Title
Efficient and Practical Query Scoping in Sensor Networks

Authors
Henri Dubois-Ferriere
Deborah Estrin

Publication Date
2004

Efficient and Practical Query Scoping in Sensor Networks

Henri Dubois-Ferrière*
School of Computer and Communication Sciences
EPFL
Lausanne, Switzerland
henri.dubois-ferriere@epfl.ch

Deborah Estrin
Department of Computer Science
UCLA
Los Angeles, CA 90095
destrin@cs.ucla.edu

Abstract

A driving scenario for sensor networks is environmental monitoring: nodes gather data and send it back to a sink (i.e., a basestation with an internet connection or persistent storage) via a multi-hop tree topology. In order to form the data-gathering tree, and in order to configure the sensor nodes, the sink periodically disseminates messages into the network. For scaling, robustness, and load-balancing reasons, it is desirable to introduce multiple sinks and have nodes send data to their closest sink (under a metric, such as hop-count or energy cost) rather than all nodes sending to a unique sink. With multiple sinks, it becomes important to constrain dissemination of queries so that each sink does not flood the whole network. In this work we propose Voronoi scoping, a distributed algorithm to constrain the dissemination of messages from different sinks. It has the property that a query originated by a given sink will be forwarded only to the nodes for which that sink is the closest sink (under the chosen metric). Thus each query is forwarded to the smallest possible number of nodes, and per-node dissemination overhead does not grow with network size or with number of sinks. The algorithm has a simple distributed implementation and requires only a few bytes of state at each node. Experiments over a network of 55 motes confirm the algorithm's effectiveness.

1 Introduction

Environmental monitoring is a driving application for sensor networks. Some examples of environmental applications include habitat monitoring [1], micro-climate monitoring [2], agricultural monitoring [3], structural, seismic, and terrain monitoring, as well as surveillance and security applications. For all of these applications, it is necessary to measure physical phenomena with physically distributed sensors and to communicate it back to a remote location. This has been enabled by the emergence of inexpensive devices (*motes*) [4] which combine sensing, computation, and communication.

Such applications often use a *data gathering tree*, rooted at the basestation (or *sink*). Each sensor sends its data to its parent; intermediate nodes may perform some form of aggregation or compression [5] [6]. However, a single-sink system scales poorly with network size: data packets must traverse an increasing number of hops as the network diameter grows (thus reducing delivery ratio and increasing energy consumption), and nodes near the root of the tree carry a disproportionate amount of traffic. Of course there is an *infrastructure cost* to deploy additional sinks. While our purpose is not to derive an optimal relation between network size and number of sinks, we do assume that in most cases it will be worthwhile to have more than one. Thus we expect that many large-scale data-gathering networks will employ a tiered architecture with *multiple sinks*, placed at different points in the coverage area. For example, the deployment plan [2] driving this work is a habitat monitoring sensor network with approximately 100 Berkeley motes and 5 Stargate [7] microservers which connect to the internet via a 802.11 transit network.

Note that the tiered network model assumed here is not the same as that used in related work on clustering in wireless sensor and ad hoc networks [8] [9]. Specifically, the general clustering problem studied in these papers considers *jointly* the election of cluster-head nodes and the scope of their clusters, whereas here the sinks are determined *a priori* by virtue of special resources such as higher bandwidth radios, continuous energy sources, storage, etc; and therefore the algorithm itself is not responsible for selecting who can be a cluster head or sink.

This paper addresses the problem of efficient *query dissemination* and *tree construction* in a multi-sink data-

*The work presented in this paper was done while the author was visiting UCLA. It was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

gathering network. The two tasks are respectively distributing a query to sensor nodes in the system, and constructing paths from each source to a nearby sink. Note that because of inherent environmental dynamics affecting RF connectivity [10] [11] [12] [13], and because nodes may fail, paths to sink points change over time and must be dynamically configured, requiring continued dissemination of route and query messages over time.

The simplest possible solution, adopted by one-phase pull diffusion [14], is for each sink to disseminate *messages*[1] throughout the whole network. Nodes thus learn their next-hop and distance to every sink and can trivially choose the cloest one. However, this is redundant, and causes per-node overhead to increase linearly with the number of sinks. We therefore need a way to constrain the scope of queries originating at different sinks so that each flood does not propagate through the whole network. To summarize, this work is motivated by the following question: *How can we scope the dissemination of query messages and route advertisements from each sink to a subset of the network that is as small as possible, while guaranteeing that each node is reached by the message originating at its closest sink?*

This paper proposes a simple and practical algorithm called *Voronoi scoping*. The algorithm disseminates messages from each sink only to the subset of nodes that lie within the sink's Voronoi cluster, without involving nodes which lie outside of the cluster. The algorithm has zero overhead, requiring no explicit communications to establish voronoi clusters or otherwise set up algorithm state. It uses only local state and does not require nodes to have any knowledge of network topology, nor of number and location of sinks.

The remainder of this paper is organized as follows. Sect. 2 defines Voronoi scoping, some properties, and illustrates the algorithm at work. In Sect. 3 we review alternate approaches to query scoping and survey some related work. In Sect. 4 we describe our protocol implementation and discuss protocol aspects which arise due to the channel and topology dynamics of wireless networks. We show experimental results in Sect. 5 and finally conclude in Sect. 6.

2 Voronoi scoping.

2.1 Model and Assumptions

Our assumptions and model are driven first and foremost by practical systems, as this work is motivated by an field deployment. As such, our model matches that of existing proposals such as one-phase pull [14] or TinyDB [6].

[1]We use the generic term message to refer indiscriminately to query and route advertisement packets from a sink.

