delta + (w-v))dwdv \quad (16)$$

The discrete time case is developed in parallel starting with Equation 17. The former is easier to analyze while the latter is more relevant because network samples arrive in discrete packets. The analysis of the discrete case does *not* take into account the irregular arrival of samples. This is left for future work. In a slight abuse of notation, the use of index $n \in \mathcal{I}$, will denote the discrete time version, while $\Delta \in \mathcal{R}$ denotes the continuous time version. There is also no visual distinction between the autocovariance and autocorrelation functions. Not all derivations and definitions are included for both versions when they are very similar. In discrete time, the floor of λT is taken in order to ensure that it is an integer.

$$R_Y(n) = \sum_{w=-\infty}^{\infty}\sum_{v=-\infty}^{\infty} h(w)h(v)R_X(n + (w-v)) \quad (17)$$

In this application, it is assumed that inputs to the metric estimator (the averaging filter) are *iid*, at least over some

correlation time T. R_X may thus be formalized as,

$$R_X(\Delta) = \begin{cases} 1 & , \quad \Delta = 0 \\ 0 & , \quad \Delta \neq 0 \end{cases} \tag{18}$$

Because R_X only exists with input zero, a substitution of $v = \Delta + w$ allows Equation 16 to be simplified to,

$$R_Y(\Delta) = \int_{-\infty}^{\infty} h(w)h(\Delta + w)dw \tag{19}$$

$$R_Y(n) = \sum_{w=-\infty}^{\infty} h(w)h(n + w) \tag{20}$$

2) Box Filter: For the case of the box filter presented previously,

$$h_{\square}(t) = \frac{1}{T}u(t)u(T - t) \tag{21}$$

$$h_{\square}(n) = \frac{1}{\lfloor \lambda T \rfloor}u(n)u(\lfloor \lambda T \rfloor - 1 - n) \tag{22}$$

where $u(t)$ is the step function. The length of the discrete time box filter is $\lfloor \lambda T \rfloor$ because that is the expected number of arrived samples within the network correlation time. By substituting $h_{\square}$ into Equations 19 and 20,

$$\begin{aligned} R_{\square}(\Delta) &= \int_{\max(0,-\Delta)}^{\min(T-\Delta,T)} \frac{1}{T^2}dw \\ &= \frac{T - |\Delta|}{T^2}u(\Delta + T)u(T - \Delta) \end{aligned} \tag{23}$$

$$\begin{aligned} R_{\square}(n) &= \sum_{w=\max(0,-n)}^{\min(\lfloor \lambda T \rfloor - 1 - n, \lfloor \lambda T \rfloor - 1)} \frac{1}{(\lfloor \lambda T \rfloor)^2} \\ &= \frac{\lfloor \lambda T \rfloor - |n|}{(\lfloor \lambda T \rfloor)^2}u(n + \lfloor \lambda T \rfloor)u(\lfloor \lambda T \rfloor - n) \end{aligned} \tag{24}$$

By normalizing this result to produce the required autocorrelation function, the result reported earlier in Equation 8 is recovered.

$$R_{\square}(\Delta) = \left(1 - \frac{|\Delta|}{T}\right) u(\Delta + T)u(T - \Delta) \tag{25}$$

3) Exponential Filter: For the case of the exponential filter,

$$h_{exp}(t) = \tau e^{-t\tau}u(t) \tag{26}$$

$$h_{exp}(n) = (1 - e^{-\tau})e^{-n\tau}u(n) \tag{27}$$

By substituting h_{exp} into Equation 19,

$$\begin{aligned} R_{exp}(\Delta) &= \int_{\min(0,\Delta)}^{\infty} \tau^2 e^{(-w\tau)} e^{-(w+\Delta)\tau} dw \\ &= \frac{\tau}{2} e^{-|\Delta|\tau} \end{aligned} \tag{28}$$

$$\begin{aligned} R_{exp}(n) &= \sum_{w=\min(0,n)}^{\infty} (1 - e^{-\tau})^2 e^{-w\tau} e^{-(w+n)\tau} \\ &= \frac{(1 - e^{-\tau})^2}{1 - e^{-2\tau}} e^{-|n|\tau} \end{aligned} \tag{29}$$

The normalized autocorrelation function is clearly the result reported in Equation 6.

$$R_{exp}(\Delta) = e^{-|\Delta|\tau} \tag{30}$$

By substituting for τ a simplified version is derived,

$$R_{exp}(\Delta) = z^{\frac{|\Delta|}{T}} \tag{31}$$

REFERENCES

[1] A. Tannenbaum, *Computer Networks*, Prentice Hall PTR, 2002.

