Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks

Chalermek Intanagonwiwat
Ramesh Govindan
Deborah Estrin

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292, USA

Abstract

Advances in processor, memory and radio technology will enable small and cheap nodes capable of sensing, communication and computation. Networks of such nodes can coordinate to perform distributed sensing of environmental phenomena. In this paper, we explore the *directed diffusion* paradigm for such coordination. Directed diffusion is data-centric in that all communication is for named data. All nodes in a directed diffusion-based network are application-aware. This enables diffusion to achieve energy savings by selecting empirically good paths and by caching and processing data in-network. We explore and evaluate the use of directed diffusion for a simple remote-surveillance sensor network.

1 Introduction

In the near future, advances in processor, memory and radio technology will enable small and cheap nodes capable of wireless communication and significant computation. The addition of sensing capability to such devices will make distributed microsensing—an activity in which a collection of nodes coordinate to achieve a larger sensing task—possible. Such technology can revolutionize information gathering and processing in many situations. Large scale, dynamically changing, and robust *sensor networks* can be deployed in inhospitable physical environments such as remote geographic regions or toxic urban locations. They will also enable low maintenance sensing in more benign, but less accessible, environments: large industrial plants, aircraft interiors etc.

To motivate our research, consider this simplified model of how such a sensor network will work (we refine this model in Section 2). One or more human operators pose, to any node in the network, questions of the form: "How many pedestrians do you observe in the geographical region X?", or "Tell me in what direction that vehicle in region Y is moving". These queries result in sensors within the specified region being *tasked* to start collecting information. Once individual nodes detect pedestrians or vehicle movements, they might collaborate with neighboring nodes to disambiguate pedestrian location or vehicle movement direction. One of these nodes might then report the result back to the human operator.

Motivated by robustness, scaling, and energy efficiency requirements, this paper examines a new data dissemination paradigm for such sensor networks. This paradigm, which we call *directed diffusion*, is data-centric. Data generated by sensor nodes is named by attribute-value pairs. A node requests data by sending *interests* for named data. Data matching the interest is then "drawn" down towards that node. Intermediate nodes can cache, or transform data, and may direct interests based on previously cached data.

Using this communication paradigm, our example might be implemented as follows. The human operator's query would be transformed into an interest that is *diffused* towards nodes in regions X or Y. When a node in that region receives an interest, it activates its sensors which begin collecting information about pedestrians. When the sensors report the presence of pedestrians, this information returns along the reverse path of interest propagation. Intermediate nodes might *aggregate* the data, *e.g.,* more accurately pinpoint the pedestrian's location by combining reports from several sensors. An important feature of directed diffusion is that interest and data propagation and aggregation are determined by *local processing rules* that do not rely on topology-dependent metrics.

Directed diffusion is significantly different from IP-style communication where nodes are identified by their end-points, and inter-node communication is layered on an end-to-end delivery service provided within the network. In this paper, we describe directed diffusion and illustrate one instantiation of this paradigm for sensor query dissemination and processing. We show that using directed diffusion one can realize robust multi-path delivery, empirically adapt to a small subset of network paths, and achieve significant energy savings when intermediate nodes aggregate responses to queries.

In this paper, we first outline sensor capabilities and distributed microsensing in general (Section 2). We then outline the directed diffusion paradigm, and explain its key features (Section 3). That section also describes in some detail a particular instantiation of the directed diffusion paradigm for an animal tracking sensor network. We specify what local rules achieve the desired behavior of interest and data propagation. In doing so, we show how the directed diffusion paradigm differs from traditional networking, and qualitatively argue that this paradigm offers scaling, robustness and energy efficiency benefits. We quantify some of these benefits via detailed packet-level simulation of directed diffusion (Section 4).

2 Sensor Networks and Distributed Sensing

In this section, we first describe the capabilities we expect in future sensor nodes. We then argue that these capabilities imply a particular organization of future sensor networks. This organization, one that resembles a self-configuring ad-hoc network, is motivated by signal processing and energy efficiency considerations.

2.1 Sensors and Sensor Nodes

Before we can understand the future capabilities of sensor networks, we must postulate the capabilities of future sensor nodes. It is not unreasonable to expect the following features in a future sensor node: A matchbox sized form factor, battery power source, an power-conserving processor clocked at several hundred Mhz, program and data memory amounting to several tens of MBytes, a radio modem that employs some form of diversity coding [10], and an energy efficient MAC layer based on, for example, TDMA [21]. As such, this node would be capable of running a possibly stripped-down version of a modern operating system; examples of such operating systems include Windows CE and μCLinux.

Such a node could have one or more sensors. Examples of such sensors include seismic geophones, infrared dipoles and electret microphones for acoustic sensing. The analog-to-digital conversion system on such nodes might produce upto 70 ksamples per second, at upto 12 bit resolution. For reasons of power conservation, some of the common signal processing functions may be offloaded into a low-power ASIC. In this way, the main processor need be woken up only when events of interest occur. Finally, a sensor node may possess a fully-functional GPS receiver.

2.2 Sensor Networks

Because of their compact form factor and their potential low cost, it might be possible for a densely—within tens of feet of each other—packed cluster of such sensor nodes to be deployed, in a possibly unplanned fashion, *near* the phenomena to be sensed—*e.g.,* at busy intersections, or in the interior of large machinery. The advantage of such sensor networks is that, even with relatively cheap sensors, these nodes can obtain high SNR (given that the signal generated by any physical phenomena rapidly attenuates with distance). Furthermore, given the spatial density of these deployments, an individual sensor node may *not* have to frequently perform multi-target resolution (i.e. distinguish between different targets such as individuals and vehicles). Such multi-target resolution can involve complex deconvolution algorithms requiring non-trivial processing capability [20].

By contrast, today's sensor deployments fall into two categories. Large, complex sensor systems are usually deployed very far away from the phenomena to be sensed, and employ complex signal processing algorithms to separate targets from environmental noise. Alternately, a carefully engineered network of sensors is deployed in the field, but individual sensors do not possess computation capability, instead transmitting *time series* of the sensed phenomena to one or more nodes which perform the data reduction and filtering.

Should future sensor networks resemble sensor deployments of old? In particular, should sensor nodes transmit time series of data to some central node which performs the target resolution? One key consideration in sensor networks—energy efficiency—dictates otherwise. Because sensors are likely to be battery-powered, and because sensor networks will be expected to have lifetimes of several days (with possibly prolonged lulls in activity), conserving battery resources is a crucial requirement. This means that short-range hop-by-hop communication is preferred over direct long-range communication to the destination. Coincidentally, such hop-by-hop communication also provides a form of communication diversity in helping communicate around obstacles [20]. Energy efficiency also implies that it is infeasible to transmit time-series data across the network, even hop-by-hop. As [20] shows, performing local computation to reduce data before transmission can obtain orders of magnitude energy savings.

These energy efficiency considerations, coupled with the likely availability of processing power and communication capability in sensor nodes, argues for a different organization of a sensor network. In this organization, individual nodes reduce the sampled waveform generated by a target (*e.g.,* a pedestrian or a vehicle) into a relatively coarse-grained "event" description. This description usually contains a "codebook value"—an event code—for the target, a timestamp, a signal amplitude, and a degree of confidence in the estimate. Nodes can then exchange these event descriptions with their neighbors—who are also likely to have observed the target—to refine the estimation, transmitting only a short description back to a human operator.

Informally, with such an organization, a sensor network begins to look like a distributed computing system. What communication primitives can be employed in such unattended sensor networks? While it is not infeasible to design these sensor networks using IP and ad-hoc routing, the central thesis of this paper is that a different set of communication primitives can lead to more efficient sensor data dissemination.

2.3 Communication for Sensor Networks

Consider a simple sensor network for remote surveillance of a region. In practice, such a network might consist of several hundreds or thousands of sensor nodes deployed within that region. In some cases, the sensor *field* may be deployed in a regular fashion (e.g. a 2-dimensional lattice, or a linear array) within that region. More generally, however, communication and networking protocols cannot assume structured sensor fields.