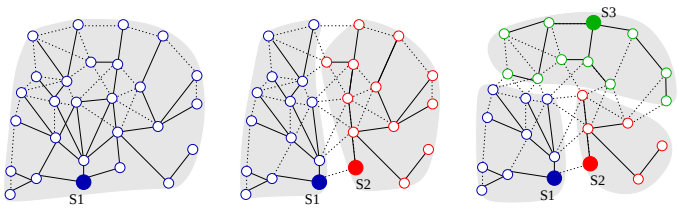


Figure 1: Voronoi clusters for $1, 2, 3$ sinks. Shaded regions represent the cluster associated with each sink under a hop count distance metric. The underlying topology is shown with dotted lines, and a Voronoi spanning tree is shown for each sink using solid lines.

Distance metric and closest sink data gathering. A data packet is successfully gathered when it arrives at any of the sinks in the network. For efficiency reasons, we would like nodes to send data to the *closest* sink. In other words each interior node in Voronoi cluster V_k sends its data to sink k. A border node may send its data packets toward either of the equidistant sinks.

Which sink is the *closest* depends on the underlying distance metric. The primarily metric considered in this paper is *hop count*; this results in choosing the sink which minimizes energy spent to deliver the data packet (assuming here that each node transmits at the same power). More sophisticated network metrics are also possible. For example, we could define the closest sink as the one which can be reached over a sequence of hops with highest cumulated energy. Indeed, if nodes consume energy at uneven rates, then it might be advantageous to take a longer path which traverses nodes with high remaining energy.

Notation. We represent our network as the graph $G = (N, E)$, where N is a set of $m + n$ nodes (containing m sinks and n sources), and $S \subset N$ is the set of sink nodes (with $|S| = m$). The set of weighted edges is denoted E, and we note $d_{i,j}$ the distance along the shortest path between i and j. With each sink k we associate the *Voronoi cluster* V_k containing the nodes whose closest sink is k. More formally, we define a Voronoi cluster V_k as:

$$V_k = \{i : \min_{j \in S} d_{i,j} = d_{i,k}\}.$$

We say that any node belonging to more than one Voronoi cluster (that is, a node having more than one equidistant closest sinks) is a *border node*, and a node belonging only to one Voronoi cluster (that is, a node having a unique closest sink) is an *interior node*. We call *Voronoi spanning tree* a tree T_k that spans all the nodes in V_k. Fig 1 illustrates Voronoi clusters and Voronoi spanning trees.

Tree formation. We assume a simple reverse-path flooding primitive is used to build the data-gathering tree (other possible approaches such as MDCS are mentioned in Sect. 3.3). Route advertisement packets contain a hop-count field

indicating the number of hops has traversed since leaving the sink. A node receiving an advertisement reforwards it, until all nodes in the network have received it. Duplicates are detected using sequence numbers and discarded. After the advertisement has propagated throughout the network each node knows its *parent* in the tree to its cloest sink. A node subsequently forwards its (and its children's) data to this parent. Beside distance, the choice of parent may also be based on both link quality or semantic (in-network processing) concerns [15]. Such questions are not the focus of this work; we simply note that Voronoi scoping applies regardless of the policy applied.

Joint routing and query dissemination. Query dissemination and tree construction are logically distinct operations. They might therefore be implemented separately, with queries being disseminated in one phase, and routing packets via another. In practice it is often more efficient to piggy-back queries and route advertisements in a same packet. We will assume such a model; of course Voronoi scoping can be used equally well if queries and route advertisements are disseminated via separate messages.

2.2 Algorithm

Our global objective is to constrain dissemination so that route advertisement and query messages are not forwarded beyond the originating sink's Voronoi cluster. From a node's local perspective, this means that a message originated by sink k should only be reforwarded if the node is an *interior node* of V_k. Specifically, a node knows that it is an interior node of V_k when messages from sink k reach i via fewer hops than messages from any other sink. If two sinks reach i via an identical number of hops, then i is a *border node* between these two sinks.

This leads to a simple distributed implementation: each node keeps a record of its closest sink and of the network distance to that sink. When a message arrives from a sink, the recipient checks if the distance traversed by the packet is less than the current estimate of closest sink distance. If so, the node updates its closest sink and parent entries and resends the message. A node also reforwards the message if distance traversed is equal to closest distance and the message came from the closest sink; this is necessary so that a subsequent message is forward by nodes in the originating sink's cluster.

The algorithm pseudocode is given in Fig. 2. It is slightly more general than the above description: messages are forwarded beyond the border of a Voronoi cluster up to an *overlap increment* of L, which is a fixed algorithm parameter set consistently across all nodes. As a not terminology, when we refer to Voronoi scoping with no further qualification, we refer to the algorithm with $L = 0$, which is the most restricted scoping. We remark that without the second

Algorithm *Voronoi*(k, d, n)
($*$ Called each time a query packet is received. $*$)
Input: A sink k, a distance d, a neighbor n
($*$ k and d are read off incoming packet header. $*$)
($*$ n is the neighbor which transmitted the packet. $*$)
($*$ d_min is the distance to closest known sink. $*$)
1. $\triangle \leftarrow d - d\_min$
2. **if** ($\triangle \leq 0$)
   ($*$ Update if closer or equal dist. than to current sink. $*$)
3. **then**
4. $parent \leftarrow n$
5. $closest\_sink \leftarrow k$
6. $d\_min \leftarrow d$
7. **if** $\triangle < L$ or ($\triangle = 0$ and $k = closest\_sink$)
   ($*$ Reforward if in Voronoi cluster or overlap region. $*$)
8. **then**
9. $increment\_distance\_on\_packet()$
10. $reforward\_packet()$

Figure 2: Voronoi scoping algorithm. Initially d_min is set to ∞. L is a fixed algorithm parameter. When $L = 0$, we have disjoint scopes; this is the setting we implicitly refer to as 'Voronoi scoping'.

condition ($\triangle = 0$ and $k = closest\_sink$) on line 7, only the first query from a sink would be forwarded by nodes in that sink's cluster. Indeed, when a node receives a second packet from a sink via the same route we have $\triangle = 0$; therefore without this check the packet would not be reforwarded. Note that this check is only relevant when $L = 0$.

