DV/DRP: A Content-Based Networking Protocol For Sensor Networks

Cyrus P. Hall[†] Antonio Carzaniga[†,‡] Alexander L. Wolf[†,‡]

[†]Faculty of Informatics
University of Lugano
6900 Lugano, Switzerland

[‡]Department of Computer Science
University of Colorado
Boulder, Colorado 80309-0430

University of Lugano
Faculty of Informatics
Technical Report 2006/04
September 2006

Abstract

An ideal sensor network would minimize communication by routing information only to those nodes requiring the information. We are exploring the use of a content-based network for this purpose, where messages containing sensor readings and associated metadata are relayed from source nodes to destination nodes based solely on the fact that the destination nodes have expressed interest in specific message content. This paper contributes a concrete protocol, called DV/DRP, that implements content-based networking for wireless sensor networks or other similarly constrained network configurations. DV/DRP augments a basic distance vector protocol to construct both primary and alternate routes. DV/DRP also features a new content-based forwarding technique called *dynamic receiver partitioning*. DV/DRP makes minimal assumptions on the underlying MAC layer. Notably, it uses only a primitive local broadcast, does not require reliability of the link layer nor the use of acknowledgments, and explicitly handles asymmetric links. We present simulations showing that the protocol scales to large networks while minimizing the resource consumption of individual nodes. We also show that the protocol is robust with respect to transient and permanent node failures, as well as asymmetries in wireless links. Finally, to demonstrate that DV/DRP is suitable for memory-constrained sensor networks, we discuss a preliminary implementation.

1 Introduction

A sensor network is typically composed of a set of nodes that are capable of measuring various types of phenomena. The kinds of data, as well as the rates at which those data can be usefully consumed, are characteristics of the applications built on top of a sensor network. Sensor nodes themselves are increasingly seen as general-purpose, commodity items that simply produce generic data. Therefore, as others have argued [14], the message traffic in a sensor network should be driven by the dynamic "interests" of the application, rather than by the particular static capabilities or configurations of the sensors. The application might, for example, be interested in receiving only temperature readings from a particular region that exceed a certain threshold or, because of its knowledge of how the data are to be used, in receiving only periodic messages containing a temperature reading.

Sensor networks are an ideal application of the *content-based networking* model of communication [6]. In this model a message is transmitted from a sender to one or more receivers without the sender having to address the message to any specific receiver. Receivers express interest in the kinds of messages they would like to receive, and the network delivers to the receivers any and only messages matching those interests. Interests are expressed by receivers through *predicate advertisements*. In this receiver-driven style of communication, the network is responsible for efficiently applying predicates to the content of messages so as to minimize the computational and communication costs of the network.

We have designed and implemented a protocol that realizes the content-based networking model specifically for wireless sensor networks and other similar network configurations. Our protocol is innovative in several respects, most notably because it: (1) requires little-to-no global pre-configuration of the network; (2) supports a service model that allows applications to tailor the network to dynamically changing requirements for data and data rates; (3) adapts to asymmetries in link-layer radio transmissions, as well as to transient and permanent relay-node failures; and (4) requires no more than a primitive, local-broadcast MAC layer that, to the contrary, is leveraged to achieve additional efficiencies in network utilization. Previous protocols for sensor networks have not achieved such a combination of high-level functionality and low-level efficiency.

To be more concrete, the design of the protocol is based on the following assumptions.

- *Primitive communication infrastructure:* Nodes in a sensor network have direct access only to their immediate neighbors and communicate with them through a local-broadcast link-layer communication service. No other type of service is assumed to be available. In particular, the link layer does not provide point-to-point communication or reliable transfer. Also, there is no multi-hop network service such as unicast, multicast, anycast, or flooding (broadcast) available. In addition, links to neighbors are mostly asymmetric in nature and nodes can not rely on stable connections.

- *Resource-constrained nodes:* Nodes are severely limited in the amount of state they can maintain, in the number of messages they can send and receive, and in the amount of energy they can consume.

- *Many senders and few receivers:* Most of the nodes will act as information producers, while only a few, resource-rich nodes will act as information consumers. The producers (i.e., message senders) are the sensor nodes. The consumers (i.e., message receivers) are commonly referred to as *base stations*, providing functions such as data correlation and gateways to higher-level applications.

In addition, we are currently assuming that sensor nodes are stationary.

We can summarize the key features of the protocol that together distinguish it from others proposed for sensor networks.

- *Receiver predicates and message content determine minimal message flows and avoid network flooding.* Predicates are applied to a message at the node where the message enters the network.[1] Using a compact bit vector, the message is annotated to record the set of receivers having a matching predicate. Relay nodes evaluate the bit vector, rather than the full predicates, and forward only toward the intended receivers.

- *Forwarding state in relay nodes and message headers combine to avoid loops and redundant paths.* The protocol uses a receiver-driven distance-vector route discovery scheme to build forwarding state corresponding to a shortest-paths spanning tree for each receiver. Relay nodes use their local forwarding state and the receiver bit vector attached to a message to decide whether they are on the best path to one or more of the

[1] Sensor nodes are entry points for the messages they produce for their own data, as well as relay nodes for the messages produced by other sensor nodes; they must be capable of exhibiting both functionalities.

intended receivers. Before forwarding the message, a relay node will remove from the receiver set (encoded in the bit vector) any receivers for which it is not on the best path. The global effect is to progressively partition the receiver set in such a way that loops and redundant paths are avoided.

- *Alternative routing paths compensate for network failures and communication asymmetries.* Piggybacking on the construction of shortest-path forwarding state, the protocol builds a configurable number of alternative paths of equal or greater length. These are used to route around transient failures in next-hop nodes. Failures are detected by the absence of a forwarding "echo", which would normally be heard within a local broadcast region when the next-hop node itself forwards the message. Thus, the forwarding echo effectively serves as an acknowledgment that the next-hop node has received the message. Alternative routing paths are also used in the presence of transient asymmetric links that lead to highly unidirectional communication flows (and, thereby, lost forwarding echoes) within a local broadcast region.

- *Piggybacked failure reports and blacklisting of links are used to detect and avoid permanent failures and asymmetries.* Whenever a failure causes a message to deviate from the primary path, a failure bit is marked in the message to alert the receiver. Upon receiving repeated failure reports, receivers reissue their predicate advertisements to obtain fresh valid routes. At the local level, repeated failures (i.e., failure to hear forwarding echoes) are interpreted as permanent link asymmetries. Asymmetric links are blacklisted and thereby ignored when setting up paths for subsequent predicate advertisements.

- *Limits on message delivery rates, rather than message production rates, allow different rates to be experienced by different receivers sharing the same senders.* Following the principle that application interests should drive message traffic in a sensor network, the protocol allows different receivers (i.e., base stations) to specify different delivery rates as part of predicate advertisements. This allows the network to optimize traffic flow for each receiver under the given rate limitations by adaptively integrating production rates across multiple senders located around the network. This feature is also essential in preventing network congestion around receiver nodes, especially in the presence of large networks.

We call our protocol the *distance vector / dynamic receiver partitioning* (DV/DRP) protocol.

In this paper we present the design of DV/DRP and give results of a quantitative evaluation conducted over a range of simulated scenarios. We realize that the validity of our evaluation depends on our choice of abstractions. Therefore, in this study we were especially careful to model the communication layers below DV/DRP in realistic detail. Specifically, the simulation of the main protocol rests on top of three stacked models. The first is a model of a MAC layer similar to B-MAC [18]. The second is a wireless-device model that captures specific properties of the radio and antenna subsystems of each node, including anisotropic emissions characterizing each antenna. The third is a model of the propagation of radio signals in the field. This model is responsible for delivering signals from transmitters to receivers taking into account the combined effect of all overlapping radio signals at any given time.

Our evaluation shows that DV/DRP is: (1) functional and stable with respect to growing networks; (2) robust in the face of transmission collisions and other transient or permanent network failures; (3) efficient in detecting and avoiding asymmetric links; and (4) effective in holding network resource consumption to a minimum for a given amount of data traffic. For example, in a network of 250 sensors with up to 20 base stations and in the presence of a large number of asymmetric links, DV/DRP is capable of maintaining a high delivery rate, with less than 10% false negatives (misdeliveries) and with an average level of control traffic that remains under 60 packets per second throughout the entire network.

2 Related Work

The sensor networking community has studied a wide variety of network protocols. A large portion of this work involves techniques for optimizing various aspects of communication. For example, network data aggregation [15, 17], node clustering techniques [12] for minimizing radio listening time, and energy efficient MAC protocols [8, 18, 21, 22] have all been explored. Many of these techniques can be applied to a network running DV/DRP to further improve its performance.

At a higher level of communication, *directed diffusion* [14] shares many of our goals. Directed diffusion is a data-driven network service that provides a similar interface to that of DV/DRP. The general routing strategy underlying directed diffusion is based on three processes, not all necessarily used together: (1) receiver interests are “diffused” throughout a sensor field to establish so-called “gradients”; (2) sensor data are sent toward receivers either in a flooding fashion or by following gradients; and (3) gradients are dynamically modified by applications through reinforcement of paths.

