Termite:
A Swarm Intelligent Routing Algorithm for Mobile Wireless Ad-Hoc Networks

Martin Roth and Stephen Wicker

Wireless Intelligent Systems Laboratory
School of Electrical and Computer Engineering
Cornell University
Ithaca, New York 14850
USA
mhr8@cornell.edu
wicker@ece.cornell.edu

Summary. A biologically inspired algorithm named *Termite* is presented. Termite directly addresses the problem of routing in the presence of a dynamic network topology. In the Termite algorithm, network status information is embedded in the network through the passage of packets. Probabilistic routing decisions are based on this information such that the use of paths of maximum utility is an emergent property. This adaptive approach to routing greatly reduces the amount of control traffic needed to maintain network performance. The stochastic nature of Termite is explored to find a heuristic to maximize routing performance. The analysis focuses on the routing metric estimator, which is known as pheromone in the biological context. The pheromone decay rate is adjusted such that it makes the best possible estimate of the utility of a link to deliver a packet to a destination, taking into account the volatility, or correlation time, of the network. Termite is compared to Ad-hoc On-demand Distance Vector (AODV), showing the former to be superior in many primary performance metrics.

1 Introduction

1.1 Overview

Routing in mobile wireless ad-hoc networks (MANETs) is complicated by the fact that mobile users cause the network topology to vary over time. This paper shows how a probabilistic routing framework based on the principles of swarm intelligence (SI) directly and successfully addresses this problem. Swarm intelligence is a framework for designing robust and distributed systems composed of many interacting individuals, each following a simple set of rules [1]. The global behavior of the system is an emergent property and is not

preprogrammed at any level. Swarm intelligence is strongly related to artificial intelligence algorithms such as reinforcement learning [2] and optimization algorithms such as stochastic gradient descent [3].

Traditional approaches to the MANET routing problem use deterministic rules to discover optimal routes. Difficulty arises when the network changes quickly and all routes must be reevaluated. Termite opts for a probabilistic approach in which utility estimates of all routes are maintained simultaneously. Good paths are successively refined as the network environment changes. Control traffic is nearly eliminated by attaching a small amount of route information to each packet, including data. All traffic updates the network as it moves between source and destination.

Termite contains mechanisms for establishing a trade-off between network exploration and exploitation. These mechanisms are considered in the context of finding a heuristic for the optimal pheromone decay rate. Optimizing pheremone decay ensures that received routing information is effectively used to estimate the current routing metric, while maintaining its accuracy over time. Insights leading to performance improvement are achieved in part by modeling the received path utility information from each packet by a non-stationary stochastic process and filtering it to track the mean of the process over time.

The performance of Termite is assessed with a comparison to the standard Ad-hoc On-demand Distance Vector (AODV) routing protocol [4] in a simulated environment.

1.2 Previous Work

Traditional MANET Routing

Over the past ten years, an enormous number of MANET routing protocols have been proposed. They are generally based on distance vector or link state routing algorithms suggested nearly fifty years ago [5] [6] [7]. Current protocols are further broken into two categories, proactive and reactive. Examples of the former include DSDV [8] and OLSR [9], and examples of the latter are AODV [4] and DSR [10]. Such algorithms focus on finding and using the best available route at any given time. While able to manage relatively stable environments, this approach can find difficulty in high dynamic environments in which the network topology changes often and a large number of routing updates are forced. A more comprehensive review is available in [11].

Ad-hoc On-demand Distance Vector Routing Protocol (AODV)

Originally based on the proactive DSDV, the Ad-hoc On-demand Distance Vector (AODV) routing protocol is one of the most popular on demand routing protocols for ad-hoc networks. It is in the continuing stages of being standardized by the Internet Engineering Task Force (IETF) MANET working group [4] [12]. AODV was originally introduced in 1999 by Perkins and Royer

[13]. It has seen extensive testing and development since its introduction, and is used as a model for many new proposals.

The protocol operates generally as follows. When a data packet arrives at a node with no route to the destination, a route request procedure is initiated. Two variations exist; if a source node has no route to the destination, an expanding ring search is started. A local flood of route request (RREQ) packets is broadcast with a limited time-to-live. If no route reply (RREP) packet is received within a timeout period, another local flood is issued with a larger time-to-live, allowing it to search a larger area. This process is retried a certain number of times until the destination is found or is declared unreachable. Sequence numbers are embedded in the RREQs in order to distribute current information about the source, and to specify a minimum freshness for the information about the destination. Route metrics are also included in RREQ packets in order to create reverse routes to the source. If an intermediate node must search for a destination, this is known as a local repair and is comprised of only a local RREQ flood. If the destination cannot be found, a route error (RERR) is returned to the source to inform it that a new route must be discovered.

Route replies (RREP) are unicast from an intermediate node with an active route to the requested destination or from the destination itself. An active route is guaranteed back to the source since reverse routes are created by the RREQs. An active route to the destination should include a sequence number at least as large as that in the route request. Once a RREP is received, any packets that had been buffered for the destination can be sent to the destination.

A route error (RRER) message is needed to inform the source that an irreparable break has occurred in the route to the destination; a new route should be found. The route error propagates from the node immediately upstream of the break back to the source. The route to the destination is deactivated and each intermediate node will require an equally fresh new route to the destination.

The Social Insect Analogy

Social insect communities have many desirable properties from the MANET perspective. These communities are formed from simple, autonomous, and cooperative organisms who are interdependent for their survival. Despite a lack of centralized planning or any obvious organizational structure, social insect communities are able to effectively coordinate themselves to achieve global objectives. The behaviors which accomplish these tasks are often emergent from much simpler behaviors, or rules, that the individuals are following. The coordination of behaviors is also robust, necessary in an unpredictable world. No individual is critical to any operation and tasks can recover from almost any setback. The complexity of the solutions generated by such simple indi-

vidual behaviors indicates that the whole is truly greater than the sum of the parts [14].

Such characteristics are desirable in the context of ad-hoc networks. Such systems may be composed of simple nodes working together to deliver messages, while resilient against changes in its environment. The environment of an ad-hoc network might include anything from its own topology to physical layer effects on the communications links, to traffic patterns across the network. A noted difference between biological and engineered networks is that the former have an evolutionary incentive to cooperate, while engineered networks may require alternative solutions to force nodes to cooperate [15] [16].

The ability of social insects to self organize relies on four principles: positive feedback, negative feedback, randomness, and multiple interactions. A fifth principle, stigmergy, arises as a product of the previous four [1]. Such self organization is known generally as swarm intelligence.

How to Build a Termite Hill

A simple example of the hill building behavior of termites provides a strong analogy to the mechanisms of Termite and SI routing in general. This example illustrates the four principles of self organization [17]. A similar analogy is often made with the food foraging behavior of ants.

Consider a flat surface upon which termites and pebbles are distributed. The termites would like to build a hill from the pebbles, ie. all of the pebbles should be collected into one place. Termites act independently of all other termites, and move only on the basis of an observed local pheromone gradient. Pheromone is a chemical excreted by the insect which evaporates and disperses over time.

A termite is bound by these rules:

1. A termite moves randomly, but is biased towards the locally observed pheromone gradient. If no pheromone exists, a termite moves uniformly randomly in any direction.
2. Each termite may carry only one pebble at a time.
3. If a termite *is not* carrying a pebble and it encounters one, the termite will pick it up.
4. If a termite *is* carrying a pebble and it encounters one, the termite will put the pebble down. The pebble will be infused with a certain amount of pheromone.

With these rules, a group of termites can collect dispersed pebbles into one place. The following paragraphs explain how the principles of swarm intelligence interplay in the hill building example.

Positive Feedback

Positive feedback often represents general guidelines for a particular behavior. In this example, a termite's attraction towards the pheromone gradient biases it to adding to large piles. This is positive feedback. The larger the pile, the more pheromone it is likely to have, and thus a termite is more biased to move towards it and potentially add to the pile. The greater the bias to the hill, more termites are also likely to arrive faster, further increasing the pheromone content of the hill.

Negative Feedback

In order for the pheromone to diffuse over the environment, it evaporates. This evaporation consequently weakens the pheromone, lessening the resulting gradient. A diminished gradient will attract fewer termites as they will be less likely to move in its direction. While this may seem detrimental to the task of collecting all pebbles into one pile, it is in fact essential. As the task begins, several small piles will emerge very quickly. Those piles that are able to attract more termites will grow faster. As pheromone decays on lesser piles, termites will be less likely to visit them again, thus preventing them from growing. Negative feedback, in the form of pheromone decay, helps large piles grow by preventing small piles from continuing to attract termites.