A *user* of this remote surveillance network would be able to contact (using, perhaps a long-range radio link) one of the sensors in the field, and pose the following *task*: "Every I ms for the next T seconds, send me a location estimate of any four-legged animal in subregion R of the sensor field". In general, the network may support a variety of task types. However, sensor networks are *task-specific*—unlike general purpose communication networks, the task types are known at the time the sensor network is deployed[1]. We leverage this important observation in our design.

Using hop-by-hop wireless communication and routing mechanisms described in Section 3, this task is conveyed to sensor nodes in the subregion R of the sensor field. Each node then tasks its sensors to collect samples, and matches the sampled waveform against a locally-stored library. If the node detects a waveform typical of a four-legged animal, it generates F event descriptions a second, each of which contains the following items: its own location, a codebook value corresponding to the animal, the intensity of the signal, and a degree of confidence in its estimation. Sensors within region R may coordinate to pick the best estimate. This estimate "packet" is then routed back towards the task originator.

The focus of this paper is the design of dissemination mechanisms for tasks and events. We describe this dissemination mechanism in the context of the sensor network described above, but with support for multiple concurrent task initiations of the type specified above. We later argue that our overall approach, *directed diffusion*, applies more generally to other kinds of distributed sensor coordination. The challenges that we face in designing these mechanisms are:

Scale These mechanisms must scale to several thousands of sensor nodes in the sensor field.

Robustness Sensor nodes may fail, may lose battery power, or may be temporarily unable to communicate due to environmental factors. The dissemination mechanisms must be robust to such failures.

Energy efficiency As described above, wireless communication even over relatively short distances consumes significant energy. The dissemination mechanisms must minimize energy usage.

[1] More accurately, sensor networks may be reprogrammable and the tasks they support may change slowly over time.

3 Directed Diffusion

Directed diffusion consists of several elements. Data is *named* using attribute-value pairs. A sensing task (or a subtask thereof) is disseminated throughout the sensor network as an *interest* for named data. This dissemination sets up *gradients* within the network designed to "draw" events (*i.e.,* data matching the interest). Events start flowing towards the originators of interests along multiple paths. The sensor network *reinforces* one, or a small number of these paths. Figure 1 illustrates these elements.

In this section, we describe these elements of diffusion with specific reference to a particular kind of sensor network—one that supports the task described in Section 2.3. Such a network performs location tracking. As we shall see, several design choices present themselves even in the context of this specific instantiation of diffusion. We elaborate on these design choices while describing the design of our sensor network. Our initial evaluation (Section 4) focuses only a subset of these design choices.

3.1 Naming

In directed diffusion, task descriptions are *named* by, for example, a list of attribute-value pairs that describe a task. The animal tracking task described in Section 2.3 might be described as (this is a simplified description, see Section 3.2 for more details):

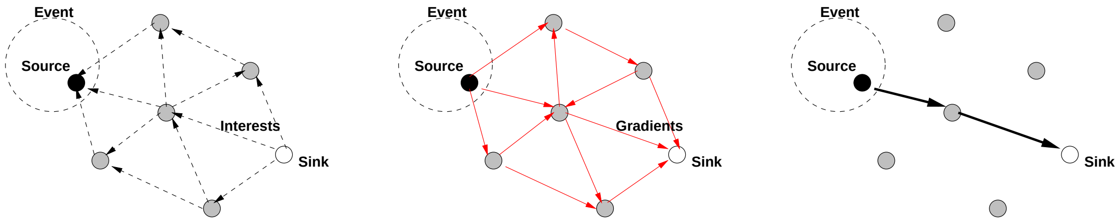
```
type = four-legged animal    // detect animal location
interval = 20 ms             // send back events every 20 ms
duration = 10 seconds        // .. for the next 10 seconds
rect = [-100, 100, 200, 400] // from sensor nodes within rectangle
```

For ease of exposition, we choose the subregion representation to be a rectangle defined on some coordinate system; in practice, this might be based on GPS coordinates.

Intuitively, the task description specifies an interest for data matching the attributes. For this reason, such a task description is called an *interest.* The data sent in response to interests are also named using a similar naming scheme. Thus, for example, a sensor that detects an animal might generate the following data (see Section 3.3 for an explanation of some of these attributes):

```
type = four-legged animal // type of animal seen
instance = elephant       // instance of this type
location = [125, 220]     // node location
intensity = 0.6           // signal amplitude measure
confidence = 0.85         // confidence in the match
timestamp = 01:20:40      // event generation time
```

Given a set of tasks supported by a sensor network, then, selecting a naming scheme is the first step in designing directed diffusion for the network. For our sensor network, we have chosen a simple attribute-value based interest and data naming scheme. In general, each attribute has an associated value range. For example, the range of the `type`



(a) Interest propagation **(b)** Initial gradients set up **(c)** Data delivery along reinforced path

Figure 1: A simplified schematic for directed diffusion.

attribute is the set of codebook values representing mobile objects (vehicles, animal, humans). The value of an attribute can be any subset of its range. In our example, the value of the `type` attribute in the interest is that corresponding to four-legged animals.

There are other choices for attribute value ranges (*e.g.,* hierarchical) and other naming schemes (such as intentional names [1]). To some extent, the choice of naming scheme can affect the expressivity of tasks, and may impact performance of a diffusion algorithm. In this paper, our goal is to gain an initial understanding of the diffusion paradigm. For this reason, we defer the exploration of possible naming schemes to future work.

3.2 Interests and Gradients

The named task description of Section 3.1 constitutes an *interest*. An interest is usually injected into the network at some (possibly arbitrary) node in the network. We use the term *sink* to denote this node.

Given our choice of naming scheme, we now describe how interests are *diffused* through the sensor network. Suppose that a task, with a specified `type` and `rect`, a `duration` of 10 minutes and an `interval` of 10ms, is instantiated at a particular node in the network. The `interval` parameter specifies an event data rate; thus, in our example, the specified data rate is 100 events per second. This sink node records the task; the task state is purged from the node after the time indicated by the `duration` attribute.

For each active task, the sink periodically *broadcasts* an interest message to each of its neighbors. This initial interest contains the specified `rect` and `duration` attributes, but contains a much larger `interval` attribute. Intuitively, this initial interest may be thought of as exploratory; it tries to determine if there indeed are any sensor nodes that detect the four-legged animal. To do this, the initial interest specifies a low data rate (in our example, 1 event per second)[2]. In Section 3.4, we describe how the desired data rate is achieved by reinforcement. Then, the initial interest takes the following form:

```
type = four-legged animal
interval = 1s
rect = [-100, 200, 200, 400]
timestamp = 01:20:40                         // hh:mm:ss
expiresAt = 01:30:40
```

Before we describe how interests are processed, we emphasize that the interest is periodically *refreshed* by the sink. To do this, the sink simply re-sends the same interest with a monotonically increasing `timestamp` attribute. This is necessary because interests are not reliably transmitted throughout the network. The refresh rate is a protocol design parameter that trades off overhead for increased robustness to lost interests.

Every node maintains an interest cache. Each item in the cache corresponds to a *distinct* interest. Two interests are distinct, in our example, if their `type` attribute differs, their `freq` attribute differs, or their `rect` attributes are (possibly partially) disjoint. Interest entries in the cache *do not contain information about the sink.* Thus, interest state scales with the number of distinct active interests. Our definition of distinct interests also allows interest *aggregation*. Two interests I_1 and I_2, with identical types, completely overlapping `rect` attributes, can, in some situations, be represented with a single interest entry.

An entry in the interest cache has several fields. A `timestamp` field indicates the timestamp of the last received matching interest. The interest entry also contains several `gradient` fields, up to one per neighbor. Each gradient contains a `data rate` field requested by the specified neighbor, derived from the `interval` attribute of the interest. It also contains a *duration* field, derived from the `timestamp` and `expiresAt` attributes of the interest, and indicating the approximate lifetime of the interest.