We use a soft-state model [16]: after some timeout, closest sink and distance estimates are cleared to null values. This means that if a sink goes offline or gets disconnected from part of the network, nodes previously in that sink's Voronoi cluster will eventually discard their route to it and accept messages from the next-closest sink.

2.3 Example

We illustrate the algorithm in a network with 3 sinks and 20 sources in Fig. 3. Nodes are disposed in a grid for clarity; a real sensor network will typically have a less regular layout. In (a), no sink has yet sent a query message; nodes therefore do not know their distance to any any sink. Sink A is the first to disseminate a message. For each node receiving this message, A is the only (and therefore closest) known sink; therefore it is forwarded throughout the network, as depicted in (b). Then in (c) and (d) B (resp C) disseminate their queries which are only forwarded within V_B (resp. V_C). Finally in (e) each sink has originated one message and the network is in steady-state. The properties stated in Sect. 2.4 now hold; further queries will be disseminated only within the Voronoi cluster of their originating

sink. Fig. (e) shows two border nodes belonging both to V_B and V_C are in a darker region; all other nodes are interior nodes.

2.4 Properties

We now list some properties for disjoint Voronoi scoping ($L = 0$). These are *steady-state* properties, i.e. they hold only after each sink has originated one message (ie., in Fig. 3 (e)).

Property 1 : (*Coverage and Scoping*) If $i \in V_k$, then i receives packets disseminated by sink k. If $i \notin V_k$, i does not receive packets originated by sink k. Equivalently, packets are disseminated to all nodes in the Voronoi cluster of their originating sink, and not beyond.

Property 2 : (*Scalability*) Since a message is never forwarded beyond its Voronoi cluster, each node in the network transmits at most one message when every sink has originated one message. Therefore, per-node dissemination overhead remains constant independently of network size and number of sinks.

Property 3 : (*Memory*) The algorithm requires only keeping 2 pieces of state: closest sink and distance to that sink.

Property 4 : (*Distributed*) Each node decides whether or not to rebroadcast a flood (query or route advertisement) packet using only information which is local to that node.

Property 5 : (*Transparency*) Operation of the algorithm does not require sinks to behave any differently than they would when originating a global, unscoped flood. Sinks need not in any way keep track of other sinks going offline or coming up. In fact, a sink need not even know the existence or location of other sinks. This obviates the need for any inter-sink coordination mechanism and is key to the algorithm's simplicity.

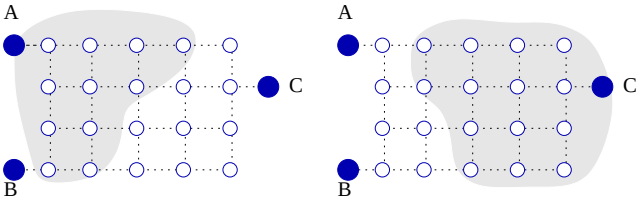
2.5 Higher order Voronoi scoping and failover

The algorithm in Fig. 2.2 considers, at each node, only the distance to the *single* closest sink. It can also be extended to take into account the N nearest sinks. We call this generalization order-N Voronoi scoping, and note V_k^N to be the order-N Voronoi cluster associated with sink k. Under this notation $V_k = V_k^1$.

In order-N scoping, cluster V_k^N is defined as the set of nodes for which k is *one of* the N nearest sinks. Implementing order-N scoping in a distributed is done similarly to order-1 scoping (Fig. 2), with the difference that each node must keep track of N closest sinks rather than 1 sink. We represent order-2 scopes in Fig. 4. This illustrates that $V_k^N \subseteq V_k^M$ for $N < M$, and of course $V_k \subseteq V_k^N$ for $N > 1$. In other terms higher-order Voronoi scoping will result in larger scopes and more dissemination overhead than regular Voronoi scoping. Of course if N is equal to or greater than the number of sinks in the network, order-N scoping will result in global flooding.

With higher order scoping, each node effectively keeps track of the N closest sinks. It is therefore useful to allow a failover function, for applications that require the ability for nodes to switch over to a backup sink available if the path to the current closest sink breaks (or if the sink itself fails). Order-N scoping thus constitutes a simple and cost effective means to ensure that all nodes have $N - 1$ available backup sinks.

Order-N scoping was not used for the data-gathering application that motivates this work because the application is loss-tolerant. Without end-to-end reliability, a source can not learn about upstream path breaks or sink failures; these failures are only corrected when a newer route advertisement overrides the current entry. Therefore, a source would not know when it is necessary to failover to another sink. In a data-gathering application with reliability semantics (ie, [17] [18]), order-N scoping can be used for rapid failover.



(a) Order-2 cluster around A. (b) Order-2 cluster around C.

Figure 4: Order-2 Voronoi clusters for sinks A and C, represented as shaded areas, under a hop-count distance metric. The order-2 cluster around node A (resp. C) contains all nodes for which A (resp. C) is the closest or second-closest sink (in hops). In comparison to Fig. 3 (e), we see that order-2 clusters contain more nodes than order-1 clusters.

3 Background and Related Work

3.1 Alternative scoping techniques.

We now survey other possible scoping techniques and discuss their respective merits and drawbacks.