In general, negative feedback is used to remove old or poor solutions from the collective memory of the system. It is important that the decay rate of pheromone be well tuned to the problem at hand. If pheromone decays too quickly then good solutions will lose their appeal before they can be exploited. If the pheromone decays too slowly, then bad solutions will remain in the system as viable options.

Randomness

The primary driving factor in this example is randomness. Where piles start and how they end is entirely determined by chance. Small fluctuations in the behavior of termites may have a large influence in future events. Randomness is exploited to allow for new solutions to arise, or to direct current solutions as they evolve to fit the environment.

Multiple Interactions

It is essential that many individuals work together at this task. If not enough termites exist, then the pheromone would decay before any more pebbles could be added to a pile. Termites would continue their random walk, without forming any significant piles.

Stigmergy

Stigmergy refers to indirect communications between individuals, generally through their environment. Termites are directed to the largest hill by the pheromone gradient. There is no need for termites to directly communicate with each other or even to know of each other's existence. For this reason, termites are allowed to act independently of other individual, which greatly simplifies the necessary rules.

Swarm Intelligent MANET Routing

The soft routing protocols proposed to date are essentially probabilistic distance vector protocols. The cost to each destination over each neighbor is estimated by each node based on routing data in received or overheard traffic. The next hop of a packet is chosen based on a distribution proportional to the routing utility of using each neighbor to arrive at the packet's destination. There are thus two key components to any probabilistic routing algorithm, the packet forwarding equation and the metric estimator (pheromone accounting). The former determines the routing distributions based on the metric estimates. The latter is responsible for producing an estimate of the cost (or inversely, the utility) to arrive at each destination through each neighbor. Costs are estimated by probing the network with packets (control or data, depending on the implementation). The network is essentially sampled for changes, with the results tabulated at each node. Because many of the network characteristics are dynamic, the estimates must be continuously updated. Maintaining per neighbor routing information allows multipath routing to be easily implemented.

There are a number of examples showing how SI routing can provide a significant performance gain over traditional deterministic approaches. These include ABC [18], AntNet [19], CAF [20], PERA [21], ARA [22] [23], MABR [24], Termite [25], ANSI [26], and AdHocNet [27] [28] [29]. A wider summary is found in [30].

VWPR and Pheromone Aversion

Virtual-Wavelength Path Routing (VWPR) introduces the concept of source pheromone aversion (SPA) to the SI routing solution [31]. This is an adaptation of the packet forwarding process in order to take advantage of additional routing information already available at each node. Packets usually only follow the pheromone gradient of their destination. SPA forces the packets away from their own source's pheromone, thus simultaneously pulling and pushing the packet towards its destination.

SPA is also used in the Multiple Ant Colony Optimization (MACO) algorithm [30] as a means to find a ranking of good solutions. MACO expands on the ant colony metaphor for routing by placing two competing colonies in a network to find routes; each is repelled by the other, thus forcing them to alternative solutions.

1.3 Structure of Paper

Section II provides a detailed explanation of the Termite MANET routing protocol. Some variants are described which allow comparison to Dijkstra's shortest path algorithm. Their performance is then compared to AODV in order to provide a common perspective with an established routing solution. A variety of conditions of interest to ad-hoc network engineers are investigated. Section III develops and tests a heuristic for the pheromone decay rate which maximizes performance. It should help in the selection of parameters to achieve optimal routing performance. A statistical approach is used to explain the performance of the system, and a solution is proposed. Section IV concludes the chapter.

2 Termite

The SI MANET routing protocol Termite is presented in detail. It is compared to the Ad-hoc On-demand Distance Vector (AODV) routing algorithm, a state-of-the-art and standards track technology. It is shown that the SI framework can be used to competitively solve the MANET routing problem with a minimal use of control overhead. A small amount of control information is imbedded in every data packet, which is usually sufficient for the network to maintain a current and accurate view of its state. The end result is a routing algorithm requiring only data traffic in the network under many circumstances. The version of Termite presented here takes advantage of a number of enhancements over a generic SI approach, including a generalized packet forwarding equation and source pheromone aversion.

2.1 Termite

The routing algorithm presented here is similar hill building proceedure presented earlier. Termite associates a specific pheromone scent with each node in the network. Packets moving through the network are biased to move in the direction of the pheromone gradient of the destination node, as biological termites are biased to move towards a hill. Packets follow the pheromone gradient while laying pheromone for their source on the same links; this is positive feedback. The specific amount of pheromone deposited by a packet on a link, as well as how that pheromone behaves over time, is governed by the pheromone accounting process. Changes in the network environment, such as topological or path quality changes, are accounted for by allowing pheromone to decay over time; this is negative feedback. Paths that are not reinforced are rendered less attractive for future traffic. Each node maintains a table indicating the amount of pheromone for each destination on each of its links. The pheromone table acts as a routing table similar to those found in traditional distance vector routing algorithms. The pheromone table also represents a

common area for packets to stigmergetically interact in order to converge on good routes.

Pheromone Table

The pheromone table maintains the amount of pheromone on each neighbor link for each known destination. It serves the same purpose as the routing table of a distance vector routing protocol, with analogous operations on its elements such as neighbor, destination, and routing metric management. Columns represent destinations and rows neighbors. Neighbors also appear in the table as destinations. When a neighbor is gained, an extra row is added to each column with the pheromone initialized to zero. An extra column is added to represent the neighbor, since the new node can also be a packet destination. If a neighbor is lost, the corresponding row is removed from the table. When a destination is gained, the current list of neighbors is replicated for the new destination but with all pheromone values reset to zero. If a destination is lost from the pheromone table, the column is simply removed.

A neighbor row is removed only when the link is explicitly lost through communications failure. There is no HELLO message link maintenance mechanism as is found in other protocols, which retains link information based on the arrival of specially formatted neighbor beacons. Even if the pheromone on that link decays, its row is still retained since the neighbor still exists as a communications option (Presumably. Although in the MANET setting, the neighbor may well have left communications range). A destination is removed if all of the pheromone in its column decays to zero. In this way Termite will know when to issue a route request for that destination. This procedure is initiated if the destination does not exist in the pheromone table.

For all nodes, n, in the network, pheromone values in the pheromone table are referenced with, $P^n_{i,d}$, where $i \in \mathcal{N}^n$, and $d \in \mathcal{D}^n$. These sets represent the current set of neighbors and destinations that node n is aware of. $P^n_{i,d}$ is thus the amount of pheromone at node n for destination node d on the link to neighbor node i.

Packet Forwarding

The forwarding equation with source pheromone aversion is used to determine the next hop neighbor. The forwarding equation models a packet probabilistically following a pheromone gradient to its destination, and repelled by the gradient towards the source. Equation 1 maps the destination d pheromone on each outgoing link i at node n, $P^n_{i,d}$, to the "pull" of that link to forward the packet to the destination, $p^n_{i,d}$. The source pheromone distribution, $p^n_{i,s}$, is computed similarly according to Equation 2, and represents the "push" of a link away from the source. The specific next hop neighbor is then chosen

randomly according to the meta distribution, $\hat{p}^n_{i,d}$, in Equation 3, which reflects the total effect of source pheromone aversion. The forwarding equation in each case is a normalization of the resident link pheromone.

Per packet computation of forwarding distributions could become too demanding for small processors handling high packet rates. In such a scenario, it would be possible to implement some optimizations, such as updating the distributions periodically in time of number of received packets.

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

$$p^n_{i,s} = \frac{\left[P^n_{i,s} e^{-(t-t^n_{i,s})\tau} + K\right]^F}{\sum_{j \in \mathcal{N}^n} \left[P^n_{j,s} e^{-(t-t^n_{j,s})\tau} + K\right]^F} \tag{2}$$

$$\hat{p}^n_{i,d} = \frac{p^n_{i,d} \left(p^n_{i,s}\right)^{-A}}{\sum_{j \in \mathcal{N}^n} p^n_{j,d} \left(p^n_{j,s}\right)^{-A}} \tag{3}$$

The exponential term multiplied against each pheromone value is included in order to deemphasize older pheromone values. The current time is t and the last time that a packet arrived from node d (or s) at node n on the link to neighbor i is $t^n_{i,d}$. The pheromone decay rate is τ. There are two primary reasons for such a deemphasis. The first is that according to the original biological inspiration, pheromone always decays; the exponential term updates the current pheromone measurement to model this continuous depreciation. A more technical reason to include such a term is that a method is needed to account for changes in the networking environment since the last received path utility measurement. The exponential term models this diminishing confidence in the accuracy of the metric estimator (the pheromone). This term is based on the network correlation time, as shown in Section III. If no packet is received from a destination for some time, the routing probabilities now reflect the fact that less timely information is available about the path quality to the destination. The routing probabilities tend towards a uniform distribution over all neighbor nodes. Past swarm intelligent routing algorithms have decayed pheromone either on regular intervals or only upon packet arrival [21] [22] [23] [25] [26].