When a node receives an interest, it checks to see if the interest exists in the cache. If no matching entry exists (where a match is determined by the definition of distinct

[2] This is not the only choice, but represents a performance tradeoff. Since the location of the sources is not precisely known, interests must necessarily be diffused over a broader section of the sensor network than that covered by the potential sources. As a result, if the sink had chosen a higher initial data rate, a higher energy consumption might have resulted from the wider dissemination of sensor data. However, with a higher initial data rate, the time to achieve high fidelity tracking is reduced.

interests specified above), the node creates an interest entry. The parameters of the interest entry are instantiated from the received interest. This entry has a single gradient towards the neighbor from which the interest was received, with the specified event data rate. In our example, a neighbor of the sink will set up an interest entry with a gradient of 1 event per second towards the sink. For this, it must be possible to distinguish individual neighbors. Any locally unique neighbor identifier may be used for this purpose. Examples of such identifiers include 802.11 MAC addresses [7], or Bluetooth [10] cluster addresses. If there exists an interest entry, but no gradient for the sender of the interest, the node adds a gradient with the specified value. It also updates the entry's `timestamp` and `duration` fields appropriately. Finally, if there exists both an entry *and* a gradient, the node simply updates the `timestamp` and `duration` fields.

In Section 3.3, we describe how gradients are used. When a gradient expires, it is removed from its interest entry. Not all gradients will expire at the same time. For example, if two different sinks express indistinct interests with different expiration times, some node in the network may have an interest entry with different gradient expiration times. When all gradients for an interest entry have expired, the interest entry itself is removed from a cache.

After receiving an interest, a node may decide to re-send the interest to some subset of its neighbors. To its neighbors, this interest *appears to originate from the sending node*, although it might have come from a distant sink. This is an example of a *local interaction*. In this manner, interests *diffuse* throughout the network. Not all received interests are re-sent. A node may suppress a received interest if it recently re-sent a matching interest.

Generally speaking, there are several possible choices for neighbors (Figure 3). The simplest alternative is to *re-broadcast* the interest to all neighbors. This is equivalent to flooding the interest throughout the network; in the absence of information about which sensor nodes are likely to be able to satisfy the interest, this is the only choice. This is also the alternative that we simulate in Section 4. In our example sensor network, it may also be possible to perform geographic routing, using some of the techniques described in the literature [13]. This can limit the topological scope for interest diffusion, thereby resulting in energy savings. Finally, in an immobile sensor network, a node might use cached data (see Section 3.3) to direct interests. For example, if in response to an earlier interest, a node heard from some neighbor A data sent by some sensor within the region specified by the `rect` attribute, it can direct this interest to A, rather than broadcasting to all neighbors.

Figure 2 shows the gradients established in the case where interests are flooded through a sensor field. Unlike the simplified description in Figure 1(b), notice that every pair of neighboring nodes establishes a gradient towards each other. This is a crucial consequence of local interactions. When a node receives an interest from its neighbor, it has no way of knowing whether that interest was in response to one it sent out earlier, or is an identical interest from another sink on the "other side" of that neighbor. Such two-way gradients can cause a node to receive one copy of low data rate events from each of its neighbors. However, as we show later, this technique can enable fast recovery from failed paths or reinforcement of empirically better paths (Section 3.4), and does not incur persistent loops (Section 3.3).

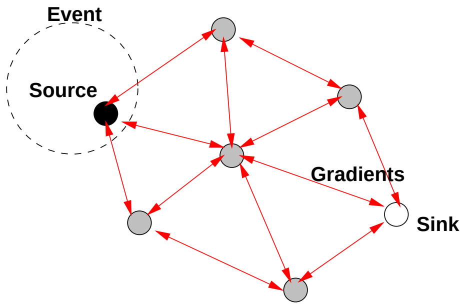


Figure 2: Gradient establishment. Note that neighboring nodes each have a gradient to the other.

Note that for our sensor network, a gradient specifies both a data rate and a direction in which to send events. More generally, a gradient specifies a *value* and a direction. The directed diffusion paradigm gives the designer the freedom to attach different semantics to gradient values. We have shown two examples of gradient usage. Figure 1(c) implicitly depicts binary valued gradients. In our sensor networks, gradients have two values that determine event reporting rate. In other sensor networks, gradient values might be used to, for example, probabilistically forward data along different paths, achieving some measure of load balancing (Figure 3).

In summary, interest propagation sets up state in the network (or parts thereof) to facilitate "pulling down" data towards the sink. The interest propagation rules are *local*, and bear some resemblance to join propagation in some Internet multicast routing protocols [9]. One crucial difference is that join propagation can leverage unicast routing tables to direct joins towards sources, whereas interest propagation cannot.

In this section, we have described interest propagation rules for a particular type of task. More generally, a sensor network may support many different task types. Interest propagation rules may be different for different task types. For example, a task type of the form "Count the number of distinct four-legged animals in rectangle R seen over the next T seconds" cannot leverage the event data rate as our example does. However, some elements of interest propagation are similar to both: the form of the cache entries, the interest re-distribution rules *etc.*. As part of our future research, we hope to cull these similarities into a *diffusion* substrate at each node, so that sensor network designers can use a library of interest propagation techniques (or, for that matter, rules discussed in the subsequent sections for data processing and reinforcement) for different task types.

3.3 Data Propagation

A sensor node that is within the specified `rect` processes interests as described in the previous section. In addition, the node tasks its local sensors to begin collecting samples. In this paper, we do not discuss the details of target recognition algorithms. Briefly, these algorithms simply match sampled waveforms against a library of pre-sampled, stored waveforms. This is based on the observation that a four-legged animal has a different acoustic or seismic footprint than, for example, a human being. The sampled waveform may match the stored waveform to varying extents; the algorithms usually associate a degree of confidence with the match. Furthermore, the intensity of the sampled waveform may roughly indicate distance of the signal origin, though perhaps not direction.

A sensor node that detects a target searches its interest cache for a matching interest entry. In this case, a matching entry is one whose `rect` encompasses the sensor location, and the `type` of the entry matches the detected target type. When it finds one, it computes the highest requested event rate among all its outgoing gradients. The node tasks its sensor subsystem to generate event samples at this highest data rate. In our example, this data rate is initially 1 event per second (until reinforcement is applied, Section 3.4). The source then sends to each neighbor for whom it has a gradient, an event description every second of the form:

```
type = four-legged animal // type of animal seen
instance = elephant       // instance of this type
location = [125, 220]     // node location
intensity = 0.6           // signal amplitude measure
confidence = 0.85         // confidence in the match
timestamp = 01:20:40      // local time when event was generated
```

This *data* message is, in effect[3], unicast individually to the relevant neighbors.

A node that receives a data message from its neighbors attempts to find a matching interest entry in its cache. The matching rule is as described in the previous paragraph. If no match exists, the data message is silently dropped. If a match exists, the node checks the *data cache* associated with the matching interest entry. This cache keeps track of recently seen data items. It has several potential uses, one of which is loop prevention. If a received data message has a matching data cache entry, the data message is silently dropped. Otherwise, the received message is added to the data cache and the data message is re-sent to the node's neighbors.

By examining its data cache, a node can determine the data rate of received events[4]. To re-send a received data message, a node needs to examine the matching interest entry's gradient list. If all gradients have a data rate that is greater than or equal to the rate of incoming events, the node may simply send the received data message to the appropriate neighbors. However, if some gradients have a lower data rate than others (caused by selectively reinforcing paths, Section 3.4), then the node may *downconvert* to the appropriate gradient. For example, consider a node that has been receiving data at 100 events per second, but one of its gradients (*e.g.,* set up by a second sink originating an indistinct task with a larger `interval`) is at 50 events per second. In this case, the node may only transmit every alternate event towards the corresponding neighbor. Alternately, it might interpolate two successive events in an application-specific way (in our example, it might choose the sample with the higher confidence match).

Loop prevention and downconversion illustrate the power of embedding application semantics in all nodes (Figure 3). Although this design is not pertinent to traditional networks, it is feasible with application-specific sensor networks. Indeed, as we show in Section 4.4, it can significantly improve network performance.