Although originally conceived as a particular routing protocol, directed diffusion is now presented as a general conceptual framework for a family of protocols [11]. Many fundamental protocol decisions are not defined, but instead delegated to the application (e.g., gradient establishment and reinforcement), giving great flexibility to the protocol designer.

DV/DRP is rather different from directed diffusion in this respect. DV/DRP is a fully specified protocol with an application interface that clearly isolates the application from routing decisions (see Section 3). We chose this architecture for two main reasons. First, our target is a network of extremely simple and resource-constrained nodes. In such networks, deploying and executing application-specific logic on each node may be either infeasible or undesirable. Second, and perhaps more importantly, we see potential problems in delegating routing decisions to the application developer. In particular, doing so offers little or no guarantees in terms of interoperability, global optimality, or even stability.

In recent work, the general directed diffusion framework has been instantiated as three classes of protocols [20]. Two of these protocol classes, “push” and “two-phase pull” diffusion, are based on a common scheme in which sensor data are flooded throughout the network, and receivers establish preferred routes through positive reinforcement. The primary benefit of this scheme is that by keeping multiple paths alive (i.e., by replicating messages) the network is probabilistically more reliable. There has also been work done on using multipath routing in directed diffusion [9]. However, that solution still replicates messages when there are no failures in the primary path.

In DV/DRP we adopt a rather different approach to route recovery. In particular, DV/DRP deals with transient failures using *local alternate routes* and permanent route failures and asymmetries using a *reactive repair* mechanism. While we have not yet performed a detailed analysis, we hypothesize that the overhead involved in continuously inflating traffic on a per-message basis, as done in push and two-phase pull diffusion, is greater than the cost of maintaining local alternate routes and repairing broken paths on a per-receiver basis.

DV/DRP is most similar to the third directed diffusion protocol class, “one-phase pull”. One-phase pull is purely a publish/subscribe scheme, where interests (subscriptions) are flooded to all nodes, and where data (publications) follow backwards the paths established by subscriptions. Algorithms for such a scheme have been extensively studied in the publish/subscribe literature and in the area of content-based networking [4, 5, 7]. While forming the theoretical foundation of DV/DRP, the algorithms in the work described here are substantial adaptations for use in the resource-constrained and primitive environment of a wireless sensor network.

3 Routing and Forwarding

The content-based networking model [6] offers a promising approach to energy-efficient, network-level communication services in resource-constrained sensor networks. Instead of using explicitly configured and addressed end points to form communication paths, routes are formed by receivers advertising a message selection predicate to the network. A message containing data that match one or more selection predicates is forwarded toward the receivers advertising those predicates.

Consider a base station in a sensor network monitoring wildland fire conditions. The base station may want to be notified (on behalf of some higher-level application) if the humidity or wind speed have reached critical values. The base station might advertise the selection predicate shown in Figure 1. Following the simple model of Carzaniga and Wolf [7], a predicate is a disjunction of *filters*, which in turn are made up of conjunctions of *constraints*. In Figure 2, the predicate is formed from two filters, indicated by the horizontal line separating them. Each constraint has a *type* (from a collection of primitive types), a *name*, an *operator*, and a *value*. The horizontal

`int` wind_speed >= 30
`int` wind_dir > 0
`int` wind_dir < 160
`int` temperature > 150
`int` humidity <= 5

Figure 1: Example Receiver Predicate

```
int wind_speed = 45
int wind_dir = 78
int node = 13
```

```
int wind_speed = 47
int wind_dir = 180
```

Figure 2: Example Messages

line in this example indicates that the predicate is a disjunction of two expressions; each expression is itself a conjunction of more primitive constraints on individual values.[2] Sensor nodes might produce messages like the ones shown in Figure 2. A message is a list of *attributes*. Much like constraints, attributes consist of a type, a name, and a value, but the operator is always equality. The messages are routed toward receivers that have advertised matching predicates.

Clearly, this form of network-level service model is akin to the application-level communication model known as *publish/subscribe*. Similar models have been proposed and used in the design of sensor networks [14].

In the context of a multi-hop network protocol, forwarding information is derived, in part, from the predicates. Predicates are used in the construction of forwarding tables residing at individual nodes. The forwarding algorithm performs a match against these predicates. If a message matches the constraints associated with one or more outbound interfaces, then the message is forwarded to the next node(s) along the path(s) to the intended receiver(s). In the example above, the first message matches the predicate, while the second one does not.

In a resource constrained environment such as a sensor network we must ensure that the routing and forwarding information is small enough to fit within the limited memory available. Moreover, because the network protocol directly influences the total network energy expenditure, we must ensure that paths are optimal and that extraneous communication is minimized. Our hypothesis, substantiated in this paper, is that a content-based network allows us to push filtering functionality deep into the network layer, resulting in a reduction of unwanted message traffic.

3.1 Routing

Routing in DV/DRP provides an abstraction over a MAC layer from which we require only an unreliable local broadcast. This minimalistic assumption means that DV/DRP is adaptable to a wide range of MAC protocols. In Section 4 we discuss our choice of a specific MAC protocol for experimentation. However, for the purposes of this section, we simply assume that any "level 2" communication is either a transmission or a reception of a local-broadcast packet; DV/DRP controls the propagation of messages and, in fact, all communications across overlapping local-broadcast regions.

Not having any point-to-point connection between nodes, we use the term *link* exclusively to refer to the routing/forwarding state maintained by DV/DRP. In particular, path information maintained by the routing algorithm consists of *upstream* and *downstream* next-hop links that reside within the same local-broadcast region. This terminology refers to the flow of messages from senders to receivers. Thus, downstream links point toward receivers, whereas upstream links point toward senders and away from receivers.

As we discuss below, DV/DRP makes extensive use of piggyback communication, where an upstream flow is also used to send some information back downstream (or vice versa). This kind of counter-flow communication works under the assumption that links are bidirectional. Thus, we assume that if node y is in the local-broadcast region of node x, then node x is in the local-broadcast region of node y. While this assumption is reasonable under many general network configurations, some degree of link asymmetry, which might lead to unidirectional links, is unavoidable. DV/DRP copes with unidirectional links by detecting and then avoiding them, whenever possible. We discuss this aspect of DV/DRP in Section 3.3.3.

3.1.1 Distance Vectors

DV/DRP uses a straightforward distance-vector routing protocol augmented to maintain an array of alternate next hops. When a receiver node r advertises a predicate p_r, the routing protocol propagates p_r to every node in the network, thereby forming a content-based forwarding tree rooted at r. This can be thought of as the equivalent of a *route advertisement*. The content-based forwarding tree "attracts" messages matching p_r toward r.

[2] Attribute names, such as "wind_speed", need not be strings; we envision using enumerated integers for actual deployment.

The routing protocol uses a table to represent a distance vector. For convenience, the same table also stores receiver predicates and, therefore, serves as a forwarding table. For each receiver r in the network, the table stores path information as well as path-independent information. The path-independent information consists of the receiver predicate p_r, a sequence number s_r, a bit-vector position b_r, the minimum inter-arrival interval Δ_r, and the time of the last forwarded message t_r.

For a given node, and for each receiver r, downstream links form an array of distance-vector entries (n_r, l_r), (n'_r, l'_r), (n''_r, l''_r), etc. The length of this array is a static parameter of the protocol. The first entry in the array represents the usual distance-vector information. In particular, n_r is the next-hop node on the shortest path toward r, and l_r is the length of that path. The other entries represent the *local second-best path*, the *local third-best path*, and so on, which are used as alternate paths in case of path failure. We define the *local* n^{th}*-best path* as the n$^{\text{th}}$-best path computed by a traditional distance-vector protocol.

In practice, an alternate path goes one hop away from the current node (along a sub-optimal path), and then follows a primary (or, if necessary, alternate) path from there. This is illustrated in Figure 3, which shows a network of five nodes and a content-based forwarding tree for a receiver node s. The figure also shows two alternate paths from node v to s.

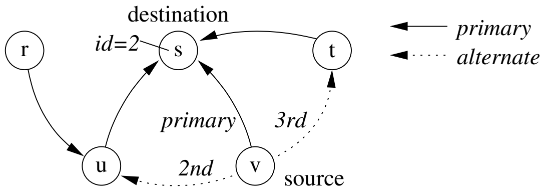
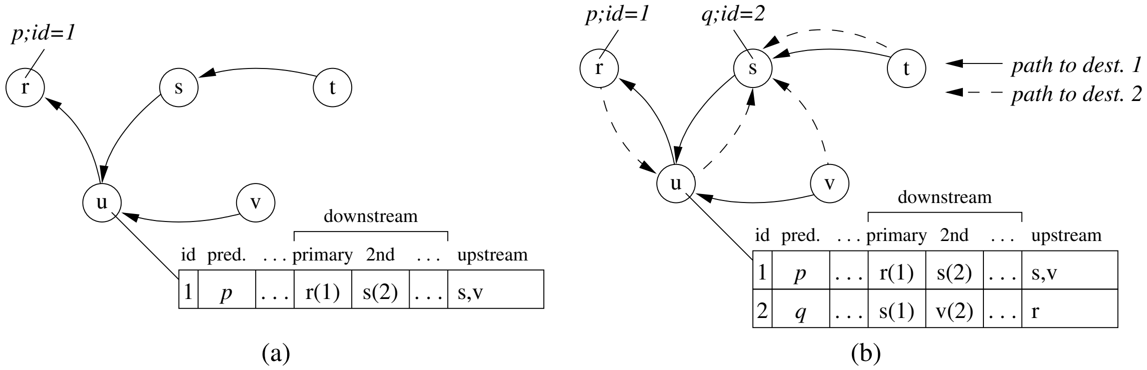


Figure 3: Primary and Alternate Paths

Upstream links are simply the reverse of downstream links. Specifically, for each receiver r, a node x maintains a set of upstream nodes U_r containing any and all neighbor nodes that have x as their primary (downstream) next-hop toward r. For example, in the scenario of Figure 3, node u will have $U_2 = \{r\}$, while node s will have $U_2 = \{t, u, v\}$.

