The Virtual Pheromone Communication Primitive

Leo Szumel and John D. Owens

University of California at Davis
Department of Electrical and Computer Engineering
One Shields Ave, Davis, CA 95616, USA
{lpszumel,jowens}@ucdavis.edu

Abstract. We propose a generic communication primitive designed for sensor networks. Our primitive hides details of network communication while retaining sufficient programmer control over the communication behavior of an application; it is designed to ease the burden of writing application-specific communication protocols for efficient, long-lived, fault-tolerant, and scalable applications. While classical network communication methods expect high-reliability links, our primitive works well in highly unreliable environments without needing to detect and prune unreliable links. Our primitive resembles the chemical markers used by many biological systems to solve distributed problems (pheromones). We develop and analyze the performance of an implementation of this primitive called Virtual Pheromone (VP). We demonstrate that VP can attain performance comparable to classical methods for applications such as sleep scheduling, routing, flooding, and cluster formation.
This is a minor revision of the paper included in the Proceedings of the Second International Conference on Distributed Computing in Sensor Sytems that includes an additional reference to related work in Section 2.3.

1 Introduction

Most wireless sensor network (WSN) and ad-hoc networking applications demand *efficiency*, *long life*, *fault-tolerance*, and *scalability*. We refer to these as ELFS applications. The goal of this paper is to demonstrate that Virtual Pheromone (VP) is an effective tool for building ELFS applications.

It is commonly accepted that cross-layer design is necessary in order to achieve the levels of efficiency desired in most WSN and energy-constrained ad-hoc network applications [1, 2]. Eschewing the classical network layered abstraction model (OSI) enables advancements in energy efficiency. This *energy* efficiency comes at the cost of *programmer* efficiency, because integration and debugging of these applications can be very complex. It is desirable to have mechanisms that balance the utility of abstraction against the flexibility of cross-layer design. The sensor network field has entered an era of consolidation wherein point solutions are being generalized to create low-level abstractions that are useful across many applications; TinyOS 2.0 and UCLA's Tenet are examples [3, 4]. In short, some energy efficiency must be traded for generality and ease of programming if sensor networks are to become ubiquitous.

1.1 Motivation

Sensor networks bring computer and network technology in closer entanglement with the natural world than ever before. Assumptions made in classical computing no longer hold: communications are unreliable, nodes are unreliable, a deterministic mapping of the system state may be unattainable, and energy is a finite resource. The combination of a lossy and random environment means that the behavior of a sensor network cannot be perfectly determined; rather, we must be satisfied with specific behavioral qualities or bounds on expected performance.

Because ELFS are bound so tightly with the physical world, it stands to reason that natural communication mechanisms may provide insight into the design of artificial communication mechanisms appropriate for that environment. Many species use pheromones (scent signals) to communicate information and organize behavior in complex and challenging environments. For instance, ants successfully forage for food and build nests in spite of continually changing physical parameters—paths are blocked, hazards are presented, food sources come and go.

Natural systems are dynamic in the relationship between actors (or agents) and the environment in which they reside. In a data-collection wireless sensor network, the sources of dynamism are radio interaction and node failure. When a node is asleep, it cannot participate in multi-hop communication, and when packets are sent, they may collide with other transmitters' packets. In more complex scenarios, e.g. mobile or sensor-actuator networks, dynamism is increased along more parameters. We argue that pheromone-inspired communications can be useful in the simple case and are very desirable in the complex case.

1.2 Contributions

Because communication cost is expected to dominate sensor network energy costs, we have designed a communication primitive that is designed to approach the efficiency of applications that are specifically tailored to their tasks. Our goal is to trade a minimal amount of energy efficiency for increased programmer efficiency and code reuse. Moreover, the nature of our communication primitive leads to efficient, scalable application design, because every transmission is recognized as a broadcast. Efficiency is chiefly provided by a single-layer abstraction: simulated diffusion of virtual pheromone signals. This abstraction benefits from the presence of unpredictable links, rather than requiring that they are pruned out. Finally, by using a common primitive for all (or most) communication, a powerful optimization point is created such that enhancements to the primitive can benefit a large set of applications.

2 Overview

In the biological sense, a pheromone is a chemical marker. It can be deposited by an organism and detected by that and/or other organisms, e.g. for the direction of a male moth to a female [5]. A pheromone dissipates via an evaporative process; as a result the strength of the pheromone decays with increasing distance from the source in the

spatial and temporal axes. Spatial decay maintains locality of communication; temporal decay reduces system complexity by ensuring that old information is purged from the system.

2.1 Interesting Properties

Why is a spatio-temporally decaying information process interesting in the context of sensor networks? Let us examine the properties of this information process that are in common with many sensor network applications. **Spatially-limited sharing** of information: sensor networks can only scale if internode information sharing is limited by some process. **Encoding of distance** to source: hop counts, for instance, are commonly used as a cost metric in routing algorithms. **Encoding of time** since deposition: the relevance of information is crucial in allowing nodes to collaborate with each other; there is a significant difference between a sensor reading from one minute ago and a reading from one month ago. **Superposition** of like pheromones: akin to aggregation by sum; allows reinforcement. **Implicit gradient** leading to source: gradients enable *efficient* and *scalable* routing algorithms because the routing information is stored in a distributed fashion.