The constants F, K, and A are used to tune the routing behavior of Termite. The *pheromone threshold*, K, determines the sensitivity of the probability calculations to small amounts of pheromone. If $K \geq 0$ is large, large amounts of pheromone will have to be present before an appreciable effect is seen on the routing probability. The nominal value of K is small compared to the expected pheromone levels. The probability of any link is prevented from going to zero, ensuring a minimum level of exploration of the network. Similarly, the *pheromone sensitivity*, $F \geq 0$, modulates the differences between

pheromone amounts. $F > 1$ accentuates differences between links, while $F < 1$ will deemphasize them. $F = 1$ yields a simple normalization. The *source aversion sensitivity*, $A \geq 0$, controls the amount by which the packet is repelled by the source pheromone.

Packet Pheromone Accounting

Pheromone carried by a packet is updated immediately upon packet reception to reflect the utility of the previous hop. This is done as shown in Equation 4, where γ is the pheromone resident on the packet (equivalent to the packet's total path utiltiy) and $c_{r,n}$ is the cost of the link from previous hop r to current node n. Note that utility is the inverse of cost, only costs are additive (not utilities), and that costs are strictly non-negative (and in any realistic implementation, positive).

$$\begin{aligned} \gamma &\leftarrow \left(\gamma^{-1} + c_{r,n}\right)^{-1} \\ &\leftarrow \frac{\gamma}{1 + \gamma c_{r,n}} \end{aligned} \tag{4}$$

The link pheromone is then set according to the pheromone update method with the new packet pheromone value.

Pheromone Update Methods

The pheromone table is updated with new pheromone values only upon packet arrival. A pheromone update method describes how the update is managed based on the pheromone carried by a packet which arrives at node n from source node s and previous hop r. If n is not designated as the packet's next hop, the node updates the pheromone table in the way described here and drops the packet. The time at which the packet is received is t, and the last time at which the pheromone for node s on the link to r was updated at node n on is $t^n_{r,s}$.

Three update methods are reviewed, including the γ pheromone filter, the normalized γ pheromone filter, and probabilistic Bellman-Ford. These methods are chosen due to their prevalence in the litterature and their favorable emergent properties.

γ Pheromone Filter (γPF)

γPF is the classic biological model for pheromone update. Current pheromone decays based on the amount of time since the last packet arrived, and the pheromone carried by the new packet, γ, is added to the total. γ is equivalent to the utility of the path that the packet has taken.

$$P^n_{r,s} \leftarrow P^n_{r,s} e^{-\left(t - t^n_{r,s}\right)\tau} + \gamma \tag{5}$$

γPF is directly comparable to a biological model of pheromone deposition with exponential decay.

Normalized γ Pheromone Filter ($\bar{\gamma}$PF)

$\bar{\gamma}$PF is a normalized version of γPF. It is a normalized one-tap infinite impulse response averaging filter. Only a fraction of the received pheromone is added based on link observation time. $\bar{\gamma}$PF effectively limits the amount of pheromone on a link with its averaging properties. γPF allows much larger amounts since it is unnormalized.

$$P_{r,s}^n \leftarrow P_{r,s}^n e^{-\left(t-t_{r,s}^n\right)\tau} + \gamma\left[1 - e^{-\left(t-t_{r,s}^n\right)\tau}\right] \tag{6}$$

$\bar{\gamma}$PF has no biological equivalent. But by virtue of implementing a low pass filter, it is able to produce an estimate of the average utility of using a particular neighbor to arrive at a particular destination. γPF produces only a relative ranking, and ultimately causes only the best links to be used [2].

Probabilistic Bellman-Ford (pBF)

The probabilistic Bellman-Ford algorithm is designed to turn Termite into an asynchronous version of the Bellman-Ford (link state) routing algorithm. Unlike the filter techniques presented, this one is nonlinear. If a packet is received with information of a better path over the receiving link than is already known (taking into account pheromone decay), then the utility estimate is updated. Otherwise the pheromone table entry is left untouched.

$$if\ P_{r,s}^n e^{-\left(t-t_{r,s}^n\right)\tau} < \gamma,\ P_{r,s}^n \leftarrow \gamma \tag{7}$$

Route Discovery

In case there does not exist any destination pheromone at a node for a packet to follow, a route discovery procedure is initiated. Termite uses the traditional flooding approach and does not use any optimizations such as gossiping [32] or the expanding ring search as found in AODV. The reason for this is primarily one of complexity; because Termite requires so little control traffic in the first place, the extra complexity required to manage flooding optimizations is not deemed necessary. The use of flooding optimizations (or some other route discovery scheme alltogether) is entirely possible, and would result in a yet lower control packet overhead. Another tradeoff for route discovery schemes is discovery delay. A route request packet (RREQ) is broadcast and each receiver rebroadcasts it if it cannot answer the query. If the RREQ has been received before, the packet is dropped. Route requests also serve to spread source pheromone into the network. A route reply (RREP) packet is returned if a node is the request destination, or has destination pheromone in its pheromone table. This is done even if other neighbors have already transmitted a route reply to the RREQ source. A RREP is unicast back to the source normally by probabilistically following the source pheromone in the same way that data packets do. A RREP is formatted such that its source is the destination of the

RREQ. This creates a destination pheromone trail back to the RREQ source. It is necessary to maintain a list of previously seen RREQ packets according to the source address and a source-unique sequence number in order to prevent the retransmission of previously received RREQs.

The packet that triggers a route request is cached for a route request timeout period before it is dropped. Any additional packets received during this period for the same destination are also cached. If the hold time since the first packet was held is exceeded then all of the held packets for the sought after destination are dropped. If a route reply is received while there are packets cached for the destination, they are processed normally according to the forwarding equation.

Route Repair

Termite has no concept of route repair in the traditional sense. Each next hop is computed online, and every node has an estimate of the utility of each link to deliver a packet to a destination. If a link should fail, the neighbor is simply removed from the routing table, the next-hop probabilities are recomputed for the remaining set, and the packet retransmitted. If all neighbor nodes are found to have disappeared (possibly after many unsuccessful retransmissions), the packet is dropped. There is no such thing as a route error or route error packet (RRER) in Termite.

Loop Prevention

A typical approach to loop prevention is for each node to maintain a list of packets already routed. If the same packet is seen a second time, an error procedure is executed, such as dropping the packet or sending control traffic into the network to fix routing tables [4] [10] [23] [28].

Termite does not make any special effort to prevent loops. All received packets are handled as described, regardless of the number of times they have visited any particular node. SPA helps to mitigate the number of routing loops by making hops towards the source less likely. Nodes closer to the source (measured according to the network metric) will have more source pheromone and less destination pheromone on them; travel towards the source is generally also travel away from the destination.

Termite is also very liberal with other packet reprocessing issues. Consider the case when node n transmits a packet to neighbor h, which then forwards the packet itself. Node n will overhear the retransmission and processes this packet normally. The effect is not adverse because h will have updated the pheromone on the packet, which will have less an effect on n's table than the original update did since the utility can only go down. Besides, it is important to keep track of underperforming alternative routes in case they improve.

Hello Packets

Some protocols use special hello packets to help nodes advertise their existence to the network. This functionality is also possible in Termite if a node transmits a route request for itself. The time-to-live of such a packet should be set to one. Each receiving node will automatically have pheromone for the requesting node due to the pheromone update procedure, and will be able to return a route reply. The advertising node then also learns about its neighbors (and possibly they about each other). A node broadcasts self requests only when it has no known neighbors (the pheromone table is empty).

The implementation of Termite presented here does not use this HELLO functionality.

Packet Structure

There are three types of packets in Termite, data (DATA), route request (RREQ), and route reply (RREQ). They can be all considered within one generic packet format, with the fields listed below. This list does not assume a pheromone stack, first introduced by AntNet, which maintains the per hop metric in each packet. Certain information such as the previous and next hop IP address can often be obtained from lower layers of the network stack, as seen in the AODV specification [4]. The total size of the header shown here is 24 bytes, excluding any user data. If necessary, an additional four bytes of flags could be added.