Figure 4 shows the five-node network at two points in time during which nodes r and s are established as receiver nodes. Initially, node r advertises predicate p. The predicate propagates through the network generating a content-based forwarding tree rooted at node r represented by the solid arrows in Figure 4a. (For simplicity, the diagram shows only the downstream links.) Later, node s advertises predicate q. This predicate propagates throughout the network, creating the forwarding tree represented in Figure 4b with dashed arrows. The figure shows partial states of the routing/forwarding table of node u. Notice that a distance-vector entry gives the next-hop node and the distance to the receiver, whether through a primary path (shown in the diagrams) or through an alternate path (not shown in the diagrams).



			downstream			
id	pred.	. . .	primary	2nd	. . .	upstream
1	*p*	. . .	r(1)	s(2)	. . .	s,v

(a)

			downstream			
id	pred.	. . .	primary	2nd	. . .	upstream
1	*p*	. . .	r(1)	s(2)	. . .	s,v
2	*q*	. . .	s(1)	v(2)	. . .	r

(b)

Figure 4: Routing/Forwarding Tables

3.1.2 Route Setup

The propagation of predicates follows a simple variation of a traditional distance-vector protocol. Figure 5 shows the concrete format of a predicate-advertisement packet; the various fields are described below.

receiver id. (16 bits)
downstream id. (16 bits)
distance (8 bits)
sequence num. (8 bits)
bit-vector pos. (5 bits)
max rate (9 bits)
predicate (variable size)

Figure 5: Packet Format: Predicate Advertisement

In the service model supported by DV/DRP, a receiver can dynamically change its predicate. The receiver maintains a sequence counter and associates a sequence number with the predicate it advertises.

A predicate is new to a node when the predicate advertisement arriving at that node refers to a new receiver, or when the predicate advertisement carries a sequence number greater than the sequence number for that receiver stored in the node's routing/forwarding table. In the first case, a new entry is added to the table. In the second case, the existing entry is updated with the new predicate, its primary path is updated, and all the alternate paths are cleared (and then rebuilt). In both cases, the predicate is transmitted to all neighboring nodes in the local-broadcast region using a single broadcast packet.

A predicate is obsolete, and therefore immediately dropped, if its sequence number is lower than the one already stored in the table. If a predicate arrives with the current sequence number for its receiver, then it is inserted in the path array according to its distance, and propagated if it is inserted as the primary path. A zero sequence number in the predicate is used to reset the sequence counter in the routing/forwarding table. A "null" predicate can be used to remove a route entry if a receiver is no longer interested in receiving messages.

Distance-vector updates determine a node's downstream links in the obvious way: an advertisement received by node x from neighbor d carrying a new predicate from receiver r (or a better path to r) would cause x to set its primary downstream link toward r to d, so x sets $n_r \leftarrow d$.

In addition to x's downstream links, the update should also cause d to add x to its upstream links for r, that is, d must set $U_r \leftarrow U_r \cup \{x\}$. Therefore, x must somehow tell d that it elected d as its downstream next-hop node toward r. Fortunately, DV/DRP can transmit this information to d with minimal overhead and without the need for additional messages simply by piggybacking onto the normal advertisement propagation. In particular, after the update, x (re)broadcasts the advertisement to its neighbors to announce the new predicate (or the shorter route to r). By including d's identifier as the chosen downstream node in the advertisement packet, x can also signal d to update its upstream links for r.

3.2 Forwarding

The forwarding state produced by the DV/DRP routing protocol is intended to "attract" matching messages toward receiver nodes. A node forwards an incoming message by matching the message against all the predicates in the routing/forwarding table. The basic idea is that a message m matching predicates $p_1, p_2, \ldots, p_i$ is forwarded to all the next-hop nodes $n_1, n_2, \ldots, n_i$. This matching process is also called *content-based forwarding* [7]. The concrete format of a message packet is shown in Figure 6.

3.2.1 One-Time Predicate Evaluation

Every node has complete knowledge of all the (small number of) receivers in the network. Therefore, the forwarding algorithm can apply predicate evaluation for a message m, once and for all, at the node where m first enters the network. The results of the evaluation are stored in a header field of the message, called the *receiver set*. At every subsequent hop in the path, the predicate evaluation can be avoided; all that is necessary is an evaluation of the receiver set against the forwarding information in the routing/forwarding table.

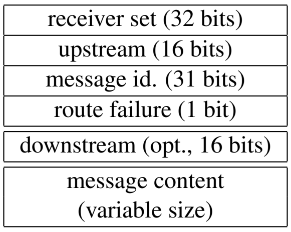


Figure 6: Packet Format: Message

3.2.2 Receiver Sets

One way to represent a receiver set is as an array of receiver identifiers. This approach is very simple, and can be used directly with existing statically assigned node identifiers, such as MAC addresses. The problem, however, is that it introduces a prohibitive communication overhead due to the potentially large size of the array. An approach to reduce the space requirements for receiver sets would be to use Bloom filters [3] to obtain a compact representation of a set of receiver addresses. Unfortunately, Bloom filters offer only a probabilistic membership test. Also, Bloom filters are not ideal for set-partitioning operations, which as we discuss below, are an essential part of this protocol.

Our solution is to use a simple fixed-size bit vector, where each receiver is represented by a dynamically assigned bit position. The obvious advantage of this solution is that it offers a compact representation that incurs no collisions. In principle, the fixed size of the bit vector could be a serious disadvantage, as it imposes an upper bound on the number of active receivers in a network. In practice, such a limitation does not affect many applications in sensor networking, where the vast majority of the nodes are senders, and only a few nodes act as receivers at any given time.

The disadvantage of the bit-vector solution is that it requires nodes to somehow negotiate their bit-vector position. Although static assignment of the position could be performed before deployment, a better approach is to support dynamic assignment. Fortunately, this can be done with a minor extension to the routing protocol, and with a simple local conflict-resolution protocol. Specifically, predicate advertisements are extended to carry the bit position of the receiver (see Figure 5). When a node r advertises a predicate p_r for the first time, r randomly chooses its own bit-vector position, b_r, among the ones that are not already in use by other receivers. The node then sends out the predicate advertisement, following the usual distance-vector protocol, with the receiver identifier r, the predicate p_r, and the bit position b_r.

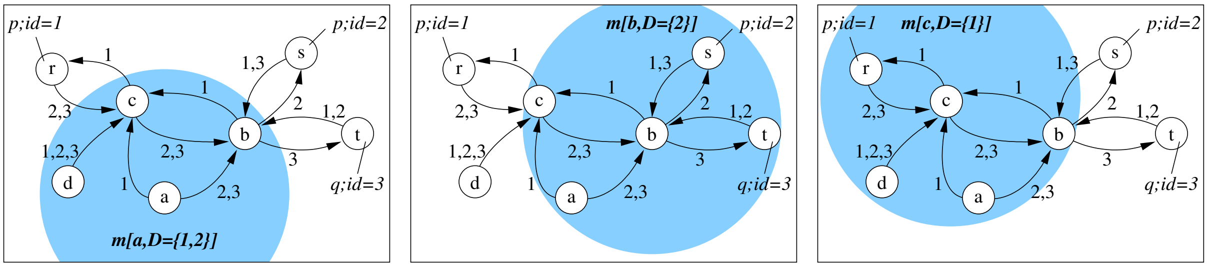
Because it takes some time for predicate advertisements to propagate through the network, it is possible that two or more nodes will pick the same bit-vector position. This rare event is detected by the conflicting nodes as soon as they receive each other's respective advertisements. When a conflict is detected, the node with the lowest node identifier (also carried by advertisements) is given priority and, therefore, keeps its chosen bit-vector position. All the other nodes in conflict are forced to choose a different bit-vector position, and resend their advertisements. This protocol is stable and converges to a conflict-free assignment of bit-vector positions.

3.2.3 Dynamic Receiver Partitioning

It it easy to see that content-based forwarding delivers messages to interested receivers. However, content-based forwarding alone will also exhibit duplicate deliveries and route loops. As an example, consider the scenario of Figure 4b. Assume a message m matching both p and q is sent by node v. Following the content-based paths of both q (dashed) and p (solid), the message gets to nodes s and u. Then, the copy that went to node s is sent again to node u, and vice versa, thereby creating duplicates. Loops also occur between nodes s and u, and between nodes r and u.

The loops can be avoided by a forwarding algorithm that does not send a message back to the node where it came from. However, it is easy to construct examples with loops of three or more nodes, where that simplistic control would not be effective. To avoid duplications and loops, we have designed a forwarding protocol that augments the basic content-based forwarding with a process called *dynamic receiver partitioning*.