The net result of these properties is that all information exchange is via constrained broadcast through a shared medium. This maps very well to the behavior of a radio communication system.

2.2 The VP Communication Primitive

VP also exhibits the five aforementioned properties, each of which serves a purpose in the design of an ELFS application. Efficiency is driven by low-cost transmission of information; VP requires no ACK signals and uses an efficient flooding technique (Section 3.1). Fault-tolerance requires a level of redundancy and robustness to errors; VP's inherent redundancy provides a tradeoff of reliability against cost, and because no explicit point-to-point communication is used, information transfer is highly tolerant of node failure (as long as the network remains unpartitioned). Application scalability requires highly localized communication and efficient coordination amongst neighboring nodes; VP pheromones propagate proportional to the strength of the initial deposit, which creates a user-selectable bound on distance. The encoding of distance and time in the pheromone strength, combined with superposition of like pheromones, allows neighboring nodes to coordinate without requiring expensive point-to-point protocols that are sensitive to faults.

2.3 Related work

The topic of communication methods in wireless networks is broad; we reference representative works from two important categories: localized communication/control primitives and applications using pheromone-like concepts.

In directed diffusion [6], information sinks (consumers) publish *interests* which are propagated through parts of the network; information sources (producers) publish

named data objects which are routed along the interest gradient. Geolocation is assumed so that interests can be distributed locally. Directed diffusion is an application facilitating data transfer between sinks and sources and works best when a flow of information will pass from a source to one or more sinks. In contrast, VP is a lower-level primitive intended for many communication tasks (including diffusion-like routing; see Section 4.2).

RUGGED [7] is a routing protocol that utilizes *data gradients*, or "fingerprints," rather than the interest gradients created by directed diffusion. Fingerprints are disseminated by an environmental processes—not by the sensor network—and this creates a possible efficiency benefit. Simulated annealing techniques are used to overcome local minima or maxima in the sensor field, including regions in which the sensed level is zero. Fingerprint routing is targeted specifically at data-collection applications in which natural gradients exist in the phenomena to be measured.

In his master's thesis [8], McLurkin develops a similar pheromone primitive for sensor networks. His pheromone is closer to a true diffusion process, with nodes periodically sharing pheromone state and cooperatively decaying values (VP distributes information once and decay occurs as a distributed process). Significantly, the design and evaluation of VP is efficiency-driven, whereas McLurkin's concentration was on exploring inter-node cooperation and the structural primitives that can be constructed with pheromones.

Payton et al. explore using "virtual pheromones" [9] for robotic control. The pheromones are sent via infrared or other line-of-sight communication method; gradient descent necessarily indicates an unobstructed path that the robots may use. Characterization of the infrared source and its spatial decay is used as a distance estimator. Parunak et al. use a pheromone-inspired memory model [10] to assist in the coordination of distributed decision-making systems. Brooks et al. use biologically-, chemically-, and physically-inspired techniques (including one based on pheromones) in sensor network adaptation techniques [11]. They find these techniques to be very robust to errors while attaining satisfactory power consumption.

2.4 Qualitative Expectations

It is instructive to consider the expected behavior of VP, as compared to the classical primitive of point-to-point transmission, before delving into technical analysis. We expect VP to work well in dense networks since the signal can propagate many hops. We expect VP to perform less well when used, naively, to implement classical protocols that utilize handshaking, acknowledgements, or other point-to-point information exchanges. VP should be highly tolerant of node failure and spurious communication because it benefits from the redundancy of broadcast and can utilize unreliable links.

Furthermore, due to the large time constants involved in pheromone decay, we expect VP to be most useful in latency-insensitive applications. While fields can propagate quickly, the rate of accomplishing a specific action-reaction pair using pheromones is not likely to exceed a classical communication approach. For instance, any application that leverages the superposition principle of pheromone signaling will take some time to reach a steady state response.

We begin by discussing the parameters of a pheromone communication primitive, and the details our specific implementation, in Section 3. The metrics relevant to ELFS are presented in Section 4 and then applied to four application scenarios and analyzed in Section 4.2. Section 5 describes the future direction of this research, and Section 6 concludes the paper.

3 Design

We desire to implement a mechanism displaying the properties listed in Section 2.1: encoding of space and time, spatially limited flooding, superposition, and gradient generation. Energy efficiency constrains our implementation options. We analyze the design problem as the following set of sub-problems:

Programmer API. How should a program deposit and detect pheromones? What information should be provided, and in what form? Our goal is to minimize the amount of code required to handle communications, yet provide enough flexibility to make the primitive useful.

We believe deposition should encode, at a minimum, the following fields: TYPE, STRENGTH, and SOURCE. PAYLOAD (arbitrary data) should be an optional parameter. Reading a pheromone should provide at least the first two fields: TYPE and STRENGTH. SOURCE may be required for some applications. The arbitrary payload field could be used as a way to encode complex data, such as a non-integer data type, or an uniquifier.

