

Balancing Push and Pull for Efficient Information Discovery in Large-Scale Sensor Networks

Xin Liu*

Qingfeng Huang and Ying Zhang

Dept. of Computer Science

Palo Alto Research Center (PARC)

University of California

3333 Coyote Hill Road

Davis, CA 95616, USA

Palo Alto, CA 94304, USA

Email: liu@cs.ucdavis.edu

Emails: {qhuang, yzhang}@parc.com

Abstract

In this paper we investigate efficient strategies for supporting on-demand information dissemination and gathering in large-scale wireless sensor networks. In particular, we propose a “comb-needle” discovery support model resembling an ancient method: use a comb to help find a needle in sands or a haystack. The model combines push and pull for information dissemination and gathering. The push component features data duplication in a linear neighborhood of each node. The pull component features a dynamic formation of an on-demand routing structure resembling a comb. The

*Corresponding author. The research of the corresponding author was supported in part by National Science Foundation through CAREER Award CNS-0448613 and Grant CNS-0520126, and by Intel Inc. through a gift grant.

comb-needle model enables us to investigate the cost of a spectrum of push and pull combinations for supporting query and discovery in large scale sensor networks. Our result shows that the optimal routing structure depends on the frequency of query occurrence and the spatial-temporal frequency of related events in the network. The benefit of balancing push and pull for information discovery is demonstrated.

Keywords: Information discovery, query, geographical routing, wireless sensor networks.

1 Introduction

Many emerging sensor network applications involve dissemination of observed information to interested clients. For instance, a sensor network might be deployed for enhancing soldiers' battle field situation awareness when visibility is low (either at night or due to smoke). A soldier might be interested in where the tanks are in the battlefield. The nodes detecting the tanks can periodically push (broadcast) the information throughout the network in anticipation of soldier's needs. This push-based information dissemination strategy is efficient when there are many soldiers in the network constantly in need of the information. On the the hand, when the demand for the information is low, a pull-based information dissemination strategy is more efficient. The soldier broadcasts a query when the information is needed. When nodes with requested information receive the query, they send the information to the soldier. Clearly, which strategy is more efficient depends on the relative frequency of queries and events. A natural question is as follows: can we combine the advantages of both push and pull strategies and build an efficient query-support mechanism that adapts to the frequencies of queries and events?

In this paper, we propose a "comb-needle" query support mechanism that integrates

both push and pull data dissemination, and analyze its performance in large scale wireless sensor networks. In the comb-needle model, each sensor node pushes its data to a certain neighborhood (resembling a needle) and the query is disseminated only to a subset of the network (resembling a comb), as illustrated in Figure 1.

Figure 1. A Comb-Needle Example.

One can view this query process as combing for needles in the sands or haystack. The most desirable property of this mechanism is that the combing structure is dynamic. Its granularity adjusts dynamically based on query and event frequencies to minimize communication cost. Combs are finer and needles shorter when the query frequency is relatively low compared to the event frequency, and vice versa. This dynamic property differs our scheme from the state-of-the-art information gathering schemes in the literature. Using this model, we are able to explore a whole spectrum of push and pull strategies.

The remainder of the paper is organized as follows: We first discuss related work in Section 2. In Section 3, we describe and analyze the comb-needle mechanism. We also discuss an adaptive comb-needle strategy, the reverse comb, fixed node query and sequential query. In Section 4, we present simulation results for supporting queries on both regular and random networks. Discussion and conclusion are in Sections 5 and 6, respectively.

2 Related Work

Many schemes have recently been proposed to reduce the cost of discovery and query in wireless ad hoc networks. The following approaches are the most relevant to this work.

The first approach aims to improve flooding efficiency by reducing the number of potentially redundant forwarding in the flooding process using neighborhood topological in-

formation [1, 2] or via probabilistic re-transmissions [3]. The goal of this approach can be viewed as to reduce the constant implicitly used in $O(N)$, where N is the number of nodes in the network. In contrast, the comb-needle scheme achieves $O(\sqrt{N})$ in the best case by balancing push and pull. In another related work [4], Braginsky and Estrin propose the interesting idea of rumor routing in which the query information and the event information are propagated in the network independently via random walk with a time-to-live parameter. A query “discovers” a relevant event when it meets the path traversed by the respective event agent. The comb-needle scheme could be regarded as a structured way to achieve the goal of rumor routing, in situations where node location information is available. Furthermore, in [5], Shakkottai shows that the query failure probability of rumor routing decays as $t^{-5k/8}$, where t is the average “propagation depth” of the query and event, and k is the number of rumor routing agents for each query and event.