- *Packet Type* This field is one byte in size. It's value describes the purpose of the packet, data, route request, or route reply.
- *Source IP Address* This field is four bytes and describes the IP address of the data source.
- *Destination IP Address* This field is four bytes and describes the IP address of the data destination.
- *Previous Hop IP Address* This field is four bytes and describes the IP address of the previous hop.
- *Next Hop IP Address* This field is four bytes and describes the IP address of the next hop.
- *Pheromone* This field is four bytes and describes the amount of pheromone carried by the packet.
- *Time-To-Live (TTL)* This field is one byte and describes the remaining allowed hop-count for the packet. It is initialized to the maximum TTL and decremented at each visited node. The packet is dropped when this counter reaches zero.
- *Data Length* This field if two bytes and describes the length of the data field in this packet; the amount of data carried by this packet.
- *Data* This field contains all of the data carried by the packet.

General Mechanisms

Two general mechanisms are used to enhance the effectiveness of the algorithm.

Piggybacked Routing Information

Packets carrying routing information is not a new technique by any means. However, it should be stressed that this information is piggybacked on *all* packets. Algorithms such as ANSI or AdHocNet use a stack to store more path history, but do so only for control packets. This approach is not currently implemented in Termite in order to reduce the complexity of the algorithm, though it would likely result in a performance increase at the cost of additional packet overhead.

Promiscuous Mode

Nodes are expected to exist in a broadcast (radio) medium. They may eavesdrop on the communications of neighbors and incorporate overheard routing data into their own routing table. This technique is used in many other MANET routing algorithms, and is found to be particularly useful in Termite. In principle it is not obligatory, however it does afford a notable increase in performance.

2.2 Simulation

A number of different scenarios are simulated to compare the performance of Termite using the presented pheromone update methods. These results are also compared to the standard MANET Ad-hoc On-demand Distance Vector routing protocol, as described in [4]. Simulations are designed to test the effect of node mobility on the global performance metrics.

Simulation Environment and Parameters

A common test scenario is used in which 100 mobile nodes are distributed uniformly over an area 2200 meters by 600 meters. Each node uses a simulated IEEE 802.11b MAC layer with 2 Mbps data rate and a 250 meter transmission range. The standard MAC layer has been modified to allow promiscuous reception of all in-range transmitted packets, and also to return unsuccessfully transmitted packets back to the routing layer for reassessment. Nodes move according to the random waypoint mobility model with zero pause time and a uniform speed. These parameters are the same as those in [33]. Speeds are varied over 1, 5, 10, and 20 meters per second. All runs are 600 seconds long and all data points are averaged over at least two runs for high speeds and five runs for low speeds. Ten nodes send 512 byte data packets with exponentially distributed interarrival times with a mean of 0.5 seconds to a unique

communications partner, which replies to each received packet with an acknowledgement. Both AODV and Termite optimize for path length. AODV parameters are set according to the specifications found in [4]. The maximum time-to-live (TTL) for any packet is 32. Termite parameters include, $K = \frac{1}{32}$, $F = 10.0$, $\tau = 2.0$, and $A = 0.5$. The value of K is determined by the minimum possible received pheromone value. Since the TTL of any packet is 32, the minimum utility of any path is $\frac{1}{32}$. Termite holds packets for as long as AODV's ACTIVE_ROUTE_TIMEOUT parameter, which in this case is 1.28 seconds. The parameters are generally chosen to reflect a generic mobility and communciations scenario using radio characteristics similar to commercially available hardware, and mirror parameter choices made by the ad-hoc routing community. All simulations are completed using Opnet [34].

Evaluation Metrics

A number of metrics are used to determine the utility of the evaluated algorithms. These include data goodput, control packet overhead, control packet distribution, medium load, medium efficiency, medium inefficiency, link failure rate, and end-to-end delay.

Data Goodput

Data goodput is a classic evaluation metric for routing algorithms. It is the fraction of successfully delivered data packets. This metric should remain as high as possible under any circumstances.

Control Packet Overhead

Any routing algorithm should use as little control traffic as possible in order to successfully deliver data packets. Control packet overhead measures the fraction of control packets to the total number of transmitted packets in the system. Successively transmitted packets are counted individually.

Control Packet Distribution

This metric shows how many of each type of control packet were transmitted. This helps to identify the effectiveness of route request and discovery procedures.

Medium Load

This metric characterizes how inefficient the algorithm is in delivering packets. It is the ratio of the total number of packet transmissions, data or control, to the number of data packets successfully delivered. Successively transmitted packets are counted individually. This is not a general metric, but since the algorithms are optimizing for hop count, the fewer the transmissions the better. This metric should be as low as possible, however it will always be larger than the path length.

Medium Efficiency

Medium efficiency is the ratio of the number of transmissions of successfully arriving data packets to the number of total packet transmissions. Multiple transmissions of the same packet are counted individually. Since access to the communications medium often comes at a premium, it is important that it is only accessed in order to move packets that will arrive at their destination. Medium efficiency is a number between zero and one and values close to unity are desired. That would indicate that the only transmissions in the system are those for packets that are delivered.

Medium Inefficiency

This metric is related to the previous two and helps to fill in the full performance picture. Medium inefficiency is the fraction of transmitted packets to the number of data packets offered to the network for delivery. Lower numbers are better. Consideration of this metric should ensure that the routing algorithm is making an effort to deliver all packets, instead of just ones that are easy to deliver.

Link Failure Rate

The link failure rate measures the average number of links that are lost per node per second. It is a relative measure of how fast the network topology is changing, and thus how much time each node has to acquire a sensible local routing pattern before its local topology transforms.

End-To-End Delay

The average end-to-end delay of all successfully delivered data packets is measured. This metric gives another perspective on the overall performance of each algorithm. Delay should be minimized.

Results and Analysis

Data Goodput

As shown in Figure 1, the data goodput performance of Termite is higher than that of AODV. The latter is able to deliver at least 90% of its packets, and Termite outperforms it with a moderate 95%. The former sees a more graceful degradation of performance over the latter as node speed increases.

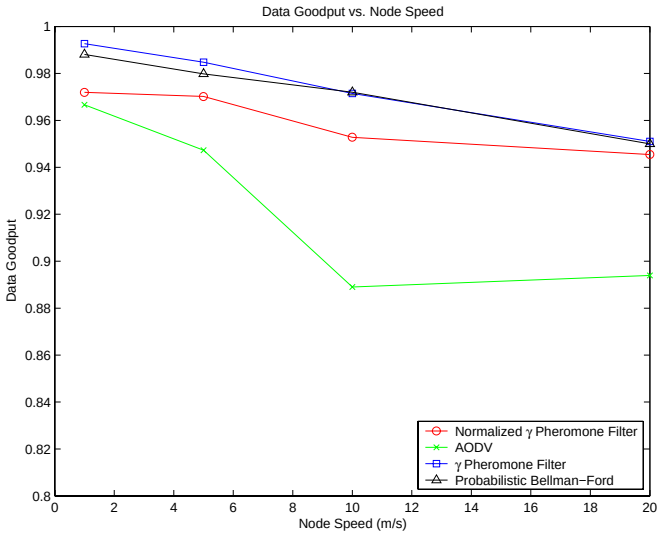

Fig. 1. Data Goodput vs. Node Speed

Control Overhead

Termite shows favorable control overhead properties as compared to AODV in Figure 2. Not only does Termite have an order of magnitude less control overhead, but it also produces a nearly constant amount over a large range of node mobility. This is true despite the use of flooding for route discovery, and speaks to the effective use of route information caching on the part of Termite. AODV suffers so much overhead because it floods the network with a new route discovery every time a route breaks. This weakness is partially addressed with an expanding ring search [12].

Control Packet Distribution

Termite uses little control traffic compared to AODV. As mentioned above, AODV must issue a route discovery flood whenever there is a route break. Termite is spared this because of its retransmission link repair policy. A full route discovery is almost never needed, which eliminates the majority of the control overhead. For AODV this proportion increases with node speed as links break more often. The most limiting factor is the number of route request packets; there are so many because such packets are flooded which ultimately generates a choking amount of overhead.

Alternatively, Termite produces more route reply packets than route requests. This attests to both the liberal route reply policy (any overhearing node with route information can generate a route reply) and also to the route caching of Termite. Since each node keeps route information about every other node, it is not difficult to find a pheromone trail.