Spatial Dissipation Model. The desired behavior is to have the strength of the field decay with increasing distance from the pheromone source. This encodes physical distance from the sender and limits the range over which data will propagate. Candidate dependent variables for this decay include hop count, RSSI, or geolocation.

Temporal Dissipation Model. The desired behavior is to have the strength of the field decay with increasing time since the deposition of the pheromone. Exponential decay is ideal because it favors recent information while allowing old information to persist at a low level. Half-life may be global, indepdenent for each pheromone, or chosen from a small set (e.g., {fast-decay, slow-decay}), depending on application needs.

Pheromone Encoding & Transmission Strategy. The transmission strategy of the pheromone information can have a large impact on the energy efficiency of the operation. If maximum propagation speed is desired, a packet must be sent for each deposition. If the constraint of propagation speed is relaxed, packet overhead may be amortized over several pheromone depositions.

3.1 VP: A Design Implementation

VP encodes TYPE, STRENGTH, SOURCE, and PAYLOAD in a table and provides two functions to store (and implicitely transmit) and retrieve pheromone signals: DEPOSIT

and SMELL. Utility functions to support common uses of pheromones will reduce program size and be useful across applications. To facilitate the applications in Section 4.2, we have implemented a function to forward a packet along a pheromone gradient (FWDGRADIENT), and a function to return a list of distinct sources for a pheromone type (SMELLDISTINCT).

DEPOSIT(TYPE, STRENGTH, PAYLOAD): Deposit a pheromone identified by TYPE with strength equal to STRENGTH; PAYLOAD is optional. The subsystem decides if the information should be propagated to neighboring nodes according to the rules in Section 3.1.

SMELL(TYPE): Return the net strength of the pheromone(s) matching TYPE.

SMELLDISTINCT(TYPE): Return a list of each distinct pheromone detectable at this node. This bypasses superposition and includes the payload of each pheromone (if present).

FWDGRADIENT(PACKET, TYPE, DIRECTION): Forward a packet according to the gradient of pheromone TYPE; DIRECTION may be *uphill*, *downhill*, or *level*.

Storage and Retrieval of Pheromone State. The internal data structure is a $4 \times N$ table (Table 1). To support superposition, each unique type/source pair may have its own entry in the table, but when the pheromone is read all rows with a common type are summed together. Payload is suppressed when using SMELL but may be read using SMELLDISTINCT.

Table 1. A simple data structure for storing pheromones. The strength level associated with type a from source i is denoted l_{ai}. For example, SMELL(a) = $l_{ai} + l_{aj}$ and SMELL(b) = l_{bk}. Payload is labeled equivalently to strength.

type	source	strength	payload
t_a	s_i	l_{ai}	p_{ai}
t_a	s_j	l_{aj}	p_{aj}
t_b	s_k	l_{bk}	p_{bk}

Deposition and Propagation of Pheromones. When a pheromone is deposited with strength s, the table is first checked for a hit on (TYPE,SOURCE). There are three possible cases:

1. If there is no hit, the data is stored and then scheduled for transmission as a pheromone update *with strength* $(s-1)$.
2. If the hit has a strength less than s, the "no hit" action (1) is taken.
3. If the hit has a strength $\geq s$, the deposition is ignored.

This algorithm sets up a cone-shaped field *prior to to initiation of the decay process* (Section 3.1). The initial field has a slope of one pheromone unit per hop; as a result, distance is derived from radio hops.

Broadcast Suppression Technique. Pheromone updates attempt to limit redundant transmissions. A depositing node that wants to send an update moves through the phases listed in Table 2. While *observing*, the node snoops for and accumulates the count of concurrent depositions by neighboring nodes. In the *transmitting* phase, the snoop count is compared to a threshold, η. If this threshold is not exceeded, the node transmits the pheromone; in either case, it calculates the expected pheromone distribution time and waits for the propagation to complete. This delay equals the pheromone strength at this node, s_i, times the expected observation delay at each hop (including packet transmit time); this algorithm forms a schedule for initiation of the distributed decay process. The *decaying* phase lasts until the pheromone reaches a minimum threshold, ε (Section 3.1).

Table 2. A pheromone deposition is a distributed process including three phases at every node *i*: observation (for suppression), transmission (or suppression) and a pause, and then decay. t_{tx} is the expected transmit time, s_i is pheromone strength at node *i*, and τ is a constant.

phase	duration
observing	$t_{\text{obs}} = [0, \tau]$
transmitting (or suppressing)	$(t_{\text{obs}} + t_{\text{tx}})s_i$
decaying	until pheromone is depleted

Time-Decay of Pheromones. All nodes concurrently run a decay process on their pheromone tables. The decay process is a discretized approximation of continuous exponential decay, updated every τ seconds: $s(t + \tau) = \alpha s(t)$. The process terminates when $s(t) < \varepsilon$, where α is constant, τ is the update interval, ε is the termination threshold, and $s(0)$ is the initial strength at this node as defined in Section 3.1.