3.4 Reinforcement

In the scheme we have described so far, the sink initially diffuses an interest for a low event-rate notification (1 event per second). Once sources detect a matching target, they send low-rate events, possibly along multiple paths, towards the sink. After the sink starts receiving these low data rate events, it *reinforces* one particular neighbor in order to "draw down" higher quality (higher data rate) events. In general, this novel feature of directed diffusion is achieved by *data driven* local rules. One example of such a rule is to reinforce any neighbor from which a node receives a previously unseen event. To reinforce this neighbor, the sink re-sends the original interest message but with a smaller `interval` (higher data rate):

```
type = four-legged animal
interval = 10ms
rect = [-100, 200, 200, 400]
timestamp = 01:22:35
expiresAt = 01:30:40
```

When the neighboring node receives this interest, it notices that it already has a gradient towards this neighbor. Furthermore, it notices that the sender's interest specifies a higher data rate than before. If this new data rate is also higher than that of any existing gradient (intuitively, if the "outflow" from this node has increased), the node must also reinforce at least one neighbor. How does it do this? The node uses its data cache for this purpose. Again, the same local rule choices apply. For example, this node might choose that neighbor from whom it first received the latest event matching the interest. Alternatively, it might choose all neighbors from which new events[5] were recently received (this is the alternative we evaluate in Section 4).

[3] The exact mechanism used is a function of the radio's MAC layer and can have a significant impact on performance (Section 4.5).

[4] In our simulations in Section 4, as a simplification, we include the data rate in the event descriptions.

[5] The statement "reinforce a neighbor from which new events are received" implies that we reinforce that neighbor only if it is sending low data rate events. Obviously, we do not need to reinforce neighbors that are already sending traffic at the higher data rate.

Diffusion element	Design Choices
Interest Propagation	• Flooding • Constrained or directional flooding based on location • Directional propagation based on previously cached data
Data Propagation	• Reinforcement to single path delivery • Multipath delivery with selective quality along different paths • Multipath delivery with probabilistic forwarding
Data caching and aggregation	• For robust data delivery in the face of node failure • For coordinated sensing and data reduction • For directing interests
Reinforcement	• Rules for deciding when to reinforce • Rules for how many neighbors to reinforce • Negative reinforcement mechanisms and rules

Figure 3: Design Space for Diffusion

Through this sequence of local interactions, a path is established from source to sink transmission for high data rate events.

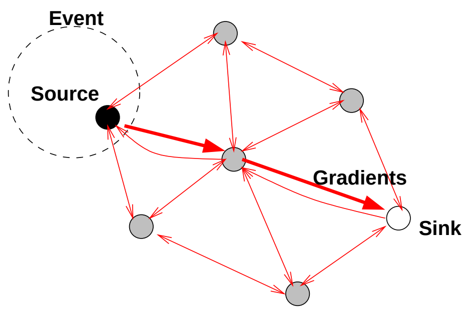


Figure 4: Gradients after the sink reinforces the empirically lowest delay path. Line thickness distinguishes between low and high data rate.

The local rule we described above, then, *selects an empirically low delay path* (Figure 4 shows the path that can result when the sink reinforces the path). It is very reactive to changes in path quality; whenever one path delivers an event faster than others, the sink attempts to use this path to draw down high quality data. However, because it is triggered by receiving one new event, this could be wasteful of resources. More sophisticated local rules are possible (Figure 3), including choosing that neighbor from which the most events have been received, or that neighbor which *consistently* sends events before other neighbors. These choices trade off reactivity for increased stability; exploring this tradeoff requires significant experimentation and is the subject of future work.

The algorithm described above can result in more than one path being reinforced. For example, if the sink reinforces neighbor **A**, but then receives a new event from neighbor **B**, it will reinforce the path through **B**[6]. If the path through **B** is consistently better (*i.e.,* **B** sends events before **A** does), we need a mechanism to *negatively reinforce* the path through **A**.

One mechanism for negative reinforcement is to time out all high data rate gradients in the network unless they are explicitly reinforced. With this approach, the sink would periodically reinforce neighbor **B**, and cease reinforcing neighbor **A**. The path through **A** would eventually degrade to the low data rate. Another approach, and one that we evaluate in this paper, is to explicitly degrade the path through **A** by re-sending the interest with the lower data rate. When **A** receives this interest, it degrades its gradient towards the sink. Furthermore, if all its gradients are now low data rate, **A** negatively reinforces those neighbors that have been sending data to it at a high data rate. This sequence of local interactions ensures that the path through **A** is degraded rapidly, but at the cost of increased resource utilization.

To complete our description of negative reinforcement, we need to specify what local rule a node uses in order to decide whether to negatively reinforce a neighbor or not. Note that this rule is orthogonal to the choice of mechanism for negative reinforcement. One plausible choice for such a rule is to negatively reinforce that neighbor from which no new events have been received (*i.e.,* other neighbors have consistently sent events before this neighbor) within a window of N events or time T. The local rule we evaluate in Section 4 is based on a time window of T, chosen to be 2 seconds in our simulations. Such a rule is a bit conservative and energy inefficient. For example, even if one event in ten was received first from neighbor A, the sink will not negatively reinforce that neighbor. Other variants include negatively reinforcing that neighbor from which fewer new events have been received. Significant experimentation is required before deciding which local rule achieves an energy efficient global behavior.

In describing reinforcement so far, we may have appeared to implicitly describe a single-source scenario. In fact, the rules we have described work with multiple sources. To see this, consider Figure 5. Assume initially that all initial gradients are low data rate. According to this topology, data from both sources reaches the sink via both of its neighbors

[6] This path may or may not be completely disjoint from the path through neighbor **A**.

C and **D**. If one of the neighbors, say **C** has consistently lower delay, our rules will only reinforce the path through **C** (this is depicted in the figure). However, if the sink hears **B**'s events earlier via **D**, but **A**'s events[7] earlier via **C**, the sink will attempt to draw down high quality data streams from *both* neighbors (not shown). In this case, the sink gets both sources' data from both neighbors, a potential source of energy inefficiency. Reinforcement rules that avoid this is the subject of future work.

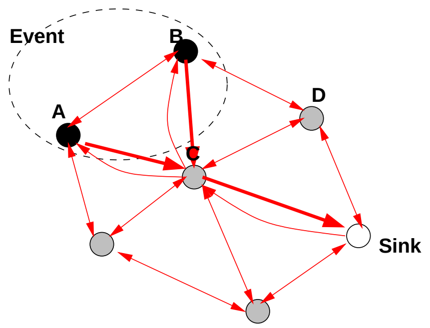


Figure 5: Gradients after reinforcement, multiple sources.

Similarly, if two sinks express identical interests, our interest propagation, gradient establishment and reinforcement rules work correctly. Without loss of generality, assume that sink **Y** in Figure 6 has already reinforced a high quality path to the source. Note however, that other nodes continue to receive low data rate events. When a human operator tasks the network at sink **X** with an identical interest, **X** can use the reinforcement rules to achieve the path shown. To determine the empirically best path, **X** *need not wait* for data—rather, it can use its data cache to immediately draw down high quality data towards itself.

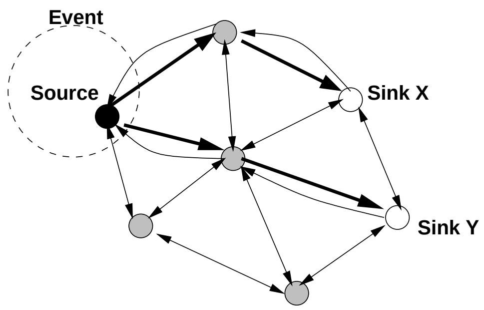


Figure 6: Gradients after reinforcement, multiple sinks.