We describe dynamic receiver partitioning by referring to the example of Figure 7. The diagrams show a small network in which a predicate p is advertised by nodes r and s, and a predicate q is advertised by node t. The edges in the graphs represent the forwarding state installed by DV/DRP throughout the network (but only



(a) (b) (c)

Figure 7: Dynamic Receiver Partitioning

downstream links are shown, for simplicity). Labels on each arrow indicate the receivers associated with that link. For instance, the edge labeled "2,3" connecting nodes a and b represents node a's downstream links toward s and t (and implicitly node b's upstream links for receivers s and t).

We observe that duplicate paths occur when two copies of a message m destined for receiver r cross two distinct branches of the content-based spanning tree of r, and are forwarded along both branches. Figure 7a shows a scenario that would cause such a duplication.

The example starts (Figure 7a) with a message m sent by node a that matches p but not q. Node a computes the receiver set $\mathcal{D} = \{1, 2\}$ and transmits a (local) broadcast packet using the format given in Figure 6. Figure 7a highlights the fact that the packet includes m, an identifier for the upstream node a, and the receiver set $\mathcal{D}$. Nodes b, c, and d receive the message because they are all within the transmission range of a. Upon receiving the packet, each node decides whether it should relay the message or whether it should drop it. A node does this by finding all the receivers in the receiver set of the incoming message that have a as an upstream link in the forwarding table.

If one or more such receivers are identified, the node proceeds to forward the message toward those receivers. Otherwise, the node drops the message. In the example, node b forwards the message toward receiver 2 because a is in b's upstream links set U_2. Similarly, node c relays the message towards receiver 1 because a is in its upstream links set U_1. Node d drops the message because none of the message's intended receivers has a in d's upstream links.

Before forwarding the message, a and b rewrite the receiver set header based on their selection of receivers. In particular, b transmits a packet with $\mathcal{D} = \{2\}$ (Figure 7b), while c transmits a packet with $\mathcal{D} = \{1\}$ (Figure 7c).

Notice how the receiver sets associated with downstream forwarders always form a partition of the upstream receiver set. This is because upstream links form a (directed) tree rooted at the receiver. Therefore, for any given node—a in the example—there can only be one downstream link toward each receiver, and only the next-hop node along that link will have a as an upstream node toward that receiver. The partitioning of the receiver set guarantees that duplicates and loops cannot be formed.

3.2.4 Forwarding Algorithm

Figure 8 sketches the content-based forwarding algorithm used in DV/DRP. The algorithm makes use of one of two procedures, depending on where within the network the forwarding is to take place. The first procedure, cb_drp_init_forward, takes a message as a parameter, and is executed at the node where the message enters the network. The second procedure, drp_forward, takes a message and a receiver set as parameters, and is executed at each subsequent hop in the routes to the receivers. The procedures are similar in structure, differing only in that the first procedure performs the full predicate evaluation, while the second instead performs only the receiver-set membership test.

3.3 Failure Detection and Recovery

Sensor networks are subject to failures that can be modeled as transient and permanent node failures. For example, wireless links exhibit fading and other types of intermittent interference, which appear as node failures to neighbors. Also, nodes may be permanently damaged by environmental agents, or they may simply run out of power, causing permanent failures. DV/DRP deals with failures by detecting errors in data transmissions and by reacting to those errors, at first attempting to route messages around transient failures, and then attempting to repair routes that remain broken due to permanent failures and asymmetries.

```
proc cb_drp_init_forward(message  $m$) {
  set<node> $\mathcal{D} := \emptyset$    // destination set
  foreach $r \in fwd\_table$ {
    if time() $- t_r > \Delta_r \wedge$ match($m, p_r$) {
      $\mathcal{D} := \mathcal{D} \cup r$
      $t_r :=$ time()
    }
  }
  if $\mathcal{D} \neq \emptyset$ {
    broadcast(new_packet($this\_node, m, \mathcal{D}$))
  }
}
```

```
proc drp_forward(packet  $p$) {
  node $upstream :=$ get_upstream_neighbor($p$)
  set<node> $\mathcal{D} :=$ get_destination_set($p$)
  foreach $r \in \mathcal{D}$ {
    if $upstream \in U_r \wedge$ time() $- t_r > \Delta_r$ {
      $t_r :=$ time()
    } else {
      $\mathcal{D} := \mathcal{D} \setminus r$
    }
  }
  if $\mathcal{D} \neq \emptyset$ {
    set_upstream_neighbor($p$, $this\_node$)
    set_destination_set($p, \mathcal{D}$)
    broadcast($p$)
  }
}
```

Figure 8: Sketch of the Forwarding Algorithm

3.3.1 Error Detection

When a node n forwards a message m toward final destinations $\mathcal{D} = \{r_1, r_2, \ldots, r_i\}$, it keeps m, m's identifier, and the destination vector $\mathcal{D}$ in its transmission buffer for a given period of time while it waits to hear forwarding broadcasts of the same message from next-hop nodes. We refer to the downstream forwarding broadcasts heard by the upstream node as *forwarding echos* because an upstream node and its downstream next-hop nodes share the same local broadcast region and, in the absence of link asymmetries, should therefore hear each others messages. For example, in the scenario of Figure 7, node a should hear the forwarding broadcasts transmitted by next-hop nodes b and c.

When n hears a forwarding echo of m with destination vector $\mathcal{D}'$, it clears all the destinations in $\mathcal{D}'$ from its local copy of $\mathcal{D}$. (Because destination vectors are implemented as bit vectors, this can be done with two simple bit-wise operations.) Node n does this for each of the forwarding echos it receives. When the timeout expires, n checks that $\mathcal{D}$ has been completely cleared, meaning that m must have been received successfully by all of the intended next-hop nodes. In that case, n can remove the information associated with m from its transmission buffer. Otherwise, if $\mathcal{D} \neq \emptyset$, n must assume that communication to the next-hop nodes remaining in $\mathcal{D}$ has failed.

In essence, DV/DRP employs what amounts to a store-and-forward protocol where the forwarding broadcast performed by a next-hop downstream node serves as an implicit acknowledgment message to an upstream node, and where the destination vector serves as a checklist for a completely successful forward.

3.3.2 Handling Failed Nodes

DV/DRP tries to route messages around failed nodes. In particular, it (1) sets a *route-failure* flag in the buffered message, (2) inserts the message identifier into a local message cache, (3) directs the message to one of the alternate routes, setting the downstream header field (see Figure 6), and (4) rebroadcasts the message following the same error-checking procedure. If all available paths have failed, it sends the message as a flood packet to the entire network.

When a node receives a message that has the route-failure flag set, the node must first check if that message is addressed to it and confirm the message identifier is not already in its cache. If the message is not addressed to the node, the message is simply ignored. Otherwise, if the node does not find the message identifier in its cache, then it proceeds to forward that message as usual, first trying its primary path, then its secondary, etc. If the message identifier is in the cache, then the node concludes that the packet has gotten trapped in a loop by following one or more alternate routes. In this case, the node resorts to sending the message as a flood packet. DV/DRP thus tries to route a message around failed nodes by first trying alternate routes, and resorting to flooding only as a last resort.

The amount of energy a flood consumes makes it a poor choice to deliver the data to interested receivers. Instead, the the flood is intended to notify the receiver that a sender is unable to communicate with any of it's recorded neighbors and that a re-advertisement is requested.

To avoid congestion and limit the power consumption due to flood packets from failed routes, we limit the maximum rate at which nodes are allowed to issue flood packets. Upon initially sending or forwarding a request to re-advertise, a node creates a time-stamp entry. If another request is made before t_{resub}, the minimal time between predicate re-advertisements, the new request is dropped to conserve power. This does not affect the fault tolerance of the network unless the timeout is greater than the period between failures, or higher than the average latency between the node that experienced a failure and the effected receivers(s). The maximum rate is a configurable protocol parameter.

In order to compensate for local congestion, the timeout used for error detection is dynamically adjusted according to the maximum response time observed from previous transmissions. In particular, in the absence of transmission errors, the timeout follows the observed maximum timeout time plus a small fixed interval, with exponential smoothing. In the case of errors, the timeout is quickly increased up to a maximum value.

When the message gets to its final destination, the route-failure flag will inform the receiver node that one or more branches of its forwarding tree have failed. After seeing a certain number of failures, the receiver node can attempt to repair broken routes by reissuing a predicate advertisement. The policy that controls this reactive re-advertisement process is a configurable parameter of the protocol.

3.3.3 Detecting and Avoiding Asymmetric Links

Sensor networks are characterized by asymmetric links. This is because antennas exhibit anisotropic transmission and interference patterns. Moreover, different devices exhibit different transmission ranges and sensitivities. These differences may be due to unavoidable manufacturing irregularities and/or transient or permanent environmental conditions. In any case, some degree of asymmetry in wireless links is unavoidable. In a particular network configuration, depending on transmission power and node density, these asymmetries might give rise to unidirectional links. It is therefore possible for a node x to receive transmissions from a node y, but not vice versa. This condition, whether stable or transient, has an adverse effect on a protocol like DV/DRP that builds forwarding paths by following in reverse the propagation of advertisements.