VP maintains an event list large enough to keep one "next decay time" for each pheromone. For large scale systems with many pheromones, it would be more appropriate to have one global timer that decays all pheromones; this may require a smaller τ (Section 3.1).

Constants. We use the following constants in our implementation of VP: half-life = 10 s, τ = 100 ms, ε = 0.1, t_{tx} = 2 ms, and $\eta = 2$. t_{tx} is derived from a 250 kbps radio and 40-byte packet (1.28 ms to transmit, plus processing time). η is chosen to optimize efficiency (we tried values 1,2,3,4)—it must be tuned to the RF model (discussion in Section 5).

Memory Requirements. We divide the pheromone information into two categories: *user* and *system* (Table 3). Parameter precision may be adjusted to application requirements. Each pheromone requires 7–14 bytes of information (before compression) to transmit; a packet payload of 32 bytes can contain at least 2-4 pheromones.

Table 3. memory requirements to store a pheromone (bytes). The *next decay time* is not transmitted. *System* variables are used to propagate and decay the signal, and include *user* variables, which are exposed by the API. The total footprint of each pheromone is between 7 and 14 bytes.

space		parameter	bytes
system		source ID	2–4
·		spatial metric	1–2
·		next decay time	1–2
·	user	type	2–4
·	·	strength	1–2

4 Metrics & Evaluation of VP

Because VP is targeted at ELFS applications, we choose one metric for each ELFS quality:

Efficiency: the communication cost, in *packets sent to accomplish a task*. All else being equal, a task that requires less communication is more efficient.

Long-livedness: the lifetime of the network, in *seconds-until-dysfunction*. This is differentiated from efficiency because application requirements, such as coverage, depend on both node death distribution and the fault-tolerance of the algorithm. It is reasonable to expect, however, that an efficient algorithm will also be long-lived.

Fault-tolerance: the lifetime, as *long-livedness* is observed with these varying system parameters: *network density, random node failure probability, and partial network occlusion*.

Scalability: the *long-livedness* is observed when the network is scaled in *node placement density*, in nodes per unit area.

4.1 Experimental Methodology

Showcasing all possible applications of our primitive is beyond the scope of this paper; rather, we aim to discuss several applications and tasks that can be accomplished using our primitive and to provide evidence of satisfactory performance.

Our experiments are performed using a custom simulator called AHLPS. We also verify functionality with physical deployment on 25 Telos motes using a less complete implementation of VP. Whereas implementation on motes proves the functionality of the primitive, only simulation allows us to explore VP in the environment it was designed for: large scale networks.

We previously developed the Agent High-Level Pythonic Simulator (AHLPS) to allow simulation of agent behavior in large sensor networks [12]. A pheromone primitive was added to AHLPS to support the research in this paper. AHLPS uses the TOSSIM empirical radio link model [13] to simulate link quality[1] but does not simulate a PHY

[1] While the TOSSIM model is based on empirical data ranging from 0 to 40 meters, we have scaled this data so that 1 distance unit under AHLPS is equivalent to 40 meters under TOSSIM.

layer or complex MAC layer. As a result, radio contention is not modeled; this is acceptable in our simulations because the mean communication rate is low (generally less than a packet per second per node). AHLPS allows us to investigate, at a high level, the behavior of an agent-based program. Further verification is then performed in TOSSIM and/or on physical nodes using our agent framework [14].

4.2 Case Studies

We will now use results on the behavior of representative tasks and applications to verify that VP supports the requirements of ELFS: efficiency, long-livedness, fault-tolerance, and scalability. Any algorithm implemented using VP is referred to as "Pheromone."

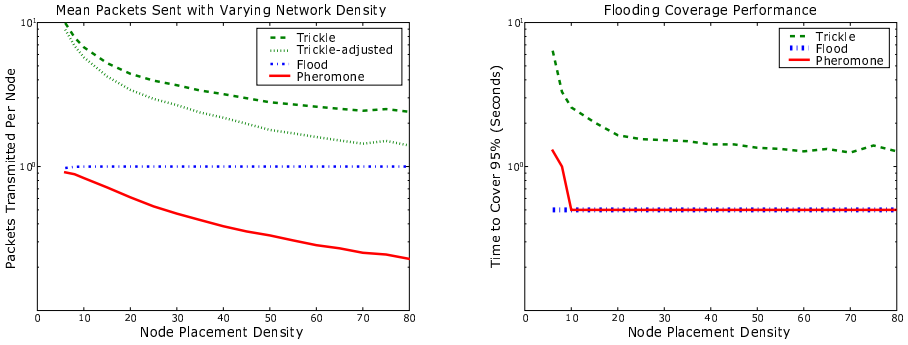
One way to measure efficiency is to take the ratio of a cost (generally energy) and a benefit (generally network lifetime). Unfortunately, results of this measure are specific to each application. We desire to separate out an *indicator* of efficiency that will imply efficiency performance for many applications. Because most sensor networks communicate some shared information, we analyze the efficiency of distributing information using three methods: flooding, epidemic routing, and VP.