So far, we have described situations in which reinforcement is triggered by a sink. However, in directed diffusion, *intermediate* nodes on a previously reinforced path can apply the reinforcement rules. This is useful to enable *local repair* of failed or degraded paths. Causes for failure or degradation include node energy depletion, and environmental factors affecting communication (*e.g.,* obstacles, rain fade). Consider Figure 7, in which the quality of the link between the source and node **C** degrades and events are frequently corrupted. When **C** detects this degradation—either by noticing that the event reporting rate from its upstream neighbor (the source) is now lower, or by realizing that other neighbors have been transmitting previously unseen location estimates—it can apply the reinforcement rules to discover the path shown in the figure. Eventually, **C** negatively reinforces the direct link to the source (not shown in the figure). Our description so far has glossed over the fact that a straightforward application of reinforcement rules will cause all nodes downstream of the lossy link to also initiate reinforcement procedures. This will eventually lead to the discovery of one empirically good path, but may result in wasted resources. One way to avoid this is for **C** to interpolate location estimates from the events that it receives so that downstream nodes still perceive high quality tracking. We are currently investigating other approaches.

[7] Note that in directed diffusion, the sink would not be able to associate a source with an event. Thus, the phrase "**A**'s events" is somewhat misleading. What we really mean is that data generated by **A** that is distinguishable in content from data generated by **B**.

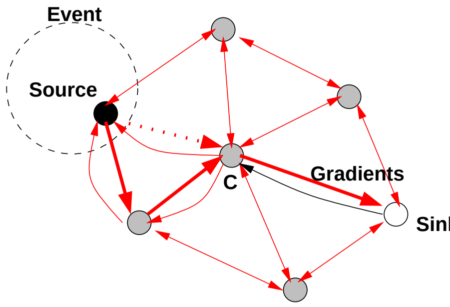


Figure 7: Gradients after local repair caused by connection quality degradation.

3.5 Discussion

In introducing the various elements of directed diffusion, we also implicitly described a particular *usage*—interests set up gradients drawing down data. The directed diffusion paradigm itself does not limit the designer to this particular usage. Other usages are also possible, such as the one in which nodes may propagate data in the absence of interests, implicitly setting up gradients when doing so. This is useful, for example, to spontaneously propagate an important event to some section of the sensor field. A sensor node can use this to warn other sensor nodes of impending activity.

Our description points out several key features of diffusion, and how it differs from traditional networking. First, diffusion is data-centric; all communication in a diffusion-based sensor network uses interests to specify named data. All communication in diffusion is neighbor-to-neighbor, unlike the end-to-end communication in traditional data networks. In other words, every node is an "end" in a sensor network. Second, there are no "routers" in a sensor network. Each sensor node can interpret data and interest messages. This design choice is justified by the task-specificity of sensor networks. Sensor networks are not general-purpose communication networks. Third, sensor nodes do not need to have globally unique identifiers or globally unique addresses. Nodes, however, do need to distinguish between neighbors. Finally, in an IP-based sensor network, for example, sensor data collection and processing might be performed by a collection of specialized servers which may, in general, be far removed from the sensed phenomena. In our sensor network, because every node can cache, aggregate, and more generally, process messages, it is possible to perform coordinated sensing close to the sensed phenomena.

However, diffusion is clearly related to traditional network data routing algorithms. In some sense, it is a *reactive* routing technique, since "routes" are established on demand. However, it differs from other ad-hoc reactive routing techniques in several ways.

- No attempt is made to find one loop-free path between source and sink before data transmission commences. Instead, constrained or directional flooding is used to set up a multiplicity of paths, and data messages are *initially* sent redundantly along these paths.

- Soon thereafter, reinforcement attempts to reduce this multiplicity of paths to a small number, based on empirically observed path performance.

- A message cache is used to perform loop avoidance. The interest and gradient setup mechanisms themselves do not guarantee loop-free paths between source and sink.

Why this peculiar choice of design? At the outset of this research, we consciously chose to explore path setup algorithms that establish network paths using strictly *local* (neighbor-to-neighbor) communication. The intuition behind this choice is the observation that physical systems (*e.g.,* ant colonies [5]) that build up transmission paths using such communication scale well and are extraordinarily robust. However, using strictly local communication implies that path setup cannot use global *topology* metrics; local communication implies that, as far as a node knows, the data that it received from a neighbor came from that neighbor [8]. This can be energy efficient in highly dynamic networks when changes in topology need not be propagated across the network. Of course, the resulting communication paths may be sub-optimal. However, the energy inefficiency due to path sub-optimality can be countered by carefully designed in-network aggregation techniques. Overall, we believe that this approach trades off some energy efficiency for increased robustness and scale.

Finally, it might appear that the particular instantiation that we chose, location tracking, has limited applicability. We believe, however, that such location tracking captures many of the essential features of a large class of remote surveillance sensor networks. We emphasize that, even though we have discussed our tracking network in some detail, much experimentation and evaluation of the various mechanisms is necessary before we fully understand the robustness, scale and performance implications of diffusion in general, and some of our mechanisms in particular. The next section takes an initial step in this direction.

[8]The location information in a data message might reveal otherwise, but that information still doesn't contain topology metrics.

4 Evaluating Directed Diffusion

In this section, we report on some results from a preliminary performance evaluation of our location tracking sensor network. We use packet-level simulation to explore, in some detail, the implications of some of our design choices. This section describes our methodology, compares the performance of diffusion against some idealized schemes, then explores impact of network dynamics on simulation.

4.1 Goals, Metrics, and Methodology

We implemented our animal tracking instance of directed diffusion in the *ns-2* [2] simulator. Our goals in conducting this evaluation study were four-fold:

- Place the performance of diffusion in the context of idealized schemes, such as flooding and *omniscient multicast* (described below). This serves as a sanity check for the intuition behind directed diffusion.

- Understand the impact of dynamics—such as node failures—on diffusion.

- Explore the influence of the radio MAC layer on diffusion performance.

- Study the sensitivity of directed diffusion performance to the choice of parameters.

We choose two metrics to analyze the performance of directed diffusion and to compare it to other schemes:

Average dissipated energy measures the ratio of total dissipated energy *per node* in the network to the number of *distinct* events seen by sinks. This metric computes the average work done by a node in delivering useful tracking information to the sinks. The metric also indicates the overall lifetime of the sensor network.

Average delay measures the average one-way latency observed between transmitting an event and receiving it at each sink. This metric defines the temporal accuracy of the location estimates delivered by the sensor network.

We study these metrics as a function of sensor network size.

In all our experiments, we operate the sensor network in a regime far from overload. Thus, our sensor nodes do not experience congestion. We do this to simplify our understanding of the results. Exploring the behavior of diffusion under congestion is the subject of future research. In passing, we note that there exist plausible approaches (such as in-network data rate downconversion or aggressive data quality reduction through aggregation) for dealing with congestion in diffusion-based sensor networks.

Despite this focus on uncongested operating regimes, directed diffusion can incur event losses, particularly under dynamics. In these situations, another metric for the performance of diffusion, is the *event delivery ratio*. This is the ratio of the number of distinct events received to the number originally sent. A similar metric was used in earlier work to compare ad-hoc routing schemes [4].

To completely specify our experimental methodology, we need to describe the sensor network generation procedure, our choice of radio parameters, and our workload. The following paragraphs do this.

In order to study the performance of diffusion as a function of network size, we generate a variety of sensor fields of different sizes. In each of our experiments, we study five different sensor fields, ranging from 50 to 250 nodes in increments of 50 nodes. Our 50 node sensor field generated by randomly placing the nodes in a 160m by 160m square. Each node has a radio range of 40m. Other sizes are generated by scaling the square and keeping the radio range constant in order to approximately *keep the average density of sensor nodes constant*. We do this because the macroscopic connectivity of a sensor field is a function of the average density. If we had kept the sensor field area constant but increased network size, we might have observed performance effects not only due to the larger number of nodes but also due to increased connectivity. Our methodology factors out the latter, allowing us to study the impact of network size alone on some of our mechanisms. For each network size, our results are averaged over three different generated fields.