[2] J. Boyan, M. Littman, *Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach*, *Advances in Neural Information Processing Systems*, Morgan Kaufmann, 1993.

[3] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, *Ant-Based Load Balancing In Telecommunications Networks*, *Adaptive Behavior*, 1996.

[4] G. Di Caro, M. Dorigo, *Mobile Agents for Adaptive Routing*,Technical Report, IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.

[5] M. Heusse, D. Snyers, S. Guérin, P. Kuntz, *Adaptive Agent-Driven Routing and Local Balancing in Communication Networks*, ENST de Bretagne Technical Report RR-98001-IASC, 1997.

[6] J. Baras, H. Mehta, *A Probabilistic Emergent Routing Algorithm for Mobile Ad-hoc Networks*, WiOpt '03: Modeling and Optimization in Mobile, Ad-Hoc, and Wireless Networks, 2003.

[7] M. Günes, M. Kähmer, I. Bouazizi, *Ant Routing Algorithm (ARA) for Mobile Multi-Hop Ad-Hoc Networks - New Features and Results*, The Second Mediterranean Workshop on Ad-Hoc Networks, 2003.

[8] M. Heissenbüttel, T. Braun, *Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks*, Kommunikation in Verteilten Systemen (KiVS), 2003.

[9] K. M. Sim, W. H. Sun, *Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 33, No. 5, September 2003.

[10] M. Roth, S.Wicker, *Termite: A Swarm Intelligent Routing Algorithm for Mobile Wireless Ad-Hoc Networks*, Springer SCI Series: Swarm Intelligence and Data Mining, Springer, 2005.

[11] S. Rajagopalan, C. Shen, *A Routing Suite for Mobile Ad-hoc Networks using Swarm Intelligence*, unpublished, 2004.

[12] F. Ducatelle, G. Di Caro, L. M. Gambardella, *Using Ant Agents to Combine Reactive and Proactive Strategies for Routing in Mobile Ad-Hoc Networks*, Technical Report No. IDSIA-28-04-2004, 2004.

[13] H. Wedde, M. Farooq, *The Wisdom of the Hive Applied to Mobile Ad-hoc Networks*, IEEE Swarm Intelligence Symposium 2005 (SIS 2005), June 2005.

[14] J. Dowling, E. Curran, R. Cunningham, V. Cahill, *Using Feedback in Collaborative Reinforcement Learning to Adaptively Optimise MANET Routing*, IEEE Transactions on Systems, Man and Cybernetics (Part A), Special Issue on Engineering Self-Orangized Distributed Systems, vol. 35, no. 3, pages 360-372, May 2005.

[15] http://www.faqs.org/rfcs/rfc2328.html

[16] P. Perkins, E. Royer, *Ad-hoc On-demand Distance Vector*, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[17] D. Subramanian, P. Druschel, J. Chen, *Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks*, Proceedings of the International Joint Conference on Artificial Intelligence, 1997.

[18] M. Roth, S. Wicker, *Asymptotic Pheromone Behavior in Swarm Intelligent MANETs: An Analytical Analysis of Routing Behavior*, Sixth IFIP IEEE International Conference on Mobile and Wireless Communications Networks (MWCN), 2004.

[19] M. Roth, *A Framework and Model for Probabilistic Routing: The Markovian Termite and other Curious Creatures*, The Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2006), September 2006.

[20] P. J. Brockwell, R. A. Davis, *Introduction to Time Series and Forecasting*, Spinger-Verlag, 2002.

[21] J. Norris, *Markov Chains*, Cambridge University Press, 1998.

[22] P. Heegard, O. J. Wittner, *Self-tuned Refresh Rate in a Swarm Intelligence Path Management System*, International Workshop on Self-Organizing Systems (IWSOS 2006), September 2006.