Fault-tolerance is also very application-specific because of the specificity of the term "fault." We attempt to pick two common fault scenarios that many networks expect to incur: a long-period disruption to a large section of the network, and intermittent dropouts of specific nodes. Because it is impossible to generalize all applications' susceptibility to these specific faults, we choose a basic measure: the ability to route information from one part of the network to another. This embodies two concepts: inter-node collaboration, and network connectivity; most sensor network applications require both.

Efficiency, fault-tolerance, and scalability are requisites for long-livedness, and analyzed separately. Our aim is to show that VP enables longevity extension in a way applicable to many applications. We chose to examine cover-constrained sleep scheduling because it can be used in many applications to extend lifetime. While not a proof of generality, such an example provides evidence for our argument.

Scalability is extremely important in many sensor network designs. Unfortunately, few truly scalable networks have been deployed in the field. We are developing a multi-hop clustering algorithm that we believe will be crucial in the scalable operation of very large networks, allowing collaboration and organization of large (but constrained) subsets of nodes. Multi-hop clustering has been mostly a footnote in the literature, although it does resemble the "Multiple Sink Network Design Problem" [15] posed by Oyman and Ersoy. Our goal is to show that our multi-hop clustering task scales perfectly using VP.

Efficiency. Flooding is the most basic form of dissemination. Epidemic algorithms, such as Trickle [16], perform efficiency optimizations using suppression. We compare the use of VP to disseminate a unit of information to a network of 1000 nodes using naive Flood and Trickle. Comparison is on the basis of packets sent, per node, per unit of information, with the independent variable of network density (Figure 1). The simulation is run for 20 seconds but statistics are collected at the earliest time for which *cover* (fraction of nodes having received the information unit) is 95% or higher.



(a) (b)

Fig. 1. Performance of Trickle, Flood, and Pheromone in information dissemination. Both Trickle and Pheromone benefit from increased network density. Pheromone is bounded above by both Trickle and Flood in terms of efficiency, while matching Flood's rapid speed.

Flood has a constant cost because all nodes participate in every flood event. Trickle benefits from increasing network density because more nodes can be suppressed; however, there is an unavoidable baseline cost in order to run the algorithm's periodic probes. Increasing the maximum probing interval would decrease this cost but also decrease the responsiveness of the system. Because each run is begun from scratch, we also plot "Trickle–adjusted," in which every node is given one "free" packet to perform the initial round of probes.

There are two reasons that Pheromone performs better than Trickle in this test. First, there is the aforementioned base cost. Second, Trickle's probe-response protocol means that, in the limit of extreme density, distributing a unit of information would require at least three transmissions: broadcast a new version, broadcast a suppressing data request, broadcast a data response. Pheromone would require only a minimum of two transmissions: broadcast of pheromone, and one suppressing re-broadcast.

The flooding coverage chart indicates that none of the algorithms perform satisfactorily below a density of 10. It is important to note that Trickle *guarantees* 100% cover in a connected network while Flood and Pheromone, with their probabalistic approach, attain 95-100% coverage. In addition, the lack of contention in the AHLPS radio model gives advantage to Flood and Pheromone's latency performance. Pheromone and Trickle serve different purposes, but it is clear that Pheromone is an efficient dissemination mechanism.

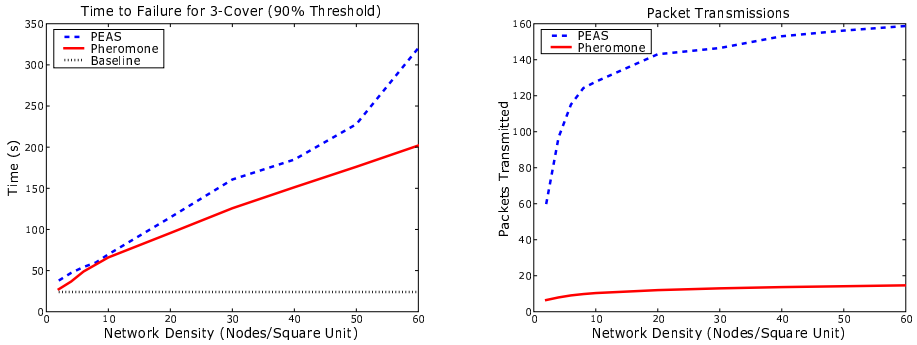
Memory and computational resources are not modeled in AHLPS. We note that all three algorithms are computationally simple, with Pheromone being the most complex because our current implementation uses a floating point multiplication (Section 3.1). The memory requirement for Pheromone is slightly greater due to the SOURCE param-

eter. (STRENGTH in Pheromone is balanced by HOPCOUNT in Flood and VERSION in Trickle.)