Global dissemination. The simplest possible approach (used in one-phase-pull diffusion [14]) is to do no scoping at all, and have messages from each sink to disseminate its messages globally. Nodes then know their hop distance to each sink, and can trivially choose the closest to send their data. This is simple and robust: since a node receives messages from all sinks at all times, even pathological connectivity dynamics should not cause nodes to starve[2]

[2]We say that a node is *starved* when it does not receive any query due to overly restrictive scoping

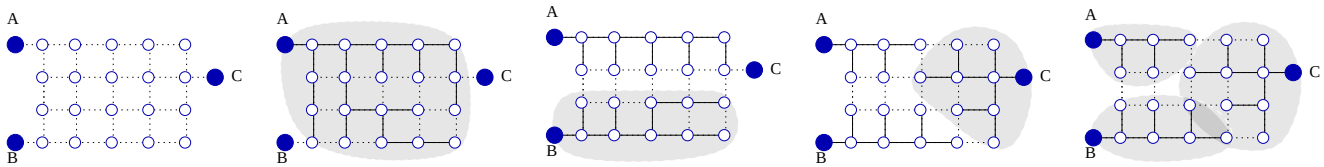


Figure 3: Voronoi scoping algorithm over a sensor network with 3 sinks (A, B, C) and 20 sources. Dotted lines represent radio connectivity, solid lines represent possible resulting voronoi spanning trees.

(we explore this further in Sect. 4.2). The drawback is of course that flooding overhead increases linearly with each additional sink. For this reason global flooding may be too costly for all but the smallest networks.

TTL scoping. A common scoping mechanism is time-to-live (TTL) scoping. Here, one might use this technique by placing a TTL value in messages originated by each each sink; the TTL is decremented at every hop, until it reaches 0 and the packet is dropped. However, choosing TTL values that guarantee coverage whilst avoiding excessive overlap of different sinks' floods is hard in a distributed and dynamic environment. Sinks might make a first *exploratory* flood, and have each node report its distance to the closest sink; then the TTL at each sink should be the distance to its furthest child. This gives the smallest possible TTL such that all nodes having a given sink as closest (ie, all nodes in the sinks Voronoi cluster) are reached by that sink's flood. But this exploratory flood is costly and must be repeated wheneve a sink fails or comes online; thus requiring that sinks run an additional protocol to keep track of other sinks status.

The second difficulty with TTL scoping is due to its isotropic nature: all nodes within the prescribed radius will be covered, in all directions. This is coarse and does not always correspond to the "shape" of a Voronoi cluster. For example, in the middle drawing of Fig. 1 both sinks must choose a TTL approximately equal to network diameter in order to cover all nodes. In such a situation both sinks essentially flood the whole network and TTL scoping can not help.

We ran some simulation experiments to quantify the minimal penalty due to the isotropic nature of TTL scoping. We considered a network of 500 nodes, varying the number of sinks from 1 to 25. Nodes were randomly dispersed under a uniform distribution. We then ran one round of dissemination (meaning that each sink originates one query) for either protocol and counted the total number of packet transmissions for TTL and Voronoi scopes. The TTL scopes at each sink were determined offline, with global knowledge of the network. For each sink, we determined the minimal TTL value with which this sink's flood would reach all nodes in that sink's Voronoi cluster. Even with these optimal TTL assignments, and even without counting the overhead to determine them, the cost of TTL scoping is greater than the cost of Vornoi scoping by a factor of 2 to 3 (Fig. 5).

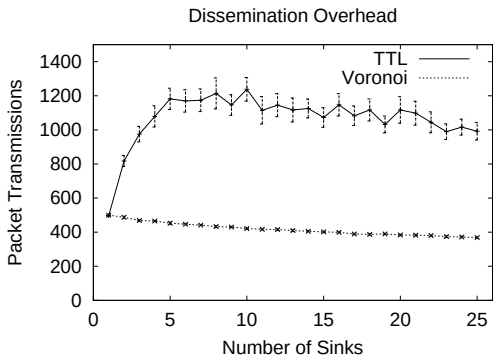


Figure 5: Comparison of dissemination overhead for Voronoi scoping and optimal TTL scoping, as a function of number of sinks, in a 500 node network. Even with optimal TTL assignments, the overlap between TTL fbods results in higher overhead than Voronoi scoping. The protocol overhead for sinks to obtain their optimal TTL assignment is not counted.

Geographic Scoping. Some data-gathering applications are of geographic nature, when the user requires information from a specific location. Of course, obtaining position information is not always feasible for tiny, constrained, sensor nodes. But in such applications, it is natural to use position information to disseminate queries only to the area of interest, rather than disseminate queries via Voronoi scoping or global dissemination.

Note that the Voronoi scoping can exploit geographic information by using euclidean distances between nodes rather than hop-count as a driving metric. In this case, the scoping algorithm would be approximating the *euclidean* Voronoi tessellation of the plane, where cluster boundaries lie on points which are at equal euclidean distance to two sinks.

Underlying flooding mechanism. This paper considers the use of Voronoi scoping in conjunction with a simple

flooding primitive. Another option would be to construct a minimum dominating connected set (MDCS) and use it for more efficient broadcasting [19] [20]. This optimisation is orthogonal to Voronoi scoping: MDCS reduces the number of packet transmissions to reach a fixed set of nodes, whereas Voronoi scoping limits the extent of floods from different points. We believe Voronoi scoping is also applicable when using a MDCS backbone, but a detailed investigation is for further work.

3.2 Application to 802.11 multihop networks

Because our work was driven by an ongoing sensor network deployment, this paper is primarily focused on the application of Voronoi scoping to sensor networks. We believe however that Voronoi scoping is also relevant in other contexts. In general terms, Voronoi scoping can be useful when a few "vantage points" in a network need to send a message in a cost-effective manner to all nodes, with the requirement that each node receive at least one message, preferably from closest originator.

We now outline another potential application, which is that of a multi-hop wireless access network. This is a network of fixed wireless (for example 802.11) nodes, with a subset of nodes having internet access and offering internet connectivity to the rest of the nodes. The MIT Roofnet project [21] is an operational example of such a network.