The approach we take in the design of DV/DRP is not to try to prevent asymmetries, but rather to detect them and to recover from their occurrence. The detection is based on the same mechanism used during forwarding. In particular, if communication to a downstream neighbor y fails repeatedly during forwarding, the node determines that y has a asymmetric link to it, and therefore blacklists y. The blacklist is then consulted by the routing protocol, specifically to avoid electing a blacklisted neighbor as an upstream next-hop node.

DV/DRP uses a simple blacklisting algorithm. The algorithm is based on a table that associates a "badness" index B_y to each neighbor y. A badness index above a given threshold $\overline{B}$ means that the neighbor is considered blacklisted. The badness index is zeroed upon successful transmission, and incremented upon transmission failures. In order to avoid blacklisting a neighbor in case of a transient failure, the blacklisting algorithm compresses message bursts during failures. In particular, the increment of B_y is conditional to the time of the last failure T_y, which is also stored in the badness table. In case of failure, an initial badness value of zero indicates that the previous communication attempt was successful, so B_y is immediately incremented. When $B_y > 0$, the node checks whether the current failure occurred more than $\overline{T}$ seconds after the previous one, and only in that case does it increment B_y.

3.4 Data Rate Limitation

By default, predicate advertisements do not impose a limit on the rate at which messages are delivered. This behavior, however, can be undesirable and even destructive, especially in large networks of sensors producing many messages matching the predicates. In these cases, a receiver and its surrounding relay nodes can be overwhelmed by the message flow.

In order to adapt the content-based service to the needs and capabilities of applications, DV/DRP allows receivers to specify a limit on the message delivery rate. This specification is given as an integral part of a predicate advertisement. A receiver r can advertise a predicate p_r and a minimum inter-arrival interval Δ_r ($\Delta_r = 0$ means no rate limitation). Minimum inter-arrival intervals are then propagated together with the advertisement, and recorded in the routing/forwarding tables.

The semantics of this rate limitation service is not intended to uphold fairness across sources. In some circumstances, sources may continually overshadow others, which would lead to a non-representative view of the network from the base station(s). Nevertheless, this approach places rate control where it is most relevant for applications, namely in the hands of the applications themselves, rather than the sensors.

4 Evaluation

We evaluated the DV/DRP protocol by implementing it in simulation and then analyzing its behavior under a range of scenarios. We also developed a basic proof-of-concept implementation on a real sensor platform. The primary goals of our simulation analysis were to: (1) assess the ability of DV/DRP to implement the content-based delivery model; (2) profile the costs incurred by DV/DRP, and thereby evaluate its applicability to networks of resource-constrained nodes; and (3) make sure that the protocol is stable and responds gracefully to application demands, as well as to network failures. In this section we focus discussion on our simulation analysis, and only briefly discuss the proof-of-concept implementation.

4.1 Experimental Methodology

The validity of our analysis depends on the level of detail and the realism of the abstractions used in the simulations. The most critical abstractions in the study of a network-level protocol such as DV/DRP are those regarding the underlying, lower layers of communication used in the network. Those layers are characterized by phenomena such as signal interference and anisotropic emissions that can have a dramatic impact on the entire network beyond the specific behavior of individual nodes.

In order to obtain realistic results, we have developed simulation models that include detailed physical models of radio propagation, antenna irregularities, and radio sensitivity. On top of the physical models, we have implemented a MAC similar to B-MAC [18], a realistic, lightweight MAC layer currently in use on some common sensor platforms (e.g., TinyOS[3] and MANTIS OS [2][4]). Compared to previous simulation frameworks, our models operate at the level of packets, as opposed to individual bits [16]. On the other hand, our models are dealing directly with the shared medium, offering greater realism and accuracy than more abstract, graph-based models that also attempt to simulate collisions and other position-dependent phenomena. Each of our models is described below, and are available for use and review.[5]

4.1.1 Propagation Model

Deployed sensor networks experience a plethora of propagation effects. Fading, constructive and destructive interference, multi-path propagation, and changing environmental phenomena can all seriously affect a node's ability to communicate. Our propagation model treats radio emissions as on/off, independent signals, starting at a given time, and having a given duration. This allows us to avoid the difficulty of having to precisely model wave forms. Emissions are characterized by their signal strength over the simulated space, and are independent in the sense that one emission does not modify the power of other emissions. Interestingly, emissions can model spurious noises or interferences caused by the environment, as well as the transmission of packets by nodes. In fact, this is how we model random communication failures.

In practice, for each emission e, the propagation model keeps track of the emission's center point $z_e = (x, y)$, its time interval $I_e = [t_0, t_1)$, and the radio/antenna model r_e of the emitter (discussed below in Section 4.1.2). Then, at a given time t, position z, and with a receiver radio/antenna model r, the model computes the combined effects of all relevant transmissions. We do not simulate the actual signal modulation, nor the electromagnetic effects produced by two or more interfering signals. Instead, each relevant emission e is considered in turn, with the following computation:

1. e's transmission power e_p is calculated by querying e's radio/antenna model r_e;
2. free-space propagation loss is then applied, based on the distance between z_e and z; and finally
3. an emission is ignored if its resulting power is below r's sensitivity threshold.

Out of all the remaining emissions, one is selected by checking the signal-to-noise ratio of the two strongest signals. The winning emission is handed off to the radio model of the receiver for further processing and possible delivery to the routing and application layer.

The advantage of this simplified propagation model is that it allows us to simulate collisions over packet transmissions, which play a crucial role in a network protocol, balancing efficiency and accuracy of the simulation.

[3] http://www.tinyos.net/

[4] http://mantis.cs.colorado.edu/

[5] http://www.inf.unisi.ch/phd/hall/software.php

Radio Parameters	
Minimum recv power	-77.0 dBm
Maximum dist (isotropic)	69.91 m
Signal-to-noise cutoff	4.0 dBm
Rx/Tx rate	19.2 Kbps
Rx power usage	19.7 mA
Tx power usage	17.4 mA

Antenna Parameters	
DOI	0.02
VDOI	0.5
VSP	0.1

Table 1: Radio and Antenna Parameters

The limitation of the model is that emissions are used to represent entire packets. Therefore, the model offers no information on interferences or other events at a lower granularity.

4.1.2 Radio and Antenna Model

Our radio model is based on the Chipcon CC2420 radio controller, since it is used on modern sensor nodes e.g., MicaZ[6] and TelOS[7]. In addition, we use the radio irregularity model developed by Zhou et al. [23] to model anisotropic antennas. We derived the parameters of the radio from the data sheets of the CC2420 and from the measurements published by Zhou et al.

The important settings for the CC2420 and radio irregularity model are listed in Table 4.1.2. Note that the *degree of irregularity* (DOI) is the mean of Zhou et al.'s measurements of ten different MicaZ nodes. The chosen *variance of degree of irregularity* (VDOI) and *variance in sending power* (VSP) are derived from the same study.

Figure 4.1.2 shows a ten-node topology we used in some of our experiments. This topology shows an example of how irregularities in antennas can result in several asymmetric links, even in small networks. Edges in the graph

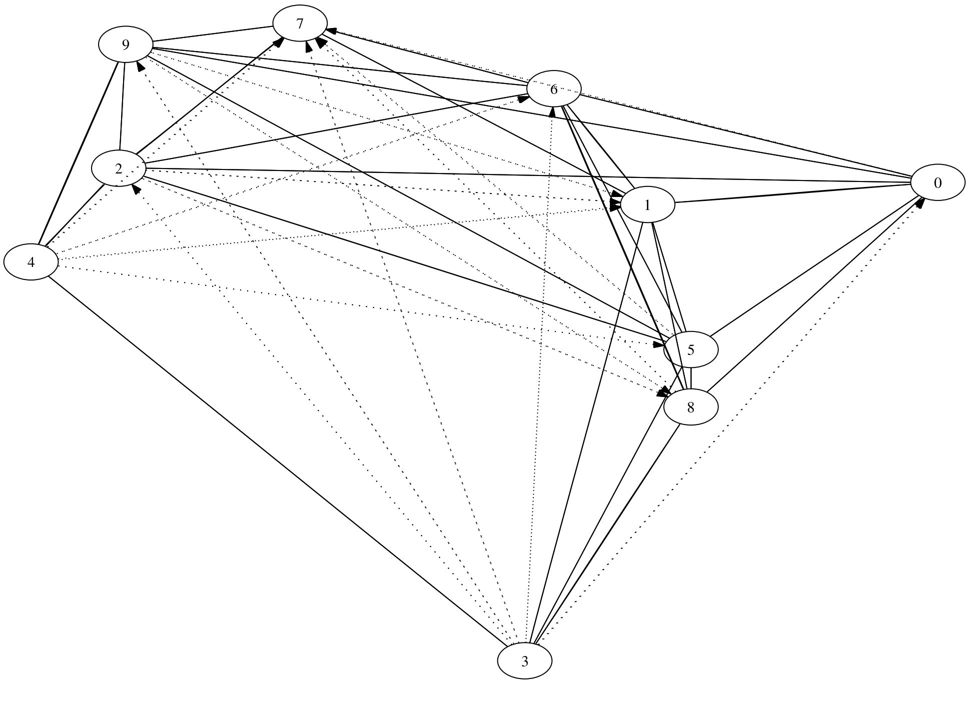


Figure 9: Ten Node Topology with Major Asymmetries

[6] http://www.xbow.com/

[7] http://www.moteiv.com/

connect pairs of nodes in which at least one of the nodes can transmit to the other. Specifically, solid lines represent bi-directional links, while dotted arrows represent asymmetric links. Notice that not all asymmetries result in one-way channels. In general, an asymmetric link indicates a pair of nodes whose probability of successful transmission in one direction is much higher than in the other direction.