Long-Livedness. A well-referenced and effective sleep scheduling algorithm is PEAS [17]. We implemented PEAS under AHLPS as well as our own scheduler that uses VP. We use a metric from the PEAS paper: *time that 90% 3-cover is maintained*. *k-cover* is defined as the fraction of network area observed by at least k nodes. We use the same operating constants as Ye et al.: idle power 12 mW, sleep power 30 μW, transmit energy 600 μJ per packet, and receive energy 120 μJ per packet.

Pheromone has a single tunable parameter: THRESH. A single pheromone is sent as a beacon by all awakened nodes; if the perceived level at a given node is greater than THRESH, the node will go to sleep with probability proportional to the difference. PEAS has two parameters, λ, the mean rate at which nodes will wake up and probe for neighbors, and R_p, the probing range. We set $\lambda = 1$ Hz and $R_p = 1.0$ (the maximum radio range).

Sensing range is set to 2.0, or 2 times the maximum radio range. Since most good links are at a distance ≤ 0.2, the sensing range is significantly larger than the probing range. 3-cover failure for three or more seconds is considered a failure. We set THRESH $= 1.0$ to match PEAS' behavior (a node sleeps if any probe responses are heard). The fundamental difference between Pheromone and PEAS is that the former is probing (broadcast announcements) while the latter is polling (broadcast request, unicast response).



(a) (b)

Fig. 2. (a) PEAS is highly effective at extending network lifetime. Pheromone is nearly as effective for lower densities but loses ground as density is scaled. (b) PEAS has a radio overhead nearly 10 times greater than Pheromone.

Figure 2 shows that While PEAS is very effective (a), it requires a high overhead in transmissions (b) because each broadcast probe is followed by multiple unicast responses. PEAS offers an additional benefit that Pheromone does not: the pursuit of λ

probing rate. (Effectively, "λ" is fixed in Pheromone.) Pheromone offers performance comparable to PEAS at a much lower transmission overhead, indicating it would perform very well in applications with a higher transmit-to-idle cost ratio. Perhaps the biggest benefit of Pheromone is that it is easily integrated into a cross-layer algorithm design.

Fault-Tolerance. We wish to examine fault models sufficiently simple to avoid loss of generality in the results. We examine two models: *partial network occlusion* and *random network dropouts*. During an occlusion, a large portion of the network becomes unreachable (e.g. due to a signal jammer) between $t = [50, 150]$ s. During a dropout, nodes may become unreachable for a brief period (e.g., fast fading); the dropout rate is λ and the dropout interval is exponentially distributed with mean 1 s. Network size is 1000 nodes.

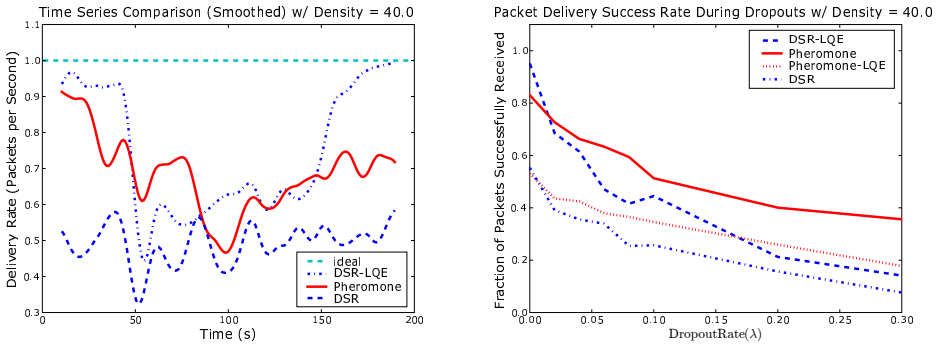
These fault models are applied to a partial implementation of DSR [18]. DSR is an on-demand point-to-point routing algorithm designed for ad-hoc networks. Because the DSR specification is complex, we implemented in AHLPS only the features involved in robust delivery from a single source to a single sink. While the DSR specification allows for asymmetric data/ACK paths, it does not specify a mechanism, so our implementation uses only symmetric paths.

We compare DSR to gradient ascent routing on a pheromone field. Packets are transmitted once per second and in the case of Pheromone, a pheromone is deposited at the sink once every 10 seconds. There are no tunable parameters for Pheromone. A significant difference is that Pheromone will follow multiple simultaneous paths when routing on a gradient (all uphill paths are followed), whereas DSR will discover parallel paths and then use just one at a time. Analysis of packet overhead (not presented here) indicates that Pheromone has a lower packet overhead for the data presented here.

Because AHLPS uses an empirical radio model, nodes have many neighbors at the fringe of connectivity and with correspondingly high loss rates. DSR does not perform well under this regime because it discovers, and attempts to use, faulty paths; it must explore many of these before finding a quality path. To alleviate this problem, we implemented an omniscient, zero-overhead, link-quality estimator (LQE) that permits only high-quality links. High quality is defined as an expected round-trip loss rate of at most 1%. Pheromone is permitted to run under both the empirical and LQE modes for comparison, but performs better without LQE because of the increased redundancy.