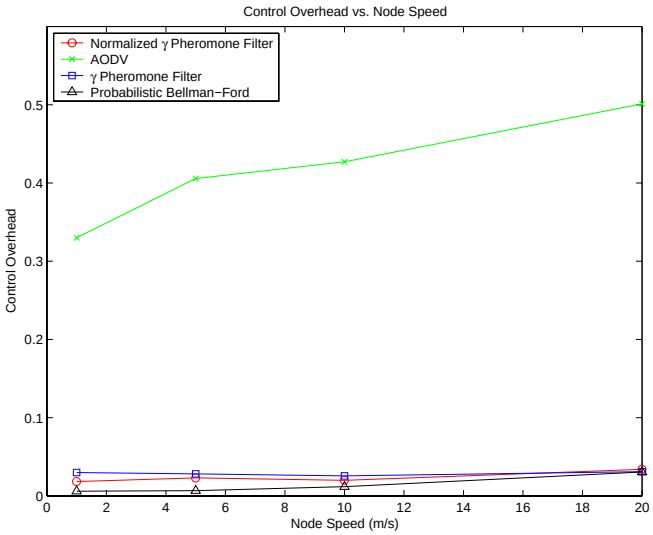

Fig. 2. Control Overhead vs. Node Speed

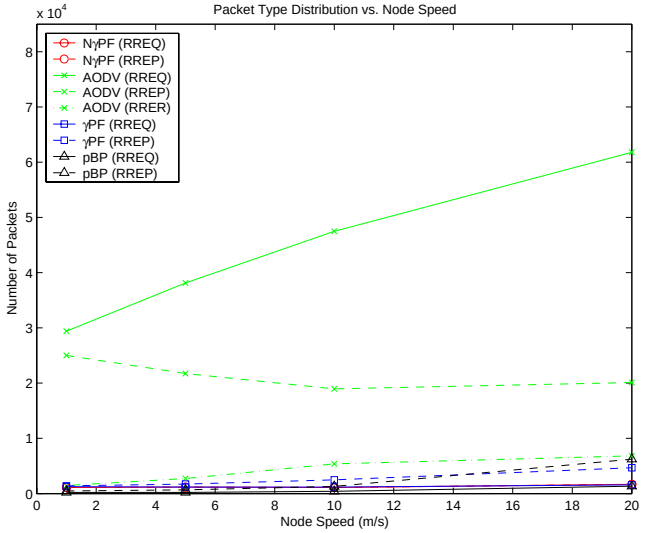

Fig. 3. Packet Type Distribution vs. Node Speed

Medium Load

While the control overhead statistics are quite positive, it is ultimately necessary to compare the total amount of access to the medium needed to deliver a packet. Figure 4 shows that Termite is able to do better than AODV with regards to the total number of transmissions required to deliver a packet suc-

cessfully. They perform equally at 20 m/s. Both algorithms show an increasing load on the medium as the network volatility increases. In the case of AODV, this is because of the increasingly large amount of control traffic generated. For Termite, this is because data packets must take longer paths to explore the network as the topology changes faster.

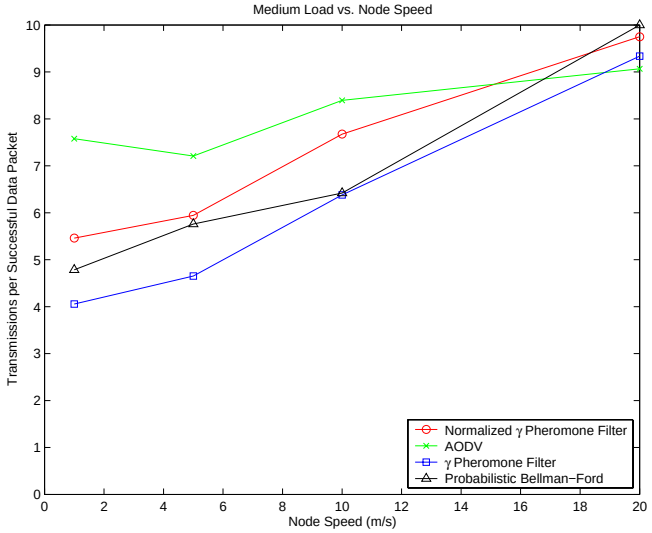

Fig. 4. Medium Load vs. Node Speed

Medium Efficiency

The results of Figure 5 show that Termite is able to easily deliver packets at low speeds. Both AODV and Termite expend more effort to deliver a packet successfully, again with similar performance at 20 m/s. Medium efficiency decreases linearly with node speed in all cases. This metric reconfirms the results of the medium load.

Medium Inefficiency

This metric complements the results of the previous two. Figure 6 shows that Termite makes a best effort to deliver all packets that are offered. It is able to do so with fewer packet transmissions at low speed but suffers at higher speed by a margin of 13%. The reason for this is due to unsuccessful data packets that wander the network until they exceed their Time-To-Live (TTL). This behavior causes unnecessary transmissions which increases the metric. Naturally this happens more at high speeds when the network topology is more volatile. When the medium inefficiency is readjusted to include only successful

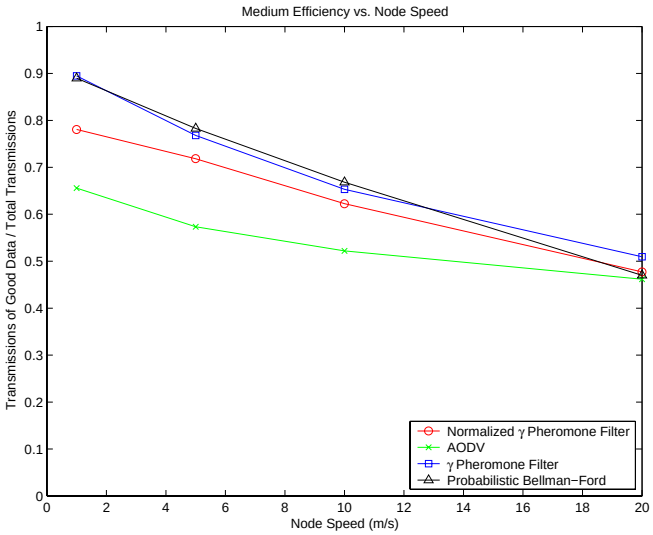

Fig. 5. Medium Efficiency vs. Node Speed

data packets (the medium load in Figure 4), the outcome is somewhat more even.

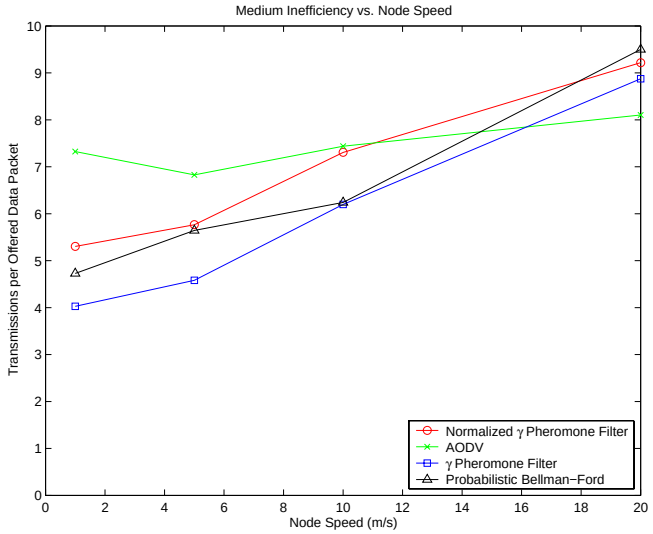

Fig. 6. Medium Inefficiency vs. Node Speed

Link Failure Rate

Figure 7 shows how the link failure rate changes with node speed. As expected from a mathematical analysis of link lifetime in [35], this trend is linear. The results may be somewhat skewed from standard link failure rates because this data is only measured when a communications attempt fails. That is, when a packet is sent on a link but the receiver is unavailable. Since the packet rate is relatively low, this measured link failure rate may also tend towards the low end. However, this metric does give an idea as to how quickly the network is changing and at what rate the nodes should reconfigure themselves.