The *ns-2* simulator implements a 1.6 Mbps 802.11 MAC layer. Our simulations use this MAC layer. This is not a completely satisfactory choice of MAC layer, since there are compelling energy efficiency reasons for selecting a TDMA-style MAC for sensor networks rather than one based on channel acquisition using RTS/CTS [20]. Briefly, these reasons have to do with energy consumed by the radio during idle intervals; with a TDMA-style MAC, it is possible to put the radio in standby mode during such intervals. By contrast, an 802.11 radio consumes as much power when it is idle as when it receives transmissions. To more closely mimic realistic sensor network radios [12], we altered the *ns-2* radio energy model such that the idle time power dissipation was about 35mW, or nearly 10% of its receive power dissipation (395mW), and about 5% of its transmit power dissipation (660mW). In Section 4.5, we analyze the impact of a MAC energy model in which listening for transmissions dissipates as much energy as receiving them.

Finally, in most of our simulations, we use a fixed workload which consists of five sources and five sinks. All sources are randomly selected from nodes in a 70m by 70m square within the sensor field. Sinks are uniformly scattered across the sensor field. Each source generates two events per second. The low data rate for directed diffusion was chosen to be one event in 50 seconds. Events were modeled as 64 byte packets, interests as 36 byte packets. Interests were periodically generated every 5 seconds, and the interest duration was 15 seconds. We chose the window for negative reinforcement to be 2 seconds. These parameter choices were informed both by the particular sensor network under consideration (small event descriptions, sources within a geographic region) and by our stated desire to explore a regime of the sensor network in a non-congested regime (see above).

4.2 Comparative Evaluation

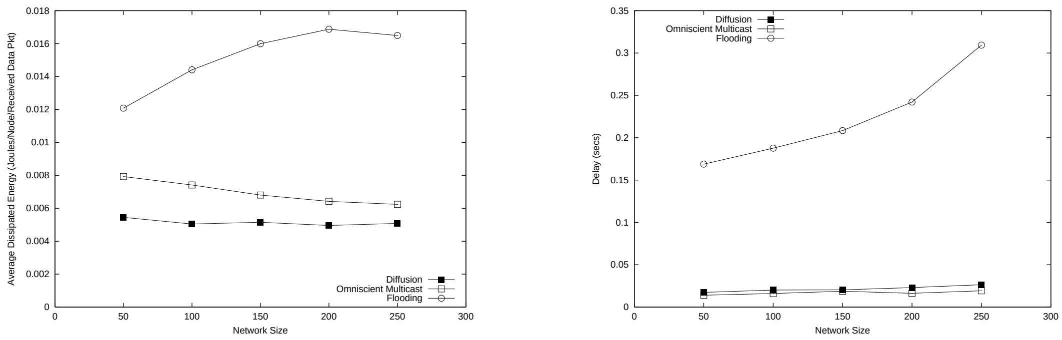
Our first experiment compares diffusion to two idealized schemes for data dissemination in networks:

Flooding In this scheme, sources flood all events to every node in the network. Flooding is a watermark for directed diffusion; if the latter is not significantly more energy efficient than flooding, it cannot be considered viable for sensor networks.

Omniscient Multicast In this scheme, each source transmits its events along a shortest-path multicast tree to all sinks. We *do not* simulate the tree construction protocols. Rather, we centrally compute the distribution trees and do not assign energy costs to this computation. Omniscient multicast approximately indicates the performance achievable in an IP-based sensor network. We use this scheme to give the reader some intuition for how our mechanism choices impact performance.

Figure 8(a) shows the average dissipated energy per packet as a function of network size. Omniscient multicast dissipates a little less than a half as much energy per packet per node than flooding. It achieves such energy efficiency by delivering events along a single path from each source to every sink. Directed diffusion has noticeably better energy efficiency than omniscient multicast. For some sensor fields, its dissipated energy is only 60% that of omniscient multicast. As with omniscient multicast, it also achieves significant energy savings by reducing the number of paths over which redundant data is delivered. In addition, diffusion benefits significantly from *in-network aggregation*. In our experiments, the sources deliver identical location estimates, and intermediate nodes *suppress* duplicate location estimates. This corresponds to the situation where there is, for example, a single four-legged animal within the specified sub-region.

Why then, given that there are five sources, is diffusion not nearly five times more energy efficient than omniscient multicast? First, both schemes expend comparable—and non-negligible—energy listening for transmissions. Second,



(a) Average dissipated energy

(b) Average delay

Figure 8: Directed diffusion compared to flooding and omniscient multicast.

our choice of reinforcement and negative reinforcement results in directed diffusion frequently drawing down high quality data along multiple paths, thereby expending additional energy. Specifically, our reinforcement rule that reinforces a neighbor who sends a new (*i.e.,* previously unseen) event is very aggressive. Conversely, our negative reinforcement rule, which negatively reinforces neighbors who only consistently send duplicate (*i.e.,* previously seen) events, is very conservative.

Figure 8(b) plots the average delay observed as a function of network size. Directed diffusion has a delay comparable to omniscient multicast. This is encouraging. To a first approximation, in an uncongested sensor network and in the absence of obstructions, the shortest path is also the lowest delay path. Thus, our reinforcement rules seem to be finding the low delay paths. However, the delay experienced by flooding is almost an order of magnitude higher than other schemes. This is an artifact of the MAC layer: to avoid broadcast collisions, a randomly chosen delay is imposed on all MAC broadcasts. Flooding uses MAC broadcasts exclusively. Diffusion only uses such broadcasts to propagate the initial interests. On a sensor radio that employs a TDMA MAC-layer, we might expect flooding to exhibit a delay comparable to the other schemes.

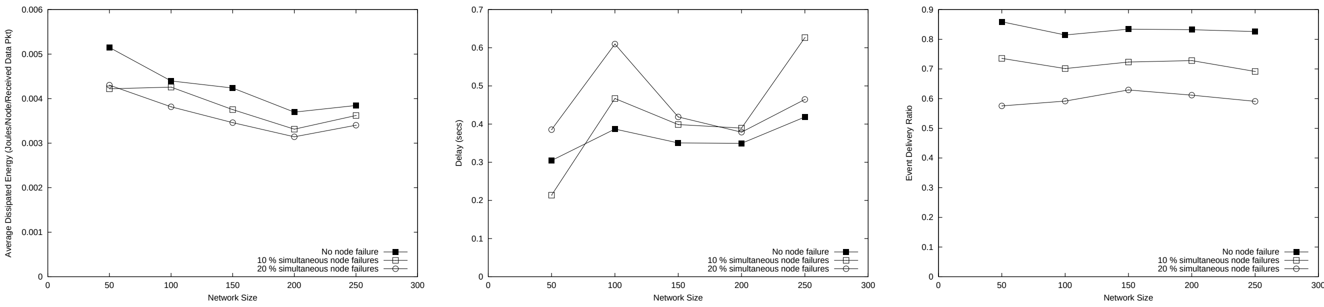
In summary, then, directed diffusion exhibits better energy dissipation than omniscient multicast and has good latency properties. Finally, all three schemes incurred an event delivery ratio of nearly one (not shown), since this experiment ignored network dynamics and was congestion-free.

4.3 Impact of Dynamics

To study the impact of dynamics on directed diffusion, we simulated node failures as follows. For each sensor field, repeatedly turned off a fixed fraction of nodes for 30 seconds. These nodes were uniformly chosen from the sensor field, with the additional constraint that an equal fraction of nodes on the sources to sinks shortest path trees was also turned off for the same duration. The intent was to create node failures in the paths diffusion is most likely to use, and to create random failures elsewhere in the network. Furthermore, unlike the previous experiment, each source sends different location estimates (corresponding to the situation in which each source "sees" different animals). We did this because the impact of dynamics is less evident when diffusion suppresses identical location estimates from other sources. We could also have studied the impact of dynamics on other protocols, but, because omniscient multicast is an idealized scheme that doesn't factor in the cost of route recomputation, it is not entirely clear that such a comparison is meaningful.