We see in Figure 3 that DSR outperforms Pheromone with the LQE feature (a) but is inferior without it. Random dropouts (b) affect DSR more than Pheromone because DSR must explore new routes more often as λ is increased. Performance without LQE is important because in the real world LQE is not free, especially in mobile or time-fading environments.

Scalability. Our aim in this section is to demonstrate that a scalable application can be constructed from our primitive. We chose multi-hop clustering and implemented two versions using only VP for communication: *generic* and *adaptive*. Generic is based loosely on LEACH [19], in which a cluster head probability, p_{ch}, controls cluster head formation. Adaptive effectively adapts p_{ch} distributedly at each node. Both algorithms



(a) (b)

Fig. 3. DSR versus Pheromone gradient routing (a) during an occlusion event and (b) with varying dropout rate (λ). $\lambda = 0$ corresponds to no dropouts. Density is 40.0 nodes per unit area in both cases and LQE indicates an omniscient, zero-overhead, link-quality estimator. Pheromone does not require LQE, an important efficiency benefit, and is far more tolerant to random node dropouts. DSR, on the other hand, performs much better when LQE is available.

use a pheromone deposit to form clusters—the cluster head deposits a pheromone and nodes choose cluster membership based on observed pheromone strengths.

We consider the algorithm complete when the *mean cluster size* has stabilized within 98% of its final value. The scalability of both algorithms is apparent in Figure 4; density does not affect the time to completion. Both algorithms are scalable because of pheromones' suppression mechanism—in a network twice as dense, twice as many nodes will be suppressed during pheromone deposition.

Adaptive and Generic both stabilize at about $t = 40$ s and have similar mean transmission rates. Adaptive has a higher *peak* transmission rate, which may be a disadvantage in some applications. Given the similar performance of the two algorithms, we feel that Adaptive is superior because tuning of p_{ch} is not required. The Adaptive algorithm is presented in code listings 1.

5 Future Work

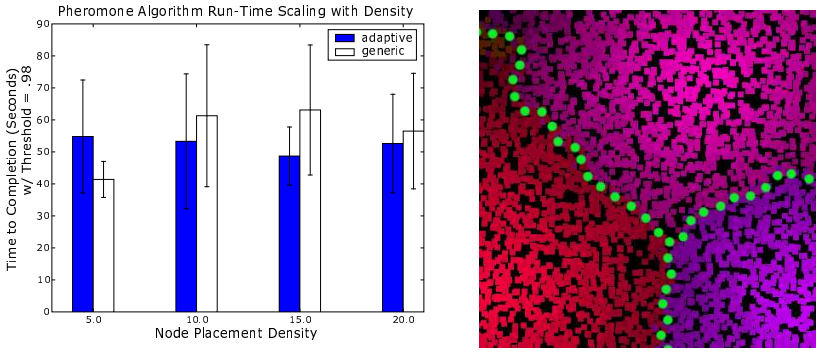
In future work it would be beneficial to examine the constants used in pheromone distribution and decay. Optimal η, the suppression constant, is likely to depend on the specific MAC and PHY. Higher η means a greater packet overhead and potential for collisions. Setting η too low will result in a lack of redundancy. In our experiments, one global pheromone decay half-life was sufficient. We would like to explore the benefit of allowing different half-life settings. Finally, local repair of the pheromone field (e.g. when a node is added or wakes up from sleep), as in that proposed by Han et al. [20], would drastically improve response time at a small overhead cost.

Algorithm 1 The behavior of a cluster member. All nodes initially take this behavior and can change to cluster head (Algorithm 2) if no cluster head pheromones are present.

```
1:  scents ⇐ SmellDistinct('clustering_pheromone')
2:  if scents is empty then
3:     WAIT for snoop interval
4:     behavior ⇐ ClusterHead
5:  else
6:     if two strongest scents are of equal strength then
7:        my_membership ⇐ EdgeMember
8:     else
9:        my_membership ⇐ CellMember
10:    end if
11: end if
```

Algorithm 2 The behavior of a cluster head. Note that behavior can change to cluster member (Algorithm 1) if this cluster head loses an instant-runoff based on random pheromone IDs (Line 5). The runoff avoids mutual simultaneous annihilation of cluster heads.

```
1:  my_strength ⇐ 10 {10 hops}
2:  my_ID ⇐ random integer
3:  scents ⇐ SmellDistinct('clustering_pheromone')
4:  if detect one or more cluster heads in scents then
5:     if my scent random ID < strongest scent's random ID then
6:        behavior = ClusterMember
7:     end if
8:  else
9:     Deposit(type='clustering-pheromone', strength=my_strength, payload=my_ID)
10: end if
```



(a) (b)

Fig. 4. (a) Clustering performance of two pheromone algorithms as a network of 4000 nodes is scaled in density. Time to complete cluster formation is independent of density. (b) A snapshot of cluster formation at $t = 19.0$ s using the Adaptive algorithm; lightness indicates pheromone strength (green dots added to emphasize cell edges).