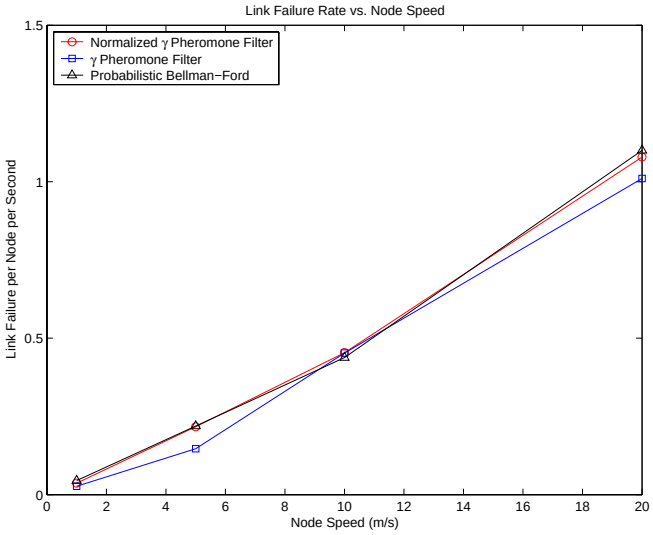

Fig. 7. Link Failure Rate vs. Node Speed

End-To-End Delay

Figure 8 shows the average end-to-end (ETE) delay for each of the compared routing algorithms. The ETE delay of AODV stays constant regardless of node speed while the delay of the Termite variants grows from less than AODV at low speed to substantially larger at high speed. The AODV results reflect those reported by [33] and is due to the fact that packets are only sent when a full route is known to exist. The Termite delay results are not positive considering that the delay of AODV is nearly an order of magnitude lower at high speeds. However, there are two extraneous issues at work in this situation. The first is that the metric only measures the delay of successful packets. AODV has a lower goodput than Termite, and so packets that might have otherwise timed out in AODV due to link or route discovery failure are delivered by Termite.

This extra goodput comes at the expense of some additional delay incurred by perhaps a longer path length or congestion. It can be shown that when the slowest 5% of packets in Termite are not considered (the difference of goodputs between AODV and Termite at 20 m/s), then the delay can be reduced by 66%. This property indicates that Termite's high delay comes from statistical outliers. These are packets that require a great deal of effort to deliver, packets that AODV does not. The second factor is Termite's use of link recovery retransmission. When the network is changing quickly, packets may be retransmitted often due to neighbor loss. If several nodes in the same area are trying to retransmit, this can lead to localized packet storms. This is especially true when using 802.11 which automatically retries up to seven times before reporting a broken link.

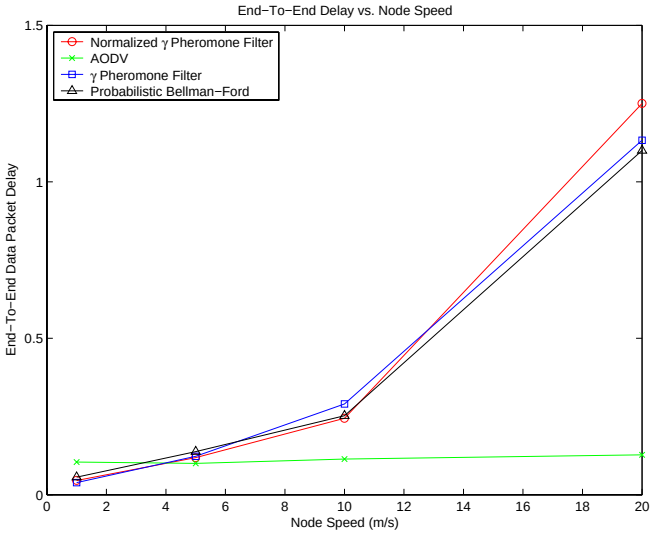

Fig. 8. End-To-End Delay vs. Node Speed

Discussion

The results show that Termite is able to outperform AODV primarily due to the lack of control traffic and effectively liberal route caching. AODV often finds itself repairing routes which requires route errors and route request floods. Termite avoids this complexity and effort through the use of piggybacked route information and promiscuous packet reception. However, Termite must find a way to explore the network as well in order to find better routes. It does this by letting data packets do the work. The medium load, efficiency, and inefficiency metrics show what the control overhead does not.

The Termite approach works well at low speed but gives fewer performance gains at high speed. The number of packet transmissions per data packet is equal between the two algorithms; they both do the same amount of work to deliver packets. The reason for this is that Termite relies on packet traffic to update routing tables, and this traffic can move erratically over the network it is changing often or little traffic is available. A proactive update procedure such as that found in AdHocNet may be helpful. Termite need the path samples in a timely fashion in order to make good routing decisions. The most noticeable difference between the algorithms at high speed is the end-to-end delay. Termite's packet storms hurt performance tremendously.

3 Towards An Optimal Pheromone Decay Rate Heuristic

Termite has several parameters which influence its performance. This section derives a heuristic for the optimal pheromone decay rate, the decay rate which maximizes the performance of Termite, and compares it to simulation data. The heuristic is based only on the correlation time of the network, or how long the network stays relatively the same. This is measured in part by average link lifetime. Additional influences on performance are not considered in the heuristic at this time. If the selected decay rate is larger than the optimal, $\tau > \tau^*$, then the network will forget its state too fast and throw away relevant information. If the decay rate is too low, $\tau < \tau^*$, then the network will retain too much information and also make suboptimal decisions.

3.1 The Model

A simple model of the network is first introduced in order to lay an analytical framework for the heuristic. The network is modeled as two communicating nodes with two independent paths available between them [36]. These paths abstract all other connections between the two nodes, including additional nodes, mobility issues, or communications effects. The physical structure is shown in Figure 9, and is the same as that used in [2].

Each node sends packets to the other with independent exponentially distributed interarrival times. The average rate at which node A sends packets to B is λ_A, and λ_B in the opposite direction. Each node decays the pheromone on its links independently. The decay rates at each node are τ_A and τ_B, respectively.

Each path, indexed by v, has a utility characterized by a non-negative random process $\Gamma_v(t)$ with mean $\mu_v(t)$. The pheromone contained in a packet arriving on a link, γ, is a sample of that process. Γ is non-stationary since link utilities change over time due to mobility and other effects, including the fact that the passage of each packet will change local routing probabilities due to pheromone update. Since each packet moving in the same direction passes through the network independently of all other packets, there is no correlation

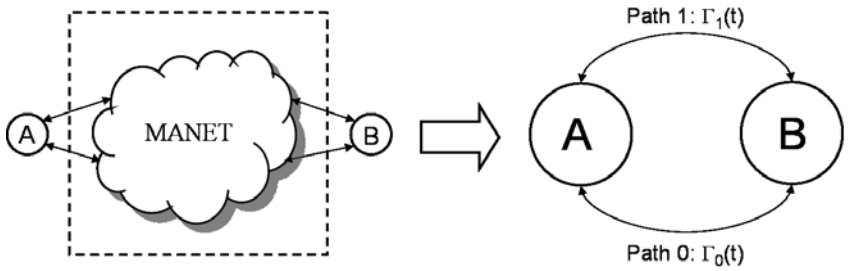

Fig. 9. Diagram of the MANET Model

between successive samples of the link utility process. The forwarding equation independently considers each packet.

3.2 Optimal Mean Estimation of $\Gamma(t)$

$\Gamma^n_{r,s}(t)$ is a non-stationary stochastic process which models the received pheromone at node n on the link to neighbor r from source s. Because the process is non-stationary, the traditional set of stochastic analysis tools are not helpful. It is assumed that the process may be modeled as stationary over some finite period of time called the correlation time, T seconds. Under such circumstances, and keeping in mind that each received sample of $\Gamma^n_{r,s}(t)$ within T is assumed to be independent and identically distributed, the optimal mean estimator is a simple box filter of a length sufficient to include all and only those samples recieved within the last T seconds [37]. While the accuracy of the resulting estimation is dependant on the number of samples received and thus may vary, there is no better estimator.

3.3 Suboptimal Mean Estimation of $\Gamma(t)$

The box filter is an unwieldy approach. The filter length must be continuously updated in order to account for the number of received packets, and all of those pheromone values must be maintained. The mean estimation thus involves possibly a large number of addition operations as well as a division. An estimator requiring less state and computation is desired, though it may be suboptimal.

The biological inspiration for Termite holds the answer. A one-tap infinite impulse response filter is used in place of the optimal box filter, such as γPF or $\bar{\gamma}$PF. The pheromone decay rate can be adjusted in order to account for (changes in) the correlation time of Γ. A heuristic for the optimal pheromone decay rate for the γ filters is developed based on these ideas and then compared to simulation results.