4.1.3 MAC Layer

We implemented a lightweight MAC layer modeled after B-MAC [18]. B-MAC exemplifies the type of minimalistic MACs that are currently in use in sensor networks. Our MAC listens before sending, to make sure the channel is clear. If the channel is busy, it queues the packet and retries after a delay corresponding to one packet transmission. The queue of packets operates in a FIFO manner. As in the memory-constrained platforms in use today, we limit the length of the queue to three packets.

As noted above, the coarse granularity of the propagation model does not distinguish information within a single emission. One consequence of this simplification is that we are unable to simulate important details of many MAC protocols. Specifically, our MAC layer cannot simulate the pre-header phase exactly as specified by B-MAC because we cannot distinguish the pre-header signal from the actual data transmission. Our simulation model works around this limitation at a higher level, by delivering the signal to the MAC layer at once, at the end of the transmission, but only if there were no interferences during the entire transmission. We can accept this limitation under the practical assumption that the frequency of DV/DRP transmissions will remain well below the frequency of MAC-layer transmissions.

4.2 Experimental Parameters

Our experiments were conducted on random network topologies of 10, 25, 50, 100, and 250 nodes. Topologies were generated by placing nodes randomly (with uniform distribution) within a square target area proportional to the number of nodes. We maintain a constant density of nodes so as to obtain a constant average connectivity. Specifically, we used a density of 0.6 to 0.8 nodes per 1000 square meters that, combined with broadcast cells of a maximum 140 meters of diameter (see Section 4.1.2 for details), yields a mean connectivity of between 10 and 12 neighbors per node. Finally, we discarded all partitioned topologies. These topology parameters conform to those used in similar simulation studies [20].

On top of each network topology, we have defined several application scenarios. A scenario is a complete definition of the behavior of each node. In particular, a scenario defines: (1) when nodes send messages, and the content of those messages; (2) when nodes advertise predicates, and the content of those predicates; (3) when emissions unrelated to the network occur, and their duration and intensity; and (4) when nodes fail due to physical defect or other physical destruction.

The messages and predicates we used are similar to those shown in figures 1 and 2. The behavior of a sender is that of a Poisson process. Receivers exhibit two types of behavior: some advertise a predicate at the beginning of the simulation, never changing that advertisement, while others change their predicate advertisement periodically throughout the simulation. Spurious emissions are modeled by their location, start time, length, and power, with both start time and length model as Poisson processes, and power as a Gaussian. Finally, the parameter that controls node failure is the mean time until failure, also following a Poisson distribution.

4.3 Content-Based Delivery

The first experiment we conducted was designed to assess the main functionality of the content-based service implemented by DV/DRP. The scenario that defines this experiment is characterized by a network of 100 nodes, in which each node generates messages at a mean rate of one every 10 seconds. The network contains five receivers that change their predicates every 30 minutes. The scenario has no failures.

The results of this first experiment, plotted in Figure 10, are quantified by the percentage of *false negatives* and *false positives* over the total number of messages sent.

We record one false negative whenever a message does not reach a receiver that has a matching predicate at the time the message is injected into the network. Our analysis shows two primary causes of false negatives. The first cause is the natural latency of the network in propagating receiver predicates. At the rate advertisements propagate through the network, it can take up to 30 seconds to advertise a predicate to all the senders in a 100-node topology. The effect of the latency of advertisements is visible especially at the beginning of the simulation, although the same latency is also responsible for the small spikes observed later in the simulation.

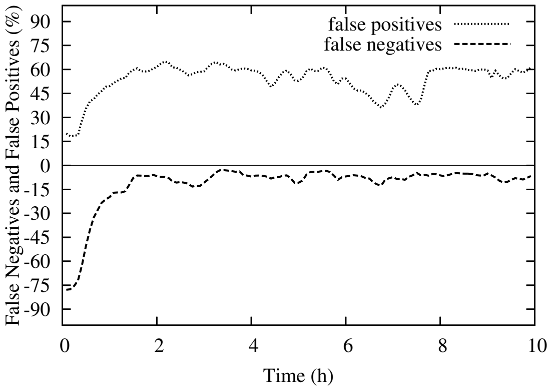


Figure 10: Functionality: False Negatives and False Positives

The second cause of false negatives is the presence of asymmetric links, which far outweighs the propagation effects. DV/DRP detects asymmetries and then avoids them. However, this process takes some amount of time. One reason is that each path correction requires a predicate re-advertisement from the receiver and, as we note above, predicate propagation is (intentionally) slow. Another reason is that the asymmetry detector is also intentionally slow, as it waits to see a minimum number of failure reports (10 in this case) before triggering a re-advertisement.

These two properties, along with the interaction required to weed out multiple asymmetric next hops, lead to a relatively slow convergence pattern. Fortunately, however, at steady state the network remains stable with a false negative rate at around 10%. We have run this simulation to over a week and confirmed that this behavior is stable

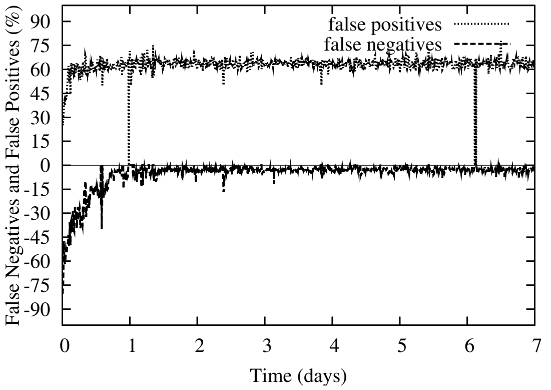


Figure 11: Functionality: False Negatives and False Positives Over One Week

(Figure 11).

False positives occur when a message reaches a receiver where it does not match the predicate currently advertised by that destination, or where a duplicate copy of the message was already received. The rate of false positives is computed over the total number of messages received by base stations. Specifically, if M is the number of received messages that correctly match the receiver's predicate, F is the number of receiver messages that do not match, and D is the number of duplicates, then the false positive rate is $(F + D)/(M + F + D)$.

False positives are also caused by the latency of the propagation of advertisements, in the case of a change of predicate. In fact, all messages matching an old predicate and sent before the sender receives the new advertisement are forwarded towards the receiver, only to be eventually discarded and accounted as false positives. However, the main source of false positives is retransmissions along secondary paths triggered by a failure to hear a downstream forwarding echo due to hidden-terminal collisions and/or asymmetries.

As shown by the graphs of Figure 11, the false positive rate stabilizes to around 60% of all message traffic. Of this 60%, roughly one third (20%) is attributable to advertisement latency. The remaining two thirds (40%) is caused by duplicate transmissions.

The primary conclusion we draw from this experiment is that DV/DRP is effective in implementing the content-based service with good reliability. Specifically, DV/DRP incurs a low rate of false negatives even in the presence of numerous failures and asymmetries. The relatively high level of false positive is, in essence, the communication overhead that DV/DRP spends to achieve this high level of reliability.

4.4 Control Traffic

The experiment above assess the main functionality of DV/DRP, which is to deliver to receivers all and only the messages matching their predicates. The following experiment analyzes the amount of control traffic incurred by DV/DRP in accomplishing this goal.

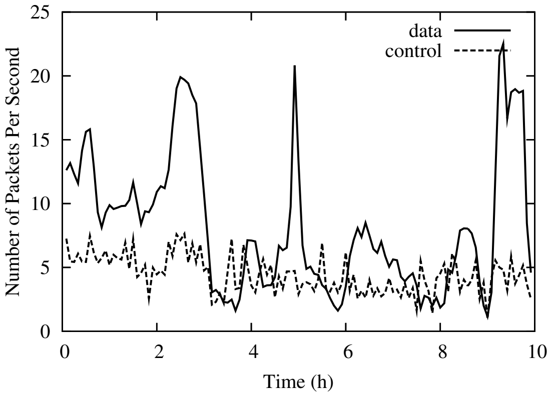


Figure 12: Data and Control Traffic

Figure 12 shows the data and control traffic for the same scenario as Figure 10. The values on the Y-axis represent the number of packets of each type going through the network at any given time. The solid curve represents the message (i.e., "data") traffic. The fluctuations are due to natural application behaviors, namely the changing predicates and the variation in message content produced by the sensors.

The second dashed curve represents the control traffic generated by predicate advertisements issued for route repair (see Section 3.3) and changing predicates. Changing predicates account for about 2 packets per second of control traffic in this simulation. The rest is due to re-advertisements issued in response to failure reports received at the base stations. Note the the overall level of control packets drops from around 6 to 3–4 per second about an hour and 45 minutes into the simulation, after many of the asymmetries are properly identified and blacklisted.