Independently of the routing protocol used, such a system requires gateways to periodically send an advertisement saying that they have internet connectivity. A similar question arises as for query dissemination in sensor networks: how should these advertisements be scoped? One solution (as described in [21]) is for gateways to flood advertisements to the whole network. This poses no problem for small-scale networks (for example roofnet has approximately 50 nodes, and 4 gateways). But if such a network is to span a large city section, with hundreds of nodes and dozens of gateways, then the overhead of periodic network-wide floods from each gateway will become significant.

Similarly to sensor networks, Voronoi scoping can be applied to contain this flooding overhead so that it remains constant at any network size. The underlying distance metric could again be hops, or it could be a more sophisticated metric such as ETX [22]. In this case, the border between Voronoi clusters would correspond to nodes with equal path quality to two sinks (expected transmission count) rather than nodes with equal hop distances. One would likely set the cluster overlap L (see Fig. 2) to be greater than 0 in order to have still some controlled amount of redundancy and overlap between different floods. One could also use order-N Voronoi scoping (Sect. 2.5) to ensure that each node always has a route to the N nearest gateways.

3.3 Related Work

Many problems derived from monitoring applications are the subject of ongoing research. To our knowledge, no prior work addresses the issue of query dissemination in a multiple-sink data-gathering context.

A large body of work in this area addresses ways to exploit the spatial correlations in sensed data in order to reduce communications overhead [5] [6]. This is also sometimes referred to as data fusion or in-network processing. This can be seen as focusing on the *upward* flow of data from sources to sink. Voronoi scoping is therefore complementary since it addresses the *outward* flow of queries and routing packets from sinks to sources.

Directed diffusion [23] is a framework for data dissemination protocols which is designed to support in-network processing and operates in a *data-centric* manner. Diffusion adopted a publish/subscribe API that isolates data producers (sources) and consumers (sinks) from the details of the underlying multi-hop routing algorithm. Since the original work on diffusion, some variations have been developped [14]; one of these is one-phase pull. Our implementation of Voronoi scoping was based on one-phase pull; we therefore describe it in more detail in Section 4.

TAG [6] is an aggregation service through which users express simple, declarative queries; TAG then distributes and executes them efficiently in a network of wireless sensors. TAG assumes an underlying routing substrate and is chiefly concerned with the efficient use aggregation operators, and how to coordinate the aggregation process between different nodes in the data gathering tree. Multiple sinks are not explicitly considered in [6]; we believe that Voronoi scoping would fit naturally within TAG if multiple sinks are introduced.

In related work [9], a clustering-based protocol (LEACH) is introduced with the objective of minimizing energy dissipation in sensor networks. In LEACH, a subset of sensor nodes are cluster-heads, and the other sensor nodes choose a cluster-head to which they transmit their data directly (no relaying). Once a cluster-head has received data from all the nodes in its cluster, it transmits the aggregate (and compressed) data to the basestation. Cluster-heads rotate over time in order to balance energy load. Other recent work on clustering in sensor networks includes [24] and [8]. A key difference of the present paper with these clustering algorithms is that they dynamically elect cluster head nodes at runtime, whereas here the sinks are determined *a priori* by virtue of special resources such as higher bandwidth radios, continuous energy sources, storage, etc; and therefore the algorithm itself is not responsible for selecting who can be a cluster head or sink.

4 Protocol and Implementation

We now delve into the more protocol-oriented aspects of Voronoi scoping, describing our implementation within directed diffusion; we also discuss some of the challenges and potential pathologies that might arise due to wireless network dynamics.

4.1 Protocol Implementation

We used the one-phase pull [14] (OPP) diffusion protocol as a framework within which to implement Voronoi scoping. We review it briefly before describing our changes.

In one-phase pull, sinks (subscribers) send query messages that disseminate throughout the network, establishing routing entries[3] from nodes to the originating sink. One-phase pull employs an attribute-based naming system to specify which data a sink is interested in; for example attributes may encode the request for a temperature reading at a given rate, or every time the magnetometer detects a vehicle passing by.

When a node receives a given query for the first time, it creates a routing entry associated with this query. It then reforwards the query; the tree associated with this query is thus built similar to the procedure described in Sect. 2.1. In order to prevent a 'broadcast explosion' queries are not immediately forwarded; a randomized timer is used. When a node receives a data packet from a child, it searches for a next-hop entry matching the data attributes represented in the packet. In other words, the packet is not addressed to a sink, but to any node with a matching subscription for this data type.

Diffusion state is maintained according to a soft state model, with periodic refreshes from sinks as long as a query is still alive. Queries are disseminated globally (as described in Sect. 3.1). Our core modification to one-phase pull was to implement Voronoi scoping. We then ran experiments using both the original version of one-phase pull (global dissemination of queries) and our modified version. This allowed us to compare fairly Voronoi scoping with global scoping, since all aspects of the protocol and implementation were identical save for the scoping of queries.

4.2 Connectivity Dynamics

Given vagaries of RF propagation such as interference, path loss, and fading, wireless connectivity is often time-varying and/or lossy, especially with simple, low-power devices such as sensor nodes. Measurement-based studies [10] [11] [12] [13] have shown that real networks behave differently from a unit-disk graph in many aspects. For one, wireless links have widely varying characteristics, with links exhibiting poor, intermediate, or good packet delivery ratios at any point in time. A second point is that connectivity is not isotropic, does not decay monotonically with distance, and is frequently asymetric. And a final challenge is that connectivity between any two nodes can vary substantially over time (even for static nodes).

Given these observations, we should emphasize that it is not a priori obvious that our scoping algorithm will behave as predicted when subjected to a real environment. As an example of how real RF conditions can affect the simplest network primitive [13] shows how trees obtained through a simple reverse-path flooding approach can have unexpected topologies.