The second approach reduces discovery/query cost by taking into account application semantics [6, 7]. For instance, discovery queries like “is there a tank in the field?” is potentially resolvable by only traversing part of the network in many cases. For this kind of queries, expanding-ring search and ACQUIRE [8] are shown to be more efficient than flooding. ACQUIRE considers a query as an active entity that is forwarded through the network either randomly or in some directed manner in search of a solution: nodes on the path that handle the active query use information from all nodes within d -hops in order to partially resolve the query. When the active query is fully resolved, the query forwarding process stops and a complete response is sent back to the query node. This type of scheme limits the scope of necessary flooding by recognizing the specific nature of the query. A comparison of ACQUIRE and comb-needle is given in [9]. Another outstanding example in this category is the Information-Driven Query Routing (IDSQ) scheme [10]. For instance, when tracking mobile

entities, sensor nodes can keep a certain memory. Therefore, when a query like “where are the tanks?” comes in, the underlying system can quickly route the query to the nodes with the most current information by following the trace of information existing in the network. This again eliminates the need for flooding in many cases. In [6], the authors investigate the relative benefit of push and pull diffusion for queries of continuous type and arrive the conclusion of a need for a family of protocols catering to different application requirements. This paper investigates and demonstrates the benefit of taking into account both the application and the environmental characteristics in designing efficient data discovery and dissemination protocols. The scaling laws for structured and unstructured information query are studied under storage and energy constraints in [11].

The third type of approach reduces the cost of search via efficient distributed indexing schemes. For instance, the authors present a distributed indexing scheme called Semantic Routing Tree (SRT) in [12]. SRT only supports query from a fixed node. It constructs a routing tree based on historical data, and thus applies well to slow changing sensor readings. In [13], the authors propose a distributed index for multi-dimensional data (DIM) which allows queries to be issued from any node. DIM achieves an average event insertion cost of $O(\sqrt{N})$ and an average query cost of $O(\sqrt{N})$ in most cases. It is also mentioned in [13] that DIM out-performs flooding as long as the ratio of the number of insertions to the number of queries is less than \sqrt{N} . Earlier exploration in this direction includes GHT [14] and DIMENSIONS [15]. The push component in our comb-needle scheme is similar in spirit to GHT and DIM in the sense that the data generated at a node is hashed/pushed to different locations. The difference is that in comb-needle, data is pushed to a geographically restricted area close to the source location, and in the previous methods, the data is pushed to nodes whose locations have no relation to the source location. By storing data in locations

close to the source location, the system has more efficient support for geographically scoped queries that are typical in many sensor network applications. Furthermore, the simplicity of this scheme also enables a simple rate-adaptive optimization strategy which no previous method could support. Our scheme achieves similar cost savings in average case to the distributed indexing schemes and is more efficient in extreme case. For instance, when each node in the network produces one observation per second, schemes like GHT and DIM could incur $O(N\sqrt{N})$ communication cost even though there could be no queries in the network, while comb-needle can have zero communication cost in theory if there is no query. Our approach achieves this by tying the data dissemination infrastructure to both the rate of data production and the rate of query.

Another line of related work is trajectory-based routing [16, 17, 18, 19, 20]. The main idea is to develop a cross-shaped trajectories to disseminate service information in the network. In [17], the authors propose to intersect horizontal and vertical lines for location service. A main routing structure (named spine/curve) is built in the network where other nodes initiate routes to intersect with the main structure for routing [16, 19]. In [18, 20], schemes are developed to realize various cross-shaped trajectories in a randomly-deployed network and the scalability issue is addressed in [20] by merging messages of duplicated sources. Our work can be considered as one alternative attempt to develop cross-shaped trajectories. Our focus is on the ability to adjust the coarseness of the trajectory (e.g., through the needle length and comb width) based on the query and event frequencies, and is complementary to the existing schemes. The schemes developed in [20] can be used to realize the comb-needle structure in a random network based on the adaptation studied in this paper. Finally, we aim to support the goal of discovering all related information in a sensor network while the objectives in the related work are to reach one of the (closest) servers in [18, 20] and to build

a routing structure in [16, 19].