4.5 Scalability

The purpose of our scalability experiments was to assess the protocol as both the size of the network and the number of receivers increases. Figure 13 shows experiments conducted over such scenarios. Each node generates messages at a mean rate of one every 10 seconds and each receiver changes its predicate at a mean rate of once every 10 minutes. In practice, receiver predicates are likely to be much more stable, so the experimental rate serves to heavily stress the protocol. The graph shows that the amount of control traffic in the entire network at any given time grows linearly with both the number of receivers and the size of the network. In any case, even for 250 nodes and 20 receivers, control traffic is limited to about one packet every five second for each node.

4.6 Transient Failures

In this set of experiments we assess how well DV/DRP handles spurious radio emissions generated within the environment and affecting the network. Spurious emissions may come from diverse sources, such as 802.11 radios, wireless phones, and microwave ovens. We model each emission with a tuple $(start, end, emissionpoint, power)$.

We first wanted to see how DV/DRP would react in the face of interference from 802.11 hot spots. A heavily loaded 802.11 network would mostly likely impede all communications in the sensor network. Therefore, we modeled lightly loaded hot spots. Emissions are modeled as Poisson processes, with a mean inter-arrival time t,

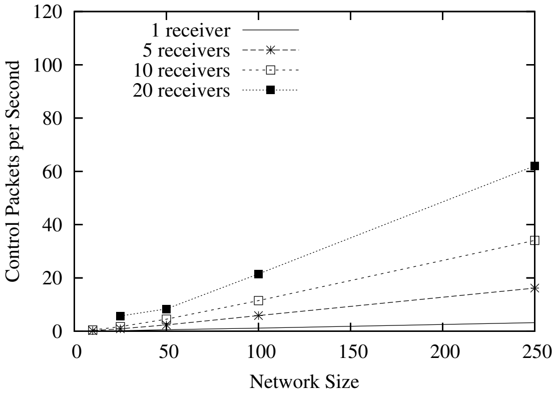


Figure 13: Scalability in Network Size and Number of Receivers

and with a mean duration modeled over a long-tail Pareto distribution with mean l. We set $l = 0.03$ seconds for all the 802.11 hot spot simulations.

We ran the experiments on the 100-node topology used in sections 4.3 and 4.4. Each sensor generates an average of one message every 10 seconds, and receiver predicates do not change. Figure 14 shows the percentage

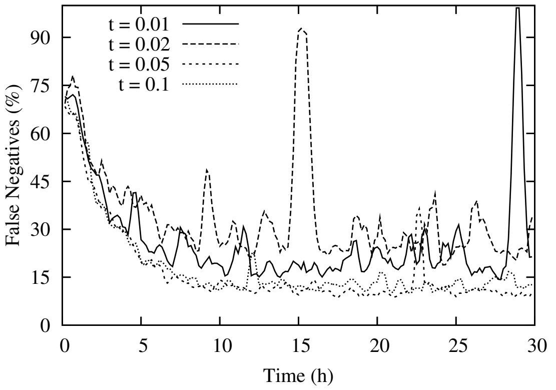


Figure 14: Behavior in the Presence of Short-Burst Interference

of false negatives over time, where each curve represents a different mean arrival rate for spurious emissions. As the arrival rate increases, the level of false negatives tends to drop. However, the correlation between the amount of interfering traffic and false negatives is clearly dependent, at least in part, on other factors. We need to investigate further to determine what these factors are.

The larger variations in false negatives are caused by longer emissions near a receiver node. Longer emissions correspond to larger files moving across the interfering 802.11 network.

Figure 15 shows the false negatives and false positives for the same workload, $t = 0.01$, both with and without spurious emissions. False negatives increase by 10–15% in the presence of spurious emissions, primarily due to hidden-terminal collisions affecting forwarding echos and ultimately flood packets (Section 3.3).

Figure 16 presents a different interference case. Instead of short and bursty traffic, we wanted to see if DV/DRP could adjust to long-term interference. In this case, emissions have a much longer duration, $l = 60$ seconds, and their inter-arrival time is much greater. Sources that provide this sort of interference include mobile phones and microwaves.

DV/DRP is able to route around long-term failures as long as another path is available. However, a small increase in false negatives is inevitable, simply because long-term emissions might interfere directly with senders, preventing them from sending packets through secondary paths or from flooding them as a last option.

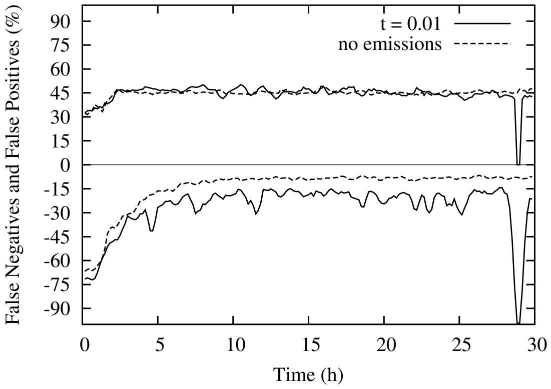


Figure 15: Short-Burst Interference Compred with No Interference

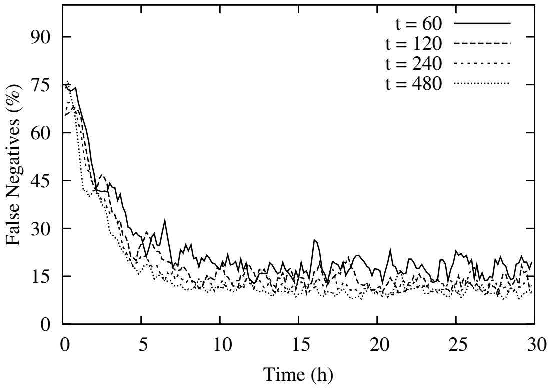


Figure 16: Behavior in the Presence of Long-Term Interference

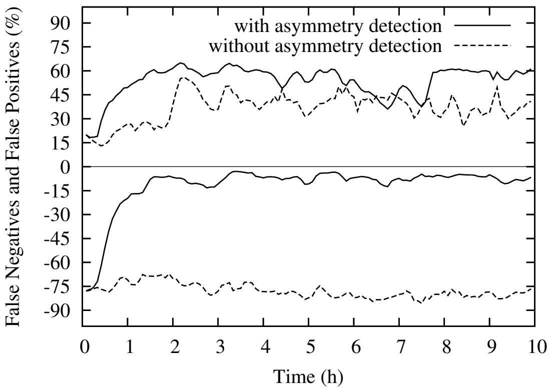


Figure 17: Benefit of Asymmetry Detection and Avoidance

4.7 Asymmetries

One of the primary features of DV/DRP is the ability to detect and avoid asymmetric links. As noted previously in Section 4.3, this feature is crucial for guaranteeing correct delivery. Figure 17 shows the results of a specific experiment that substantiates this claim. The scenario corresponds to the experiment of Figure 10. In this graph, however, we show the rates of false positives and false negatives both with and without asymmetry detection and avoidance. Notice that the rate of false positives is higher when asymmetry detection and avoidance is active. This is because asymmetry detection is based on retransmissions, which in turn may cause duplicate packets. However, the most interesting result of this experiment is that asymmetry detection and avoidance is effective in reducing the rate of false negatives from about 80% to less than 10%. We conclude that asymmetry detection is an essential feature of DV/DRP.

4.8 Permanent Failures

Protocols such as DV/DRP can exacerbate the problem of power-constrained sensor network nodes by directing a majority of total network traffic toward a small set of nodes, usually clustered around the base stations. In our next experiments, we studied the behavior of DV/DRP during periods when nodes are failing permanently due to power exhaustion. Using a given set of power metrics [1, 19], we calculated the amount of battery charge expended each time a message is sent or received. Figure 18 and Figure 19 summarize our results.

Each node was given the same limited amount of initial charge (750000 nAh), except for base stations, which were given unlimited power. The small amount of power was given to sensor nodes in order to limit the simulation run time.

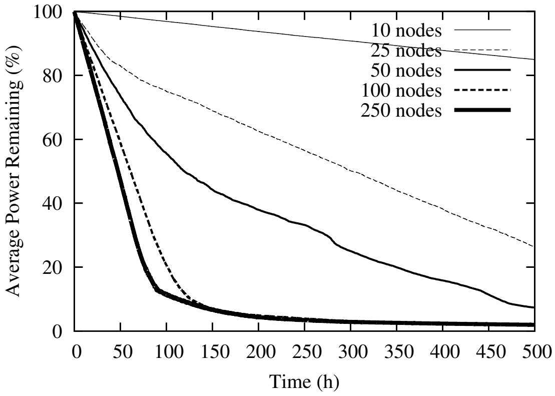


Figure 18: Power Drain for Different-Sized Topologies

Figure 18 shows the percentage of total remaining power in the network for topologies of different sizes. There are two major phases of network power usage. The network consumes power at a fast rate during an initial phase. This is a phase in which the network inevitably hits asymmetric paths and gradually routes around them. As the the network stabilizes in the second phase, the slope flattens out and power is drawn at less than a third of the original rate. This transition can be seen clearly in Figure 18 for both the 25- and 50-node topologies at 40 and 120 hours, respectively. The 100- and 250-node topologies run out of their limited power supply before they can stabilize, and the 10-node topology stabilizes too early to be seen at the scale of the graph.