One problem that might arise due to RF dynamics is *node starvation*. We say that a node is *starved* when, due to packet loss, it does not receive a query. Such a starved node delivers no information and is therefore useless. It is important to emphasize that the potential for node starvation exists whenever we have lossy links; it does not arise only as a consequence of using Voronoi scoping. Nonetheless, since Voronoi scoping results in fewer messages being disseminated, one might expect starvation to occur more often.

4.3 Link Estimation and Filtering

In wireless networks, selecting a parent by minimizing hop count only can result in selecting long-range, high-loss links, whereas a longer route with shorter-range, more reliable hops could be preferable [10] [22]. To address this problem, we run a *neighborhood tracking* module at each which estimate the quality of links to peers, by emitting periodic hello packets. The obtained values are then used to perform *link filtering*: nodes estimate the delivery ratio to and from their neighbors, and reject any packets arriving over links with quality below some threshold. More precisely, the link to a neighbor is characterized by the (estimated) incoming and outgoing delivery ratios. We set the incoming threshold to 40% and the outgoing threshold to 80%. The reason for the emphasis on outgoing quality is that route obtained is then used to send packets in the outgoing direction, toward the sink. A link below these thresholds can still be used if it is a node's best link; otherwise a weakly connected node would filter out all its neighbors.

5 Experimental Results

We performed a number of experiments over a network of 55 nodes. Our experiments were designed to mirror the overall workload of a data-gathering application: sinks periodically flooded query packets, and sources injected (synthetic) data packets. We compared Voronoi scoping with

[3] The terminology employed in [14] [23] for query message and routing entry is respectively interest message and gradient.

the global flooding approach (the default behavior of one-phase pull), because this technique is a simple and robust alternative.

We measured a number of packet-level statistics. The first quantity of interest was the flooding overhead (total number of query packets transmitted). Naturally, seeing that Voronoi scoping reduces overhead is not sufficient: an overly agressive scoping scheme might reduce flooding to the point that some nodes are starved and have no route to any sink. We therefore examined end-to-end application performance as well. In particular we counted the total number of data packets succesfully delivered to sinks. This can be seen as a *control metric*: if Voronoi scoping is too agressive and starves nodes, or results in poor routing trees, we would then see overall data delivery be adversely affected. We also compared some topological characteristics of the obtained routing trees with and without Voronoi scoping.

5.1 Experiment Setup

We ran experiments with 1, 2, 3, and 4 sinks with a network of 54 motes. For each setup we ran identical experiments with Voronoi scoping and with global scoping. We also ran a second, identical set of experiments with link filtering. There were therefore four protocol combinations for each sink configuration, giving a total of 16 different configurations. For each of these 16 configurations we ran 5 identical experiments, each lasting 30 minutes. This amounts to approximately 40 hours of run time for the results presented here.

Protocol Constants. For completeness we give all protocol constants. Each 30 minute run included a 5 minute warmup time during which statistics were not recorded. Sinks flooded a query at a fixed interval of 120 seconds (with a phase shift between sinks to avoid synchronisation). Each source generated data packets at random, exponentially distributed intervals; the mean was selected so as to have 1 data packet per second generated over the whole network.

These rates are high compared to what one might expect over the lifetime of a real deployment. The reason we set these data rates was to *scale down time* so that we would get more data points in a given duration. Note that this time-scaling is neutral to the comparative evaluation of the different protocols. A second point regarding traffic rates is they were low enough to place our experiments in a *non-congested regime*, meaning that the average traffic load should not be inducing frequent packet collisions. Note that running experiments in a congested regime would have been more favorable to Voronoi scoping: since it reduces overhead, collisions would be less frequent and data delivery would be higher relative to global flooding.

Nodes introduced a uniformly distributed random jitter between reception and forwarding of query and data packets (mean 2.4, 0.8 seconds respectively), in order to reduce collisions and avoid synchronization. Routing state had a timeout interval of 5 times the default inter-flood period (therefore, 10 minutes). Note that these jitter and soft-state settings are the defaults used by OPP; we did not attempt to otherwise tune these values for our experiments.

Hardware Setup. We used 55 Berkeley MICA1 motes, attached in a grid pattern to the ceiling of our laboratory over an area of approximately 20m by 20m. In order to maximise network diameter, the transmission power was set to the smallest value for which the network remained connected, resulting in a diameter of 7 hops. These motes are wired for power and have a serial-port connection back to a server running linux. This environment is part of the EmStar framework [25].

5.2 Results

Dissemination Overhead. The first quantity of interest was the overhead required to disseminate the periodic query messages from sinks. This overhead is represented in Fig. 6, comparing Voronoi scoping with global scoping both with and without link filtering. As expected, Voronoi scoping has same overhead as global scoping when a single sink is used. Then, with each additional sink, overhead increases linearly for global scoping; on the other hand it remains roughly constant for Voronoi scoping. Even at the moderate size of our network, this amounts to a substantial difference: with 4 sinks, Voronoi scoping reduces overhead by a factor of 3 when using link filtering, and 6.5 without.

Another observation is that for both global and Voronoi scoping, link filtering reduces packet transmissions by a factor of 2 to 3. This is a nice side-effect since the primary objective of link filtering is to improve the quality of routes obtained.

Topology characteristics. We now examine some topological aspects of the resulting data-gathering trees in order to see if Voronoi scoping introduces any distortion compared to the trees obtained with global flooding. Fig. 7 shows a snapshot of the data-gathering trees obtained in a 3-sink experiment. Note that the data-gathering trees are not static, due first to RF dynamics, and because of network density (many nodes have more than one possible parent with equal distance to a sink, and may change parent during the experiment).