The VP API is designed around superposition but applications which utilize PAYLOAD need to read unique pheromones. Currently we uniquify the fields using PAYLOAD, but this can also be done by encoding a unique number into TYPE. The algorithms in our case studies could be further simplified by offering common operations, e.g.: "give me the *n* strongest pheromones matching this type", or "give me the payload of the strongest pheromone of this type." The design and selection of these operations is the topic of future work.

6 Conclusion

We have shown that VP addresses the needs of ELFS applications: efficiency, long-life, fault-tolerance, and scalability. We compare our Pheromone algorithms to existing, well-known point solutions for the following problems: dissemination, sleep scheduling, and routing. In addition, a novel scalable clustering application is examined. In every case, algorithms using VP attain comparable performance because they leverage the abundance of lossy links in the RF environment rather than trying to avoid them. Simultaneously, VP algorithms are simple to program and rely on a spartan API, which creates a powerful common optimization point.

The key contribution of VP is that it builds a simple conceptual interface to the radio that is consistent across a broad range of system parameters (e.g. density, node dropout rate). Applications built using VP can be fault-tolerant without having to re-implement custom error control mechanisms or having to rely on link quality estimators.

Our experience using VP is that applications must think about every communication as a broadcast, and this encourages the programmer to utilize that fact. The programs we developed are small (10–30 lines of Python code) and thus easier to understand. The use

of parameter "tuning" can make optimization difficult; in our opinion it is imperative to design *adaptive* algorithms such as the clustering presented in Section 4.2.

References

1. Levis, P., Madden, S., Gay, D., Polastre, J., Szewczyk, R., Woo, A., Brewer, E., Culler, D.: The emergence of networking abstractions and techniques in TinyOS. In: Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation. (2004) 1–14
2. Goldsmith, A., Wicker, S.B.: Design challenges for energy-constrained ad hoc wireless networks. IEEE wireless communications **9**(4) (2002) 8–27
3. Levis, P., Gay, D., Handziski, V., Hauer, J.H., Greenstein, B., Turon, M., Hui, J., Klues, K., Sharp, C., Szewczyk, R., Polastre, J., Buonadonna, P., L.Nachman, G.Tolle, D.Culler, A.Wolisz: T2: A second generation OS for embedded sensor networks. Technical Report TKN-05-007, Telecommunication Networks Group, Technische Universität Berlin (2005)
4. Govindan, R., Kohler, E., Estrin, D., Bian, F., Chintalapudi, K., Gnawali, O., Rangwala, S., Gummadi, R., Stathopoulos, T.: Tenet: An architecture for tiered embedded networks. Technical Report 56, Center for Embedded Networked Sensing (2005)
5. Kuwana, Y., Shimoyama, I., Sayama, Y., Miura, H.: Synthesis of pheromone-oriented emergent behavior of a silkworm moth. In: Proceedings of the International Conference on Intelligent Robots and Systems. Volume 3. (1996) 1722–1729
6. Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, Washington, USA, ACM (1999) 263–270
7. Faruque, J., Helmy, A.: RUGGED: routing on fingerprint gradients in sensor networks. In: The IEEE/ACS International Conference on Pervasive Services. (2004) 179–188
8. McLurkin, J.D.: Algorithms for distributed sensor networks. Master's thesis, University of California at Berkeley (1999)
9. Payton, D., Daily, M., Estowski, R., Howard, M., Lee, C.: Pheromone robotics. In: Autonomous Robots. (2001) 319–324
10. Parunak, H.V.D., Brueckner, S.A., Matthews, R., Sauter, J.: Pheromone learning for self-organizing agents. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans **35**(3) (2005)
11. Brooks, R., Pirretti, M., Zhu, M., Iyengar, S.: Distributed adaptation methods for wireless sensor networks. In: IEEE Global Telecommunications Conference. Volume 5. (2003) 2967–2971
12. Szumel, L.: The agent high-level pythonic simulator. (2006) Work in progress.
13. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In: Proceedings of the First ACM Conference on Embedded Networked Sensor Systems. (2003) 126–137
14. Szumel, L., LeBrun, J., Owens, J.D.: Towards a mobile agent framework for sensor networks. In: Second IEEE Workshop on Embedded Networked Sensors. (2005) 79–87
15. Oyman, E.I., Ersoy, C.: Multiple sink network design problem in large scale wireless sensor networks. In: Proceedings of the International Conference on Communications, Paris, France (2004)
16. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: Proceedings of First Symposium on Networked Systems Design and Implementation, San Francisco, CA (2004)

17. Ye, F., Zhong, G., Lu, S., Zhang, L.: PEAS: A robust energy conserving protocol for long-lived sensor networks. In: 3rd International Conference on Distributed Computing Systems. (2003)
18. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In Imielinski, Korth, eds.: Mobile Computing. Volume 353. Kluwer Academic Publishers (1996)
19. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocols for wireless microsensor networks. In: International Conference on System Sciences, Maui, HI (2000) 3005–3014
20. Han, K.H., Ko, Y.B., Kim, J.H.: A novel gradient approach for efficient data dissemination in wireless sensor networks. In: Proceedings of the International Conference on Vehicular Technology (VTC). (2004)