Our dynamics experiment imposes fairly adverse conditions for a data dissemination protocol. At any instant, 10 or 20 percent of the nodes in the network are unusable. Furthermore, we do not permit any "settling time" between node failures. Even so, diffusion is able to maintain reasonable, if not stellar, event delivery (Figure 9(c)) while incurring less than 20% additional average delay (Figure 9(b)). Moreover, the average dissipated energy actually *improves*, in some cases, in the presence of node failures. This is a bit counter-intuitive, since one would expect that directed diffusion would expend energy to find alternative paths. As it turns out, however, our negative reinforcement rules are conservative enough that several high-quality paths are kept alive in normal operation. Thus, at the levels of dynamics we simulate, diffusion doesn't need to do extra work. The lower energy dissipation results from the failure of some high-quality paths.

We take these results to indicate that the mechanisms in diffusion are relatively stable at the levels of dynamics we have explored. By this we mean that diffusion does not, under dynamics, incur remarkably higher energy dissipation or event delivery delays.



(a) Average dissipated energy **(b)** Average delay **(c)** Event Delivery Ratio

Figure 9: Impact of node failures on directed diffusion.

4.4 Impact of Diffusion Choices

To explain what contributes to directed diffusion's energy efficiency, we now describe two separate experiments. In both of these experiments, we do not simulate node failures. First, we compute the energy efficiency of diffusion with and without aggregation. Recall from Section 4.2 that in our simulations, we implement a simple aggregation strategy, in which a node suppresses identical data sent by different sources. As Figure 10 shows, diffusion expends nearly 5 times as much energy, in smaller sensor fields, as when it can suppress duplicates. In larger sensor fields, the ratio is 3. Our conservative negative reinforcement rule accounts for the difference in the performance of diffusion without suppression as a function of network size. With the same number of sources and sinks, the larger network has longer alternate paths. These alternate paths are truncated by negative reinforcement because they consistently deliver events with higher latency. As a result, the larger network expends less energy without suppression. We believe that suppression also exhibits the same behavior, but the energy difference is relatively small.

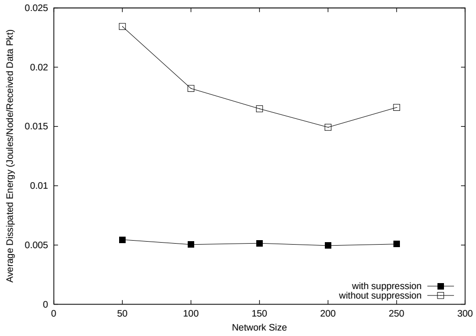


Figure 10: Impact of duplicate suppression on directed diffusion.

The second mechanism whose benefits we quantify is negative reinforcement. This mechanism prunes off higher latency paths, and can contribute significantly to energy savings. In this experiment, we selectively turn off negative reinforcement and compare the performance of directed diffusion with and without reinforcement. Intuitively, one would expect negative reinforcement to contribute significantly to energy savings. Indeed, as Figure 11 shows, diffusion without negative reinforcement expends nearly twice as much energy as when negative reinforcement is employed. This suggests that even our conservative negative reinforcement rules prune off paths which deliver consistently higher latency.

In the absence of negative reinforcement or suppression, diffusion's delay increases by factors of three to eight (the graphs are not included for lack of space). This is an artifact of the 802.11 MAC layer. In diffusion, data traffic is transmitted using MAC unicast. As more paths are used (in the absence of negative reinforcement), or more copies of data are sent (without suppression), MAC-layer channel contention increases, resulting in backoffs and subsequent delays.

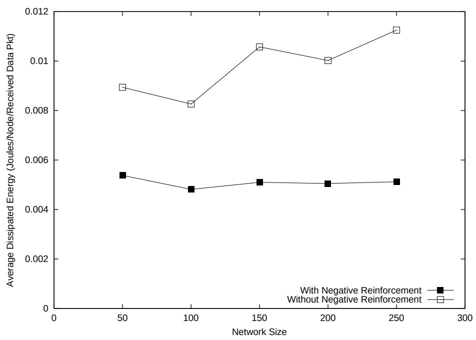


Figure 11: Impact of negative reinforcement on directed diffusion.

4.5 Sensitivity of Results

In this section, we evaluate the sensitivity of our comparisons (Section 4.2) to the various parameters we chose for simulation.

In our comparisons, we selected radio power dissipation parameters to more closely mimic realistic sensor radios [12]. We re-ran the comparisons of Section 4.2, but with power dissipation comparable to the AT&T Wavelan: 1.6W transmission, 1.2W reception and 1.15W idle. In this case, as Figure 12 shows, the distinction between the schemes disappears. In this regime, we are better off flooding all events. This is because idle time energy utilization completely dominates the performance of all schemes. This is the reason why sensor radios try very hard to minimize listening for transmissions.

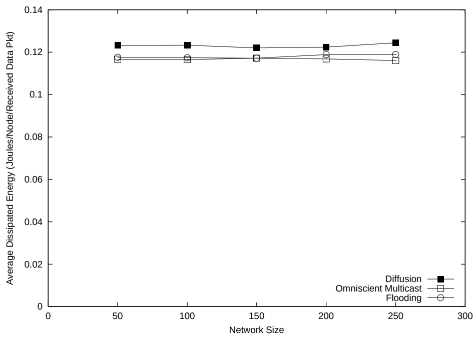


Figure 12: Impact of high idle radio power dissipation on energy efficiency.

In our comparisons, we used 5 sinks. How sensitive is our comparison to this choice? We re-ran our simulations of Section 4.2 with 25% of the nodes as sinks. As Figure 13 shows, diffusion's energy efficiency is more marked than with fewer sinks. Intuitively, this is because, with a higher fraction of the sensor nodes as sinks, fewer non-sink nodes are explored by diffusion's reinforcement rules. In this situation, diffusion is more energy efficient because it aggressively suppresses duplicates. However, unlike our earlier comparisons, flooding dissipates only twice as much energy as diffusion. Furthermore, omniscient multicast has energy efficiency comparable to flooding. Both these observations are attributable to using MAC-layer unicast for neighbor-to-neighbor transmissions in diffusion and omniscient multicast. With more sinks, and consequently increased branching at each node, each node transmits the same packet multiple times, once to each neighbor. Diffusion is less impacted by this because it performs in-network suppression of identical data. Finally, with more sinks and larger sensor fields, the event delivery ratio of diffusion falls to 80% (not shown). This can be attributed to suppression. Especially when the paths from sources to a sink overlap significantly, diffusion will suppress duplicates close to the sources and event losses nearer the sinks cannot be recovered.

We also conducted an experiment where the sources in our comparisons of Section 4.2 were uniformly distributed over the entire sensor field. For our choice of parameters, the results were not sensitive to this change.

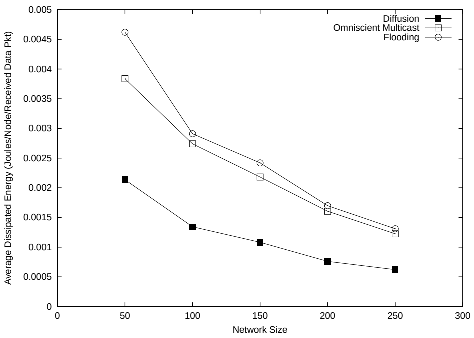


Figure 13: Impact of more sinks on energy efficiency.

4.6 Summary of Results

There are several lessons we can draw from this preliminary evaluation of diffusion:

- Directed diffusion has the potential for significant energy efficiency. Even with relatively unoptimized path selection, it outperforms an idealized traditional data dissemination scheme like omniscient multicast.
- Diffusion mechanisms are stable under the ranges of network dynamics considered in this paper.
- For directed diffusion to achieve its full potential, however, careful attention has to be paid to the design of sensor radio MAC layers.

5 Related Work

To our knowledge, distributed sensor networks have not been extensively studied in the networking literature. However, our work has been informed and influenced by a variety of other research efforts, which we now describe.

Distributed sensor networks are a specific instance of ubiquitous computing as envisioned by Weiser [25]. Ubiquitous computing systems are capable of adapting behaviors based on environmental conditions. Its computing elements are so seamlessly integrated into the environment as to be invisible to users. Early ubiquitous computing efforts, however, did not approach the issues of scalable node coordination, focusing more on issues in the design and packaging of small, wireless devices.