Since trees are continuously fluctuating, we cannot directly compare trees across different experiments. One way to characterize these time-varying trees is to compute, for each node, the average distance (in hops) to the closest sink. We computed the time-weighted average distance for each node. We then sorted the nodes by average distance (under

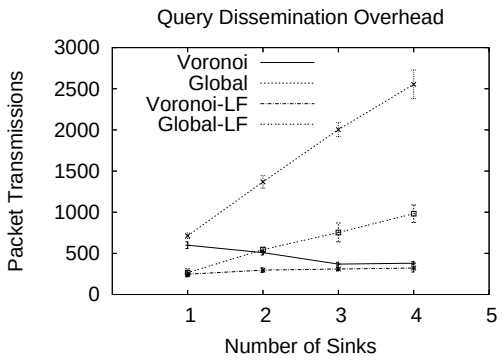


Figure 6: Comparison of query dissemination overhead for Voronoi scoping and global fboding, as a function of number of sinks. Overhead is measured as the number of query packet transmissions. With Voronoi scoping, additional sinks do not increase overhead, because query scopes are adaptively reduced by the network. With global fboding, each additional sink introduces a linear increase in overhead. Filtering out poor links scales down overhead similarly in both cases.

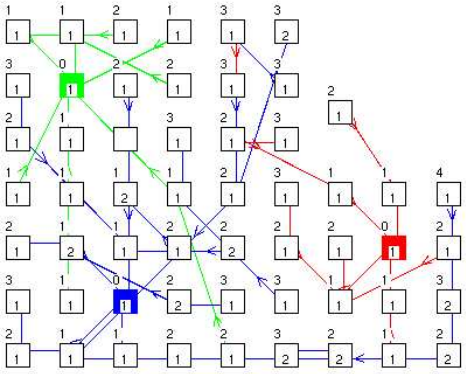


Figure 7: Voronoi trees using the EmView visualization component of EmStar. The integer value inside each box shows the number of sinks to which this node has a route, i.e., the number of sinks from which this node has received a query. The value above the box shows the number of hops to the *closest* (in hop-count) sink.

global flooding) and plotted the results for both protocols in Fig. 8.

These plots indicate that nodes have paths of similar lengths with Voronoi and global dissemination. Indeed, for 85% of nodes, the difference in average distance is less than 0.15; for the remaining nodes the difference is less than 0.25 (barring one outlier at 0.4).

Of course, average node-sink distances are only one characterisation of time-varying tree topologies; observing similar distributions for both protocols is not sufficient to state that the trees themselves are near-identical. But from an application perspective, this is an important metric since node-sink distances represent the number of hops that data packets will traverse toward the sink.

We also observe in Fig. 8 a plateau-like levelling at hop-counts 1 and 2, indicating that most nodes' route lengths remain quite stable. Some nodes have an average distances which is "far" from an integer value (for example 1.5). Such nodes therefore were less stable and alternated between closest sinks at different distances. The plots indicate that stability is not adversely affected by Voronoi scoping, since nodes are as close to an integer-valued average distance as with global flooding.

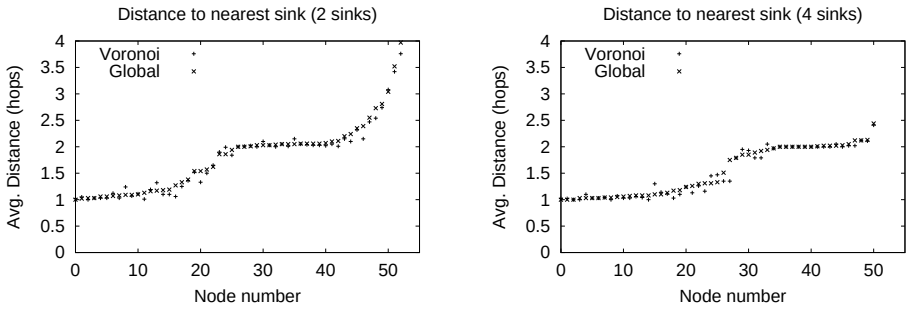
Application performance. The results on total dissemination overhead and tree topology characteristics, presented above, are important in order to quantify the *efficiency* of Voronoi scoping and get some insight on the resulting routes. But this study would not be complete without an examination of application-relevant metrics. In the context of an environmental sensor network, we consider the following two aspects as indicators of overall application performance.

The first aspect of application performance is *data delivery*. Counting the number of data packets succesfully delivered to sinks allows us to see, for example, if node starvation (Sect. 4.2) is occuring more often with Voronoi scoping. Data delivery is plotted in Fig. 9, for experiments both with and without link filtering. In both cases, we see that Voronoi scoping does not impact application performance. In fact the data delivery ratio appears slightly higher with Voronoi scoping; though this cannot be considered as definite evidence given the confidence intervals obtained.

The second aspect is *energy longevity*, or the time for which nodes have sufficient power to operate. Of course absolute energy longevity depends on many factors, such as node power, consumption, and the traffic workload imposed on the network; for this reason we do not quantify it. But qualitatively, since Voronoi scoping reduces communications overhead, it can only improve longevity.

6 Conclusions and Future Work

We have presented Voronoi scoping, a technique to control the dissemination of packets originated from multiple sinks in a network, in a way that minimizes overall flooding overhead. With Voronoi scoping, per-node flooding overhead remains constant independantly of the network size and number of sinks. This is particularly valuable in the context of sensor networks, where energy is the limiting resource. We implemented and tested Voronoi scoping over a network of 55 Berkeley motes; results confirm good efficiency compared to global flooding whilst maintaining a



(a) Network with 2 sinks. (b) Network with 4 sinks.