3 Balancing Push and Pull

3.1 System Assumptions and Performance Metrics

There are many types of sensor network applications. In some applications, sensor nodes continuously gather information and report to one or more sink nodes. In this case, directed diffusion type of schemes work sufficiently well since the initial flooding cost can be amortized over the duration of the continuous information flow from sensor nodes to the sink node. In other applications, a sensor network is considered as a distributed data-base where information can be extracted only when needed. In this paper, we focus on the latter case where not all information is needed or useful. Communications of information that is not needed result in waste of resource and should be minimized when possible.

We consider the global discovery type query such as “where are all the tanks?”, and “what locations have a temperature exceeding 90 degree?”. For such queries, the whole network needs to be traversed by the underlying query resolution mechanism in order to get a complete response. Note that not all queries require a global discovery. Queries like “is there a tank in the region?” could only traverse part of the network (up to the point where the answer is positive, which we will discuss in Section 3.7). Furthermore, a query such as “what is the temperature in Room 123” can be directly sent to sensor(s) in the related location.

We aim to support both mobile and stationary query nodes. The scheme applies to applications where a query entry point can be anywhere in the network and occur at any time. A

major application of such query generation mode is to support mobile information-gathering agents (mobile sinks) or hierarchical networks where higher hierarchies are more intelligent and may demand information. Examples of mobile sinks include fire-fighters gathering information about dangerous spots and soldiers querying locations of tanks. However, in this paper, we assume that the speed of the mobile node is much smaller than that of communication in the sense that disconnection does not happen during a query process. For instance, a fire-fighter moves at a speed of 5m/s and a query communication process takes 100ms. The fire-fighter only moves about 0.5m during the query process and we assume it can still communicate with the original query entry point.

We assume all nodes in the network have information on their own locations as in [14]. In many sensor network applications, such as target tracking, intrusion detection and monitoring, data is useful only when the event location is known. For example, the locations of dangerous spots are more important to fire-fighters than the mere information of the existence of dangerous spots. Such information can be obtained from either GPS devices or other localization techniques being developed for sensor network applications.

We assume that the network is ad hoc and uniform in the sense that all nodes are equivalent and the network does not have built-in hierarchy. We also assume sensor nodes are stationary. We start our discussion using a regular grid network with reliable links for easy illustration and analysis. We discuss the scheme in the cases of random networks and unreliably links in Section 5. We also present simulation results of such cases in Section 4.

In this paper, we use packet-hop as a metric to measure communication efficiency and thus an indication of energy consumption. For simplicity, we assume that the size of a query packet is the same as that of a data-duplication packet. The assumption is valid when the actual information bit is small and the physical layer overhead dominates, and can also

be easily extended. We also ignore the energy consumption of query processing and data storage.

3.2 Building Comb-Needle

The push-pull query support scheme proposed resembles the action of combing for needles in a haystack or in a pool of sand, and is thus dubbed as “comb-needle”. The comb-needle query support model combines both push and pull in the following way: a sensor node that detects an event of potential interest pushes its data or data pointer of the event to a certain neighborhood (resembling a needle) and a query node disseminates its request to a subset of the network (resembling a comb), as shown in Figure 1.

As an example, we consider a grid network with n^2 nodes located at (i, j) where $0 \leq i, j < n$. We assume each node can communicate with 4 nearest neighbors. When a sensor node detects an event, the sensor node sends its update (on the event) to $(l - 1)$ of its vertical neighbors and thus build a vertical needle of length l . For instance, node (i, j) will send its state update to nodes $(i, j + 1), (i, j + 2), \dots, (i, j + l/2)$ and $(i, j - 1), (i, j - 2), \dots, (i, j - l/2 + 1)$. Fig. 2 illustrates a scenario where the node in the middle duplicates its data to four neighbors (i.e., $l = 5$).

Figure 2. A comb-needle structure where $l = 5$ and $s = 3$.