More recent efforts, such as WINS [21] and Piconet [3] have begun to consider networking and communication issues for small wireless devices. The WINS project has made significant progress in identifying feasible radio designs for low-power environmental sensing. Their project has focused also on low-level network synchronization necessary for network self-assembly. Our directed diffusion primitives provide inter-node communication once network self-assembly is complete. Although the Piconet project is more focused on enabling home and office information discovery, their application designs have some similarity to the data caching and aggregation that diffusion employs.

In addition, recent work has pointed out some of the advantages of diffusion-like application-specificity in the context of sensor networks [14]. Specifically, this work showed how embedding application semantics in flooding can help achieve energy-efficiency. Directed diffusion explores some of these same ideas in the context of more sophisticated distributed sensing algorithms.

Some of the inspiration for directed diffusion comes from biological metaphors, such as reaction-diffusion models for morphogenesis. Turing proposed the reaction-diffusion model [24] to explain pigmentation patterns in animals. The model demonstrates how local interactions of two morphogens (an activator and an inhibitor), can model these patterns. To the extent that it attempts to find local interactions between neighboring nodes that result in some larger-scale objective (*e.g.,* energy efficient reinforcement), directed diffusion draws inspiration from this work.

On a similar level, directed diffusion draws inspiration from ant colony metaphors. These metaphors are based on the fact that the almost blind ants seem to be able to find shortest paths to destinations using only the pheromone trails deposited by other ants. Earlier work has tried to directly apply the metaphor to adaptively route traffic in data and communication networks [5, 23, 22]. This work employs reinforcement learning techniques, driven by route histories at each node, to compute eventually optimal paths. Despite drawing inspiration from these efforts, directed diffusion differs in some ways: earlier work has not focused on energy-efficient wireless communication, and does not attempt to leverage task-specificity, caching and aggregation.

Directed diffusion borrows heavily from the literature on ad-hoc unicast routing. Specifically, it is a close kin of the class of several reactive routing protocols proposed in the literature [11, 19, 18]. Of these, it is possibly closest to [18] in its attempt to localize repair of node failures, and its deemphasis of optimal routes. The differences between ad-hoc routing and directed diffusion have already been discussed in Section 3.5.

Many of the techniques developed for improving ad hoc routing performance can be directly applied to directed diffusion. In this class, we include techniques that reduce the impact of broadcast storms [16], techniques that localize route queries based on geographical information [13] or based on route history [6]. Directed diffusion has the additional degree of freedom in being able to use application semantics to achieve further efficiency

Directed diffusion is influenced by the design of multicast routing protocols. In particular, propagation of reinforcements and negative reinforcements are similar to joins and prunes in shared-tree construction [9]. The initial interest dissemination and gradient setup is similar to data-driven shortest-path tree setup [8]. The difference, of course, is that where Internet protocols rely on underlying unicast routing to aid tree setup, diffusion cannot. Diffusion can, however, do in-network processing of data (caching and aggregation) unlike existing multicast routing schemes.

The in-network processing feature of directed diffusion bears some resemblance to router assist for localized error recovery in reliable multicast [15, 17]. These schemes allow minimal router functionality that allows specialized forwarding modes for certain kinds of data. Directed diffusion carries this idea further, leveraging the task specificity of sensor networks to embed application knowledge in network nodes.

Finally, interest dissemination, data propagation and caching in directed diffusion are all similar to some of the ideas used in adaptive Web caching [26]. In these schemes, caches self-organize themselves into a hierarchy of cooperative caches through which requests for pages are effectively *diffused.*

6 Conclusions and Future Work

In this paper, we described the directed diffusion paradigm for designing distributed sensing algorithms. This paradigm has some novel features—data-centric dissemination, reinforcement-based adaptation to the empirically best path, and in-network data aggregation and caching. To our knowledge, no previous networking work has designed and evaluated a data distribution mechanism incorporating these features. There is a good reason for this—these features may not be justifiable in the context of traditional networks. However, as we show here, these features can enable highly energy-efficient and robust dissemination in dynamic sensor networks, while at the same time minimizing the per-node configuration that is characteristic of today's networks.

As we have emphasized before, this work represents an initial foray into the design of diffusion mechanisms. We have tried to give the reader a sense of the enormity of the design space. Our remote surveillance network represents a non-trivial exploration of this design space. Even for this network, we have not explored alternative designs such as the impact of reinforcement strategies on energy efficiency, and the energy savings obtainable by more sophisticated aggregation strategies. To draw a simple analogy, we are with sensor networks where we were with the Internet about 3 decades ago.

References

[1] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The Design and Implementation of an Intentional Naming System. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 186–201, Charleston, SC, 1999.

[2] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. Improving simulation for network research. Technical Report 99-702b, University of Southern California, March 1999. revised September 1999, to appear in IEEE Computer.

[3] F. Bennett, D. Clarke, J. Evans, A. Hopper, A. Jones, and D. Leask. Piconet: Embedded Mobile Networking. *IEEE Personal Communications*, 4(5), October 1997.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad-Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, Dallas, TX, 1998.

[5] Gianni Di Caro and Marco Dorigo. AntNet: A Mobile Agents Approach to Adaptive Routing. Technical Report 97-12, IRIDIA, Universite' Libre de Bruxelles, 1997.

[6] Robert Castaneda and Samir R. Das. Query Localization Techniques for On-demand Routing Protocols in Ad Hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, WA, 1999.

[7] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical Report 802.11-1997, Institute of Electrical and Electronics Engineers, New York, NY, 1997.

[8] S. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of the ACM SIGCOMM*, pages 55–64, August 1988.

[9] S. E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The PIM Architecture for Wide-Area Multicast Routing. *IEEE Transactions on Networking*, 4(2), April 1996.

[10] The Bluetooth Special Interest Group. Bluetooth v1.0B Specification. http://www.bluetooth.com, 1999.

[11] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad-hoc Wireless Networks. In T. Imielinksi and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[12] William J. Kaiser. WINS NG 1.0 Transceiver Power Dissipation Specifications. Sensoria Corp.

[13] Yong-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'98)*, Dallas, TX, 1998.

[14] Joanna Kulik, Wendi Rabiner, and Hari Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, WA, 1999.

[15] J. C. Lin and S. Paul. A Reliable Multicast Transport Protocol. In *Proceedings of the IEEE Infocom*, San Francisco, CA, March 1996.

[16] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)*, Seattle, WA, 1999.

[17] C. Papadopoulos, G. Parulkar, and G. Verghese. An Error Control Scheme for Large-scale Multicast Applications. In *Proceedings of the IEEE Infocom*, San Francisco, March 1998.

[18] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of INFOCOM 97*, pages 1405–1413, April 1997.

[19] Charles Perkins. Ad-Hoc On Demand Distance Vector Routing (AODV). Internet-Draft, November 1997. draft-ietf-manet-aodv-00.txt.

[20] G. Pottie and W. Kaiser. Wireless Sensor Networks. *Communications of the ACM*, 2000. To appear.

[21] G. Pottie, W. Kaiser, L. Clare, and H. Marcy. Wireless Integrated Network Sensors. submitted for publication, 1998.

[22] Ruud Schoonderwoerd, Owen Holland, Janet Bruten, and Leon Rothkrantz. Ants for Load Balancing in Telecommunications Networks. Technical Report HPL-96-35, Hewlett-Packard Laboratories Bristol, 1996.

[23] Devika Subramanian, Peter Druschel, and Johnny Chen. Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, August 1997.

[24] A. M. Turing. The Chemical Basis of Morphogenesis. *Phil. Transaction of the Royal Society of London, Series (B): Biological Sciences*, (237):37–72, 1952.

[25] M. Weiser. The Computer for the 21st Century. *Scientific American*, September 1991.

[26] L. Zhang, S. Michel, S. Floyd, V. Jacobson, K. Nguyen, and A. Rosenstein. Adaptive Web Caching: Towards a New Global Caching Architecture. In *Proceedings of the Third International Caching Workshop*, June 1998.