Figure 8: Average distance (in hops) to closest sink, as a function of node id. The nodes are sorted by increasing distance to their closest sink under global scoping. Both algorithms result in near-equivalent distances for each sink, indicating that the source-sink route lengths are not modified by introducing Voronoi scoping.

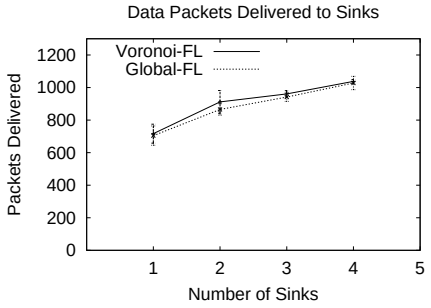


Figure 9: Comparison of data delivery for Voronoi scoping and global flooding, as a function of number of sinks. Delivery is measured as the total number of data packets successfully received at any sink. We see that voronoi scoping has similar delivery to global flooding, indicating that it provides routes of similar quality despite the reduced dissemination overhead. In all cases delivery improves with increasing number of sinks because the path lengths from node to closest sink are reduced, thus increasing the probability of delivery.

similar level of application performance.

This work focuses on the application of Voronoi scoping to sensor networks. Other applications are possible. As possible future work, we envision the application of this technique to multihop wireless networks [21], where a subset of nodes serve as internet gateways, and each wishes to broadcast route advertisements to those non-gateway nodes which are closest to it.

7 Acknowledgements

The implementation and experimental work described here leveraged a number of software and hardware facilites, notably the emstar software environment, the directed diffusion implementation, and the LECS laboratory ceiling array. We would like to thank the authors of these tools and in particular Alberto Cerpa, Jeremy Elson, Lew Girod, Fabio Silva, and Thanos Stathopoulos for their availability and help in using them. Many individuals made valuable comments that helped improve previous versions of this paper, including Razvan Cristescu, Deepak Ganesan and Lew Girod.

References

[1] Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless sensor networks for habitat monitoring. In: First ACM International Workshop on Wireless Sensor Networks and Applications. (2002)

[2] CENS/UCLA: Extensible sensing system: An advanced network design for microclimate sensing. http://www.cens.ucla.edu/Project-Descriptions/Extensible%20Sensing%20System/index.html (2003)

[3] et al, R.B.: The wireless vineyard. http://www.intel.com/labs/features/rs01031.htm (2003)

[4] Horton, M., Culler, D., Pister, K., Hill, J., Szewczyk, R., Woo, A.: Mica: The commercialization of microsensor motes. In: Sensors V. (2002)

[5] Cristescu, R., Beferull-Lozano, B., Vetterli, M.: On network correlated data gathering. In: INFOCOM, Hong Kong (2004)

[6] Madden, S., Franklin, M.J., Hellerstein, J., Hong, W.: Tiny aggregate queries in ad-hoc sensor networks. In: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI), Boston, USA (2002)

[7] Crossbow: X-scale single board computer and wireless networking platform. http://www.xbow.com/Products/XScale.htm (2003)

[8] Bandyopadhyay, S., Coyle, E.J.: An energy efficient hierarchical clustering algorithm for wireless sensor networks. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), San Francisco (2003)

[9] Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: HICSS. (2000)

[10] Woo, A., Tong, T., Culler, D.: Taming the underlying issues for reliable multhop routing in sensor networks. In: Proceedings of ACM Sensys, Los Angeles, USA (2003)

[11] Zhao, J., Govindan, R.: Understanding packet delivery performance in dense wireless sensor networks. In: Proceedings of ACM Sensys, Los Angeles, USA (2003)

[12] Cerpa, A., Busek, N., Estrin, D.: Scale: A tool for simple connectivity assessment in lossy environments. In: CENS Technical Report 0021. (2003)

[13] Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., Wicker, S.: Complex behavior at scale: An experimental study of low-power wireless sensor networks. In: UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013. (2003)

[14] Heidemann, J., Silva, F., Estrin, D.: Matching data dissemination algorithms to application requirements. In: Proceedings of ACM Sensys, Los Angeles, USA (2003)

[15] Madden, S., Franklin, M.J., Hellerstein, J., Hong, W.: The design of an acquisitional query processor for sensor networks. In: Proceedings of the ACM SIGMOD, San Diego, USA (2003)

[16] Clark, D.D.: The design philosophy of the darpa internet protocols. In: Proceedings of the ACM Symposium on Communications Architectures and Protocols (SIGCOMM), San Diego, USA (1988)

[17] Yogesh Sankarasubramaniam, Ozgur Akan, I.A.: Esrt : Event-to-sink reliable transport in wireless sensor networks. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC). (2003)

[18] Stann, F., Heidemann, J.: Rmst: Reliable data transport in sensor networks. In: Proceedings of the First International Workshop on Sensor Net Protocols and Applications, Anchorage, Alaska, USA, IEEE (2003) 102–112

[19] Das, B., Bharghavan, V.: Routing in ad hoc networks using minimum connected dominating sets. In: ICC (1). (1997)

[20] Alzoubi, K., Wan, P.J., Frieder, O.: Message-optimal connected dominating sets in mobile ad hoc networks. In: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC). (2002)

[21] Aguayo, D., et al: MIT roofnet implementation. http://www.pdos.lcs.mit.edu/roofnet/design/ (2003)

[22] Couto, D.S.J.D., Aguayo, D., Bicket, J., Morris., R.: A high-throughput path metric for multi-hop wireless routing. In: In the Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03), San Diego, USA (2003)

[23] Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: Mobile Computing and Networking. (2000) 56–67

[24] Younis, O., Fahmy, S.: Distributed clustering in ad-hoc sensor networks: A hybrid, energy-effi cient approach. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Hong Kong (2004)

[25] Elson, J., et al: Emstar: An environment for developing wireless embedded systems software. In: CENS Technical Report 0009. (2003)