When a query is generated, a comb query structure is built. Assume a query is generated at node (i, j) . The query is sent vertically from (i, j) to (n, j) and to $(0, j)$. Then the query is also fanned out horizontally from nodes $(i, j), (i \pm s, j), (i \pm 2s, j), \dots$, where s is the inter-spike spacing or combing degree. The resulting routing structure looks like a comb. The scheme is illustrated in Figure 2.

Note that each event generates one needle and each query one comb. Combs and needles are generated independently. Furthermore, multiple events (queries) will generate multiple needles (combs) independently. For instance, a mobile agent may generate a series of queries at different time instances at different locations, and thus a series of comb structures along its trajectory. A comb may discover zero or one or multiple needles that matches its query. Similarly, multiple combs with the same query can discover the same set of needles (e.g., two soldiers are asking for tank locations at the same time). The data discovered by the comb is reported back to the query node. Nodes on the comb structure do not need to store these packets. Although it is possible to consolidate multiple simultaneous combs at the cost of additional signaling overhead, we do not consider it here for simplicity of analysis.

3.3 Analysis of the Combing Strategy

In this section, we analyze the performance of the comb-needle structure using a regular $n \times n$ grid; i.e., $N = n \times n$ where N is the total number of nodes. We assume that events occur uniformly in space and time across the sensor field. The query entry point can be any node in the network with equal probability. We define the following parameters:

f_q : the arrival frequency of discovery queries;

f_e : the arrival frequency of relevant events;

f_d : $f_d = f_e/n^2$, the arrival frequency of relevant events per sensor node.

A life-time parameter, τ , can be included in the query message to indicate its time-window of interest. A small value of the life-time implies that the query is interested in “current” events. The value of the life-time parameter depends on application scenarios. A fire-fighter

may only be interested in knowing the current dangerous spots. A maintenance crew may be interested in any potential signs of mal-functions in the last two hours (e.g., temperature over 100 degree in the past two hours). We assume that the memory space is sufficient to support the desired application. In addition, if memory is scarce, data duplication can be reduced to data pointer duplication. In other words, each node only needs to store a pointer to the actual location of the information (the node that detects the event) instead of whole information of the event.

The communication cost for each query can be estimated as follows. Each data duplication builds a needle of length l and thus incurs cost

$$C_l = l - 1. \tag{1}$$

We note that the cost depends on whether unicast or broadcast is used. In many sensor network applications, sensor nodes sleep most of the time. They wake up and communicate based on a localized schedule. In this case, it is reasonable to assume unicast because the node above and the node below may not wake up at the same time (otherwise, global synchronization is required). The cost of building a needle of length l is $l - 1$. When broadcast is feasible (e.g., due to large wake periods), the transmission cost is $l - 2$ if we ignore the cost of link layer acknowledgment. If link layer acknowledgment is required, then the cost is between $l - 2$ to $l - 1$. We note that the scale of the result will not change and $l - 1$ is the conservative estimate.

The query dissemination cost C_{qd} is

$$C_{qd} = n - 1 + (n - 1)(\lfloor \frac{n - 1}{s} \rfloor + 1). \tag{2}$$

This is the cost to build one vertical query line and multiple horizontal query lines. Note that if $s = l$, the query dissemination reaches its finest level, in the sense that it can collect

event information from all nodes; i.e., every needle will be discovered by the comb. The values of s and l are determined in the comb-needle structure.

The cost of query response depends on the reply scheme including data aggregation strategies used, if any. We consider the simplest reply scheme without data aggregation, which makes the result more conservative. Each node on the comb replies immediately if it has matching data, including data on its own detected events and those pushed to it from neighboring nodes. The reply paths are the reverses of the comb growing paths for the simplicity of analysis. In reality, the reply path could be a different one generated by any ad hoc geometric routing algorithm, since the location of the destination is known.

The average query reply cost for each relevant event is αn , where $0.5 \leq \alpha \leq 1$ is the distance factor reflecting the positions of the query node. For instance, if the query node is at the corner (e.g., node $(0, 0)$), then $\alpha = 1$. To elaborate, the distance from a node (i, j) to query node $(0, 0)$ is $(i + j)$. The average distance (averaging over all locations of events) to node $(0, 0)$ is thus

$$\frac{\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (i + j)}{n^2 - 1} = \frac{2n \frac{n(n-1)}{2}}{n^2 - 1} \approx n. \quad (3)$$

On the other hand, if the query node is in the middle (e.g., node $(n/2, n/2)$), then $\alpha = 0.5$. In general, if the query node is close to the center of the network, the value of α is small, and vice versa. When the query node is moving and/or there are multiple query nodes, then the value of α depends on the mobility pattern and the distribution of the query nodes. In all cases, $0.5 \leq \alpha \leq 1$.