The power-drain curve does not change significantly when nodes begin to run out of power, and then fail. This is because, by that time, all asymmetric links will have been blacklisted. Therefore, re-advertisements triggered by failures will be efficient in repairing broken routes.

A well-behaved protocol should also minimize the rate of undelivered messages until a base station has no neighbors left and has been completely isolated from the network. Figure 19 compares the power remaining in the entire network (dashed, smooth curve) with the rate of false negatives for a 50-node topology. Five crucial points can be distinguished in the graph, each marked with a vertical dotted line. The first line from the left marks the time when the network converges to stable routes and eliminates most of the asymmetric routes. At this point the power-drain curve changes to a gentler slope, as less re-advertisements are issued to repair routes. Then, 170 hours into the simulation, the first node around one of the two base stations runs out of power and fails. Notice that at this point, and for the following 166 hours, DV/DRP is highly effective at rerouting traffic, even as nodes continue to fail.

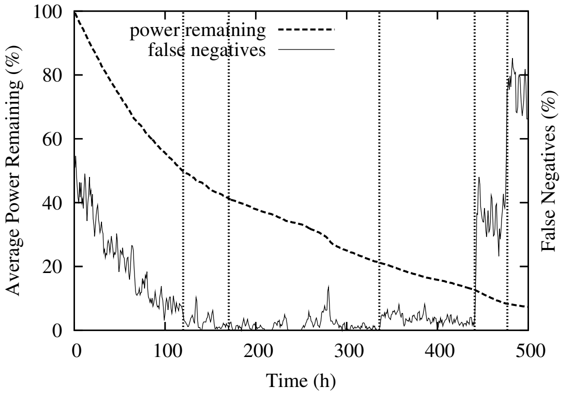


Figure 19: Tolerance to Permanent Failures

zAt hour 336 the network experiences the first permanent partition. At this time, the false negative rate shows an almost constant increase. Fortunately, in this case the two base stations remain within the largest partition, so the rate of false negatives remains under 10%. Finally, the last two lines, around 450 and 475 hours, mark the times when the two base stations become almost completely isolated from the remaining nodes, with some packets still arriving due to variance in sending power (Section 4.1.2).

We conclude that DV/DRP handles the degrading state of the network well, maintaining the rate of false negatives below 10% after stabilization, and well past the first critical failures. Even when moderate numbers of nodes close to the base stations fail, DV/DRP manages to keep false negatives under control. It is only after the base stations are completely and permanently isolated that the delivery service degrades significantly and is effectively interrupted.

4.9 Implementation

A prototype implementation of an earlier version [10] of DV/DRP was realized on a real sensor platform. The primary goals of this prototype were to confirm that DV/DRP can work on real hardware, and to profile the minimum resource usage of the protocol.

4.9.1 Platform

We choose the Mica2 sensor board[8] and used the Mantis OS (MOS) for the implementation. The Mica2 board has 4Kbytes of RAM, 128Kbytes of code space, 128Kbytes of flash memory, a CC1000 19.2 kHz radio, and a stackable sensor board interface.

MOS is an operating system designed by the Mantis Research Group at the University of Colorado [1]. MOS offers a POSIX-like API to access kernel functionality, and provides a fully multi-threading/multi-tasking environment. Unlike other sensor network operating systems [13], MOS allows users to write applications and kernel code in familiar languages such as C, using the standard GNU tools.

The Mantis OS CSMA MAC was used in all tests. This MAC does not contain an ACK functionality.

4.9.2 Functionality Tests

Several experiments were run to test the basic functionality of DV/DRP. Due to the limited number of nodes available, we tested only a few simple topologies, often with the originating sender of a packet in the same physical broadcast domain as the final receiver.

The first several tests we ran were designed to confirm the most basic functionality of the code. Two nodes were used, one a receiver and the other a producer. This test was designed to confirm several things: (1) the predicate would be correctly received by the sensor; (2) the forwarding functions would match a message against the predicate, forwarding the message toward the receiver; and (3) the receiver correctly identifies the message as matching a local predicate and delivers it to the application. After the first tests confirmed the basic functionality of the code and the algorithms implemented therein, we tested scenarios with multiple receivers and multi-hop paths, with up to two base stations advertising different predicates and two sensor nodes producing messages.

4.9.3 Resource Usage

On average, the DV/DRP implementation on MOS will result in the additional use of 944 bytes of RAM. This includes both heap and stack use, and amounts to 23% of the available RAM on the 4kbyte Mica2. Also included is the routing and forwarding table, which is large enough to hold up to five receiver subscriptions, containing a total of 10 filters and 20 constraints. The numbers of available subscriptions, filters, and other data are compile-time options to the implementation. Nodes can further shrink the pre-allocated data-structure pools to save space in more memory constrained environments.

Component	Size (bytes)
`net_thread` stack	160
`receiver_thread` stack	192
`generate_thread` stack	192
`dvdrp.o` heap size	528
`dvdrp.o` code size	3954

Table 2: Size of various DV/DRP components.

Table 2 gives the size of each component in the implementation. The first three lines in the table refer to the simple applications we used in our tests. The two lines at the bottom represent DV/DRP itself.

5 Conclusions and Future Work

We have presented a content-based networking protocol that solves many of the problems inherent in sensor network communication. DV/DRP maximizes proper message delivery and minimizes power consumption through techniques that maintain shortest paths, yet avoiding the flooding and loops found in previous protocols. Moreover, the techniques are designed to tolerate both transient and permanent network failures, as well as link asymmetries, which are inherent in the use of wireless communication. Extensive simulation studies substantiate our claims.

Work remains to improve and further assess DV/DRP. We have not addressed in-network aggregation or mobility, both of which can be critical capabilities of a sensor network. In particular, aggregation techniques could

[8] http://www.xbow.com/

add a level of fairness to the current rate-of-delivery scheme. This seems to be an extremely important area to explore as DV/DRP continues to mature.

We also need to explore improved local recovery mechanisms. Others have suggested the use of braided route techniques to create secondary paths [9]. Such paths could offer loop-free secondary paths, possibly eliminating the need for packet caches in DV/DRP. Another interesting research direction would be to look at cross-layer optimizations with transmission power control schemes. Such cross-layer optimizations could lead to to even greater energy efficiency. They may also be necessary for power-control algorithms that use global information, otherwise the route may use more (or less) hops than would be beneficial.

Acknowledgments

We thank Jeff Rose for his contribution to an earlier version of the DV/DRP protocol.

References

[1] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. MANTIS: System support for multimodal networks of in-situ sensors. In *2nd ACM International Workshop on Wireless Sensor Networks and Applications*, pages 50–59, Sept. 2003.

[2] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. *ACM/Kluwer Mobile Networks & Applications (MONET), Special Issue on Wireless Sensor Networks*, pages 263–279, Aug. 2005.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[4] F. Cao and J. P. Singh. Efficient event routing in content-based publish-subscribe service networks. In *IEEE INFOCOM 2004*, Hong Kong, China, Mar. 2004.

[5] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *IEEE INFOCOM 2004*, Hong Kong, China, Mar. 2004. To appear.

[6] A. Carzaniga and A. L. Wolf. Content-based networking: A new communication infrastructure. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, number 2538 in Lecture Notes in Computer Science, pages 59–68, Scottsdale, Arizona, Oct. 2001. Springer-Verlag.

[7] A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *SIGCOMM 2003*, pages 163–174, Karlsruhe, Germany, Aug. 2003.

[8] C. C. Enz, A. El-Hoiydi, J.-D. Decotignie, and V. Peiris. Wisenet: An ultralow-power wireless sensor network solution. *IEEE Computer Magazine*, pages 62–70, Aug. 2004.

[9] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 2002.

[10] C. P. Hall, A. Carzaniga, J. Rose, and A. L. Wolf. A content-based networking protocol for sensor networks. Technical Report CU-CS-979-04, Department of Computer Science, University of Colorado, Aug. 2004.

[11] J. Heidemann, F. Silva, and D. Estrin. Matching data dissemination algorithms to application requirements. In *First International Conference on Embedded Networked Sensor Systems*, pages 218–229, Los Angeles, California, Nov. 2003.

[12] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *33rd Hawaii International Conference on System Sciences*. IEEE Computer Society, 2000.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ASPLOS IX*, Cambridge, MA, 2000.

[14] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions On Networking*, 11(1):2–16, Feb. 2003.

[15] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *22nd International Conference on Distributed Computing Systems*, pages 575–578. IEEE Computer Society, 2002.

[16] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pages 126–137, Los Angeles, California, USA, Nov. 2003.

[17] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad hoc sensor networks. *SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[18] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys'04)*, pages 95–107, Baltimore, Maryland, USA, Nov. 2004.

[19] J. R. Polastre. Design and implementation of wireless sensor networks for habitat monitoring. Master's thesis, University of California at Berkeley, 2003.

[20] F. Silva, J. Heidemann, R. Govindan, and D. Estrin. Directed diffusion. Technical Report ISI-TR-2004-586, USC/Information Sciences Institute, Jan. 2004.

[21] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *First International Conference on Embedded Networked Sensor Systems*, pages 171–180, Los Angeles, California, 2003.

[22] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE INFOCOM 2002*, New York, New York, June 2002.

[23] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138, Boston, Massachussetts, USA, June 2004.