3.4 The τ^* Heuristic

The γ pheromone filter and the normalized γ pheromone filter update rules are repeated in Equations 8 and 9 for reference.

$$P^n_{r,s} \leftarrow P^n_{r,s} e^{-\left(t-t^n_{r,s}\right)\tau} + \gamma \tag{8}$$

$$P^n_{r,s} \leftarrow P^n_{r,s} e^{-\left(t-t^n_{r,s}\right)\tau} + \gamma \left[1 - e^{-\left(t-t^n_{r,s}\right)\tau}\right] \tag{9}$$

The heuristic is developed by examining the correlation between successive estimates of $E\Gamma(t)$, which is the pheromone resident on a link, $P(t)$, and adjusting the pheromone decay rate in order to minimize any correlation beyond the correlation time of the pheromone process. The current pheromone on a link may be generalized as in Equation 10.

$$P(t) = \Gamma(t) * h(t) \tag{10}$$

The function $h(t)$ is the impulse response of the estimation filter and the $*$ operator is convolution. The equivalency is shown for both of the pheromone filters in Equations 11 and 12.

$$h_{\gamma PF}(t) = e^{-\left(t-t^n_{r,s}\right)\tau} u(t) \tag{11}$$

$$h_{\bar{\gamma} PF}(t) = (1-\beta)\, e^{-\left(t-t^n_{r,s}\right)\tau} u(t) \tag{12}$$

The constant $(1-\beta)$ in Equation 12 is a normalization constant and $u(t)$ is the unit step function. The normalization constant for $h_{\bar{\gamma} PF}(t)$ is derived in [36]. For the purposes of this derivation the constant is unimportant.

The time correlation of the output of the estimation filter, which is the pheromone, can be found based on traditional techniques of statistical signal processing theory [37]. This includes finding the spectral density of $P(t)$. It is important to note in this case that individual samples of $\Gamma(t)$ are independent, since each packet is routed independently. It is also assumed that samples received within a period T of each other are identically distributed. The spectral density of $\Gamma(t)$ is thus flat. The correlation of either of the pheromone filters is shown in Equation 13.

$$R_{P_{PF}}(\Delta t) = e^{-|\Delta t|\tau} \tag{13}$$

The heuristic for the optimal value of τ, τ^*, can now be determined by finding its required value for the correlation to drop below a threshold at the correlation time. Equation 14 formalizes this requirement, where $c \ll 1$ is the threshold.

$$\tau^*(T) : R_{P_{PF}}(T) = e^{-T\tau^*} = c \tag{14}$$

The optimal pheromone decay rate is computed and is shown in Equation 15.

$$\tau^*(T) = -\frac{\ln c}{T} \tag{15}$$

3.5 A Return to the Box Filter

It was previously stated that a variable length box filter would be the optimal link utility estimator for this application. The filter should average all of the pheromone samples within the correlation time. It remains to be noted what the correlation structure is of the resulting estimate, such that the forwarding equation can properly account for time uncertainty in the estimate. Following the previous method used to calculate the correlation of the metric estimate, Equation 16 shows the result for the box filter.

$$R_{box}\left(\Delta t\right) = \begin{cases} 1 - \frac{|\Delta t|}{T} & , \quad |\Delta t| \leq T \\ 0 & , \quad |\Delta t| > T \end{cases} \tag{16}$$

Since the box filter has only a finite length in time, T, estimates more than T seconds apart have no correlation. In the context of metric estimation, since no node maintains information about the network more than T seconds old, the estimator cannot provide any reliable information about the network at that time; the pheromone is all gone.

3.6 Generalization of the Forwarding Equation

When using general estimators, correlation functions must be added to the forwarding equation to account for uncertainty in the estimate introduced by time. Different estimators will have different correlations between successive estimates as seen above. The generalized forwarding equation is shown in Equation 17.

$$p_{i,d}^{n} = \frac{\left[P_{i,d}^{n} R\left(t - t_{i,d}^{n}\right) + K\right]^{F}}{\sum_{j \in \mathcal{N}^{n}} \left[P_{j,d}^{n} R\left(t - t_{j,d}^{n}\right) + K\right]^{F}} \tag{17}$$

Note that when a pheromone filter is used, the pheromone decay model is recovered exactly. Otherwise the forwarding framework represents a utility estimator, taking into account time uncertainty in the estimator.

3.7 τ^* Simulation

Figures 10 and 11 show how the performance of Termite can vary according to the global pheromone decay rate. The parameter was kept constant in the previous simulations at $\tau = 2.0$ in order to make each scenario directly comparable. The figures test τ over two orders of magnitude at node speeds of 1 m/s and 10 m/s. As is shown, the appropriate selection of this parameter is of critical importance. Using the normalized γ pheromone filter at 1 m/s, $\tau \approx 0.1$ is the best choice. $\tau \approx 1.0$ is best at 10 m/s. This is most easily seen from the achieved goodput and medium efficiency. The performance of Termite was quite good in the first set of simulations, and these results show

that the performance could be even better if more appropriate parameters are chosen. The parameters of the original simulations are held constant for the purposes of fair comparison. Otherwise, the parameter space of Termite is so large that it can be difficult to choose those giving the best performance in a given environment.

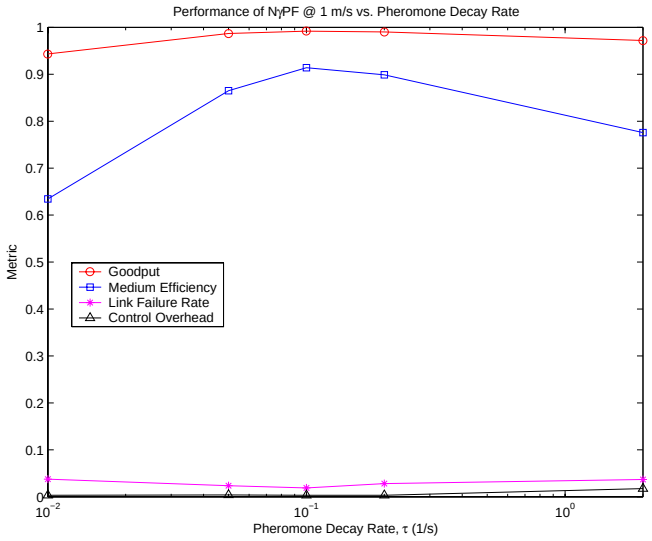

Fig. 10. Performance of NγPF @ 1 m/s vs. Pheromone Decay Rate, τ

3.8 τ^* Heuristic Analysis

The quality of the heuristic is determined by comparing its predictions to experimental data. The primary difficulty is to determine a good metric for the network correlation time (T) or the network event rate (T^{-1}). This work will use the link failure rate as a lower bound for the event rate (and thus an upper bound on the correlation time). Figure 12 takes the results reported in Figures 10 and 11 and compares them to the τ^* heuristic. A correlation threshold of $c = 0.1$ is used to calculate the heuristic.

The heuristic and the measurements match quite well. The simplicity of the heuristic allows the possibility to dynamically compute the optimal pheromone decay rate locally for each node, or even each link at each node. This would represent a departure from biological possibility, however it also would improve the performance of the system, as seen from a purely mathematical perspective. Unfortunately it is not trivial to generate an optimal decay rate in a rigorous way based only on samples of $\Gamma(t)$. A large number of calculations are generally required in addition to having a great deal of data on hand.

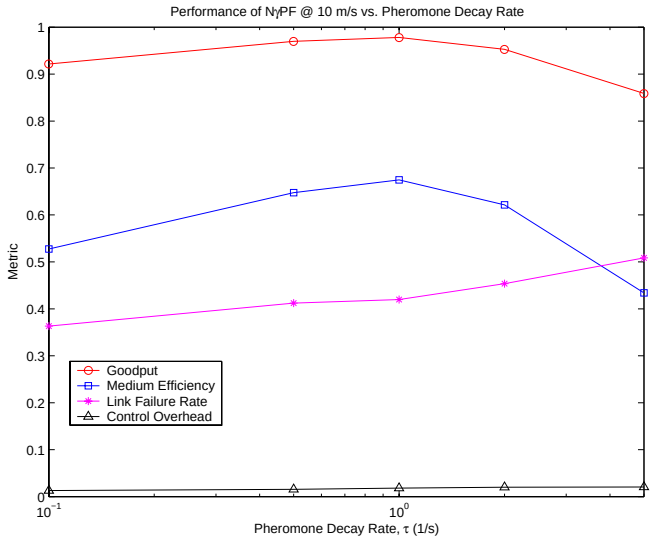

Fig. 11. Performance of NγPF @ 10 m/s vs. Pheromone Decay Rate, τ

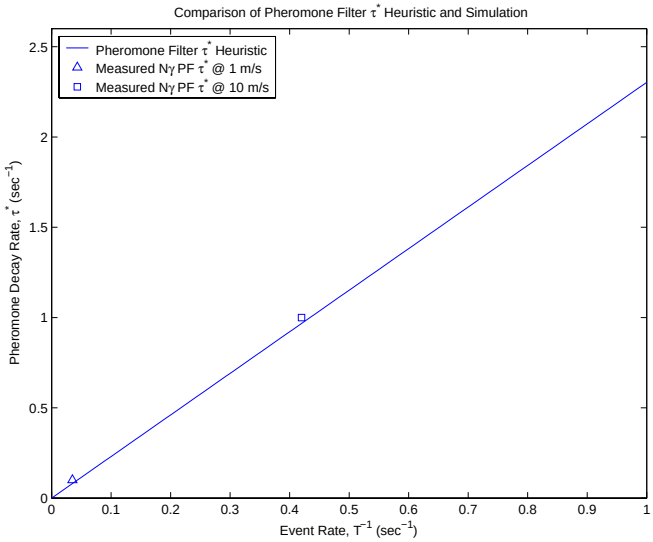

Fig. 12. Comparison of Pheromone Filter τ^* Heuristic and Simulation

Perhaps a heuristic could be generated which relates more easily to observed network parameters, such as the link lifetime (or link change rate measured here), to τ^*, as has been done here. There is the caveat however, that the link change rate is itself dependant on the decay rate whereas the network correlation time imagined in this work is an independent parameter. The link