On average, there are $f_e \tau$ events to be replied, each with an average cost of αn . Therefore, the total expected reply cost is

$$C_{qr} \sim \alpha n f_e \tau, \quad (4)$$

We consider the query reply cost as “inherent”. Consider the case where a genie knows exactly where to send all needed information. Then C_{qr} will be the total communication cost. We cannot reduce it because these packets contain required information to the query node. On the other hand, we can consider C_l and C_{qd} as the join cost to find out where the required information is (i.e., what the genie knows and we do not). Our objective is to minimize this part of the communication cost.

In summary, the total cost per query consists of data duplication cost, comb-building cost, and reply cost. Combining Eqs. (1), (2), and (4), the total communication cost per query is

$$C = C_{qd} + C_{qr} + C_l \frac{f_e}{f_q} \quad (5)$$

$$\approx \alpha n f_e \tau + 2(n-1) + \frac{(n-1)^2}{s} + s \times \frac{f_e}{f_q} \quad (6)$$

Therefore, s and l are the control parameters of the comb-needle structure. As discussed earlier, $l = s$ is required to guarantee that a query meets all relevant events. Substitute it into Eq. (6) and take a derivative over s . We have

$$s_{optimal} \approx (n-1) \sqrt{\frac{f_q}{f_e}} \approx \sqrt{\frac{f_q}{f_d}}, \quad (7)$$

where $f_q < f_e$. The minimum communication cost of comb-needle is

$$C_{optimal} = \alpha n f_e \tau + 2(n-1) + 2(n-1) \sqrt{\frac{f_e}{f_q}} = O(n) = O(\sqrt{N}). \quad (8)$$

The result indicates that when we have more queries and less events/data, the comb width s and the needle length l should be larger, i.e., more push and less pull is better in such scenarios. On the other hand, if the data frequency is relatively high compared to the query frequency, the comb should be finer and needle length shorter.

A frequently used query dissemination is the flooding-based querying (FBQ)[8]. In FBQ, the underlying query support mechanism floods the request to the whole network. It is thus a pull-based strategy. The baseline query cost (using FBQ) per query is $O(n^2)$ because each query is sent to all n^2 nodes. FBQ does not have a data duplication cost. The query response cost is the same as in comb-needle. So, comparing to the naive flooding based approach, this comb-needle scheme dramatically reduces query cost to $O(n)$. Note that the possibility of achieving this cost reduction partly relies on the knowledge of the expected event and query frequency.

We note that some approximations are involved in the analysis. We ignore boundary effects, e.g., whether there are $\lfloor n/s \rfloor$ or $\lfloor n/s \rfloor + 1$ comb spikes. We also approximate $n - 1$ as n in certain cases. Both approximations have little overall impact when n is relatively large.

3.4 Reverse Comb

Note that the scenario shown in Figure 2 is a global-pull-local-push one. If $f_e \geq (n - 1)^2 f_q$, then $s = 1$, which is equivalent to the pure pull strategy that flooding the query message in the network. On the other hand, from Eq. (7), we have $s_{optimal} > n$ when $f_q > f_e$. What it really means is that when $f_q > f_e$, the global-pull-local-push model is no longer an optimal one. Instead, a local-pull-global-push structure becomes more efficient, as the one shown in Figure 3.

Figure 3. Reverse comb.

The extreme of the reverse comb is the pure push-based strategy where each data node broadcasts its event to all other nodes. The analysis of the reverse comb is similar. The per

query communication cost is about

$$\frac{f_e}{f_q} \left(\frac{(n-1)^2}{s'} + n - 1 \right) + s' - 1 + \frac{f_e \tau}{f_q} \frac{s' - 1}{2},$$