change rate is dependant because τ affects routing probabilities, which affects what neighbors are used, which affects how often those links are tested for connectivity, which affects the amount of time necessary to detect that a link has broken. The network correlation time is instead dependant on the parameters affecting the metric that the network is optimizing for. In this paper the path length is considered; the parameters controlling the underlying mobility model must be examined. But it is again unfortunate that these parameters, such as node speed, may not be readily available in an implementation.

3.9 Full Circle - A Return to Traditional Routing

Termite is a biologically inspired algorithm, but the final algorithm is closely related to well-known distance vector solutions. Traditional implementations update packets with the best known routing metric to its source from a node, instead of letting packets carry their actual path metric as in Termite. This difference is easily reconciled, and is formalized in the context of Termite in Equation 18.

$$\gamma \leftarrow \max_{i \in \mathcal{N}^n} P_{i,s}^n R\left(t - t_{i,s}^n\right) \tag{18}$$

The correlation term is included in order to properly account for uncertainty introduced by time, at each link. This update of a packet's pheromone occurs immediately before a packet is retransmitted to the next hop; after the packet pheromone is first updated with Equation 4, and after the pheromone table update.

4 Conclusion

4.1 Review

A swarm intelligent routing algorithm named Termite has been presented and its performance evaluated. Termite is an advancement of previous work with its addition of source pheromone aversion, a pheromone decay rate heuristic, and a generalized forwarding equation. A comparison to the standard AODV routing protocol, based on simulation studies, showed the general superiority of Termite. One reason for this superiority lies in the effectiveness of using data to carry routing information, reducing control overhead and maintaining routing information. However, it was seen that the end-to-end delay performance suffers due to the fraction of packets which are difficult to deliver. This may be due to frequent route breaks, route discovery latencies, and other network effects. Ultimately, Termite is shown to deliver more packets with less overhead in more adverse conditions than AODV in a realistic medium access environment.

A heuristic for an optimal pheromone decay rate was developed based on stochastic process theory. The problem was restated to make the question of

pheromone update a question of optimal estimation. Optimal parameters were shown to exist through simulation, and the heuristic fits quite nicely with the results. It remains an open question of which independent parameters should be used to determine the optimal estimation parameters.

4.2 Future Work

Future work in this area can address a number of open issues. This work has investigated an optimal decay rate heuristic, thought it is still unclear how the other parameters which control the operation of Termite should be set or adjusted. Parameters of interest include the pheromone sensitivity, F, the pheromone threshold, K, and the source aversion sensitivity, A. These parameters could also conceivably be adjusted on a per node per link basis, though it is unclear if such an approach is possible, or even useful.

The results presented here show that Termite can find "acceptable" routing solutions "fast enough." It is still unknown how fast these solutions are reached, and what parameters help shape the solution. The development of a temporal model of SI routing could help to answer such questions, and also help to shed light on their ability to control the behavior of the network.

A discussion of parameter selection and temporal dynamics reveals a more general question regarding the control of networks. If every parameter of the routing algorithm can conceivably be adjusted in real time according the network dynamics, then the entire behavior of the network could perhaps also be controlled, regardless of the demands placed on it. As probabilistic routing is more easily mathematically analyzed than deterministic routing with non-linear rules, using such a framework may allow further insights into the control of large, distributed, dynamic systems.

References

1. E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.
2. 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.
3. N. Meuleau, M. Dorigo, *Ant Colony Optimization and Stochastic Gradient Descent*, *Artificial Life 8*, 2002.
4. C. Perkins, E. Belding-Royer, S. Das, *Ad-hoc On-demand Distance Vector (AODV) Routing*, http://moment.cs.ucsb.edu/AODV/aodv.html, 2004.
5. R. Bellman, *On a Routing Problem*, Quarterly of Applied Mathematics, 1958.
6. L. Ford Jr., D. Fulkerson, *Maximal Flow Through a Network*, Canadian Journal of Mathematics, 1956.
7. E. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik, Vol. 1, 269-271, 1959.

8. C. Perkins, P. Bhagwat, *Routing over Multihop Wireless Network of Mobile Computers*, SIGCOMM '94, 1994.
9. P. Jacquet, P. Mühlenthaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot, *Optimized Link State Routing Protocol for Ad-Hoc Network*, IEEE INMIC, 2001.
10. D. Johnson, D. Maltz, *Dynamic Source Routing in Ad-hoc Wireless Networks*, SIGCOMM '96, 1996.
11. C. Siva Ram Murthy, B. S. Manoj, *Ad Hoc Wireless Networks: Architectures and Protocols*, Prentice Hall PTR, 2004.
12. C. Perkins, E. Belding-Royer, S. Das, *Ad-hoc On-demand Distance Vector (AODV) Routing*, http://www.faqs.org/rfcs/rfc3561.html, 2003.
13. P. Perkins, E. Royer, *Ad-hoc On-demand Distance Vector*, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999.
14. B. Holldobler, E. O. Wilson, *The Ants*, Belknap Press, 1990.
15. A. MacKenzie, S. Wicker, *A Repeated Game Approach to Distributed Control in Wireless Data Networks*, 2003.
16. L. Buttyan, J.-P. Hubaux, *Enforcing Service Availability in Mobile Ad-Hoc WANs*, First Annual Workshop on Mobile and Ad Hoc Networking and Computing, 2000.
17. M. Resnick, *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*, Bradford Books, 1997.
18. R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, *Ant-Based Load Balancing In Telecommunications Networks*, *Adaptive Behavior*, 1996.
19. G. Di Caro, M. Dorigo, *Mobile Agents for Adaptive Routing*,Technical Report, IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.
20. 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.
21. 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.
22. M. Günes, U. Sorges, I. Bouazizi, *ARA - The Ant-Colony Based Routing Algorithm for MANETs*, Proceedings of the ICPP Workshop on Ad Hoc Networks, 2002.
23. 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.
24. M. Heissenbüttel, T. Braun, *Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks*, Kommunikation in Verteilten Systemen (KiVS), 2003.
25. M. Roth, S. Wicker, *Termite: Ad-hoc Neworking with Stigmergy*, The Second Mediterranean Workshop on Ad-Hoc Networks, 2003.
26. S. Rajagopalan, C. Shen, *A Routing Suite for Mobile Ad-hoc Networks using Swarm Intelligence*, unpublished, 2004.
27. G. Di Caro, F. Ducatelle, L. M. Gambardella, *AdHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad-Hoc Networks*, Technical Report No. IDSIA-27-04-2004, 2004.
28. 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.

29. G. Di Caro, F. Ducatelle, L.M. Gambardella, *Swarm Intelligence for Routing in Mobile Ad-Hoc Networks*, IEEE Swarm Intelligence Symposium, 2005.
30. 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.
31. G. Varela, M. Sinclair, *Ant Colony Optimization for Virtual-Wavelength-Path Routing and Wavelength Allocation*, Proceedings of the 1999 Congress on Evolutionary Computation, 1999.
32. L. Li, Z. Haas, J. Halpern, *Gossip-Based Ad-hoc Routing*, The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2002.
33. C. Perkins, E. Royer, S. Das, M. Marina, *Performance Comparison of Two On-Demand Routing Protocols for Ad-Hoc Networks*, IEEE Personal Communications, February 2001.
34. http://www.opnet.com/
35. P. Samar, Z. Haas, *On the Behavior of Communication Links of a Node in a Multi-Hop Mobile Environment*, The Fifth ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc), 2004.
36. 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.
37. P. J. Brockwell, R. A. Davis, *Introduction to Time Series and Forecasting, Second Edition*, Spring-Verlag, 2002.