where s' is the comb-width of the data duplication structure. In the equation, the first term, $\frac{f_e}{f_q} \left(\frac{(n-1)^2}{s'} + n - 1 \right)$, is the data duplication cost, where $\left(\frac{(n-1)^2}{s'} + n - 1 \right)$ is the data comb construction cost. The second term $(s' - 1)$ is the query cost for a query of length s' . The last term, $\frac{f_e \tau}{f_q} \frac{s' - 1}{2}$, is the data reply cost, where $f_e \tau / f_q$ is the average number of events per query. The distance between the query node and the location of a spike of an event is uniformly distributed from 0 to $s' - 1$. Therefore, the average distance is $(s' - 1)/2$. Take a derivative over s' , we have

$$s'_{optimal} \sim (n - 1) \frac{\sqrt{\frac{f_e}{f_q}}}{\left(1 + \frac{f_e \tau}{2f_q} \right)},$$

which is similar to Eq. (7).

In summary, combined with the reverse comb, the comb strategy covers the whole spectrum of the push and pull strategies, as shown in Fig. 4.

Figure 4. A Spectrum of Push-Pull Strategies.

This highlights the principle of the comb-needle structure: the communication strategy should adapt to the frequencies of queries and events. In particular, the lower the relative frequency of the query/event, the larger the number of nodes it propagates to. When the query frequency is low, global pull (query) plus local push (data) is more efficient. As the query frequency increases, combs become coarser. After a certain point ($f_e = f_q$), global push (data) combined local pull (query) becomes more efficient.

In Figure 5, we compare the performance of the comb (combined with reverse comb) strategy with pure pull-based and push-based strategies. In the figure, we have $f_q = 1$, $\alpha = 0.7$, $n = 10$, and $\tau = 0.5$. The x-axis is f_d where $f_d = f_e/n^2$, and the y-axis is the

communication cost. As expected, when f_d is small, pure push-based query performs better than that of the pull-based. The communication cost for push-based query increases linearly with the increase of f_d . For a large value of f_d , pull-based query outperforms. The comb query combines the benefit of push-based and pull-based strategy. When $f_d = 0.01$, i.e., $f_e = f_q$, the lines of push and pull intersect. At the same time, reverse comb is traversed to comb.

Figure 5. Compare Comb with Push and Pull.

3.5 Adaptive Comb-needle Strategy

The basic comb-needle strategy assumes the knowledge of the query and event frequencies and analyzes the performance in a snap shot. In practice, the query and event frequencies may be time-varying and unknown *a priori*. Thus, a good query strategy should adapt to such changes. In [21], we developed an adaptive comb-needle scheme. We summarize the basic idea of the adaptive scheme in the following and refer readers to [21] for details.

Recall that the optimal comb width (s) and the needle length (l) can be calculated using Eq. (7). Thus, a node only needs to estimate f_d and f_q to calculate s and/or l . Query and data nodes proceed as follows. If initiation information is available, s and l can be calculated using Eq. (7). If not, we set $s = l = 1$ to be conservative. After the initial stage, nodes will continuously estimate f_d and f_q . The larger the number of queries a node performs, the better the estimates of f_d and f_q are. The value of f_q is estimated based on counting the number of queries observed in a time window. A query node can estimate the value of f_d based on the number of replies it obtains. Based on the estimate, the query node calculates s .

Data nodes take a different approach for the following reasons. The estimates of f_d at a data node may be poor and so does the value of l calculated based on the estimates, especially when f_d is small. Note that a query node will have n^2 samples of f_d based on feedbacks while a data node can only observe itself. On the other hand, the important factor to guarantee high query success probability is that the needle length of data nodes should synchronize with the comb width. Based on these observations, we conclude that the values of s obtained from previous queries are important and should be used to estimate l . Furthermore, we should facilitate nodes to learn about the current value of s . A node learns about the value of s in two cases. First, a node may obtain information when it is being successfully queried. Second, we enhance the possibility that a node learns about the query by implementing a rotation of the query horizontal duplications as follows. Suppose that the current horizontal query is broadcast to lines $0, s, 2s, \dots, \lfloor (n-1)/s \rfloor * s$, then the next query will be on lines $1, 1+s, 1+2s, \dots, 1 + \lfloor (n-1)/s \rfloor * s$, and so on. A query node without the knowledge of current offset value can also randomly choose an offset between $(0, s)$. When a node is on a spike of the comb, it obtains the value of s . A data node uses its most recent information on s to synchronize the needle length l with the comb width s . The rotation scheme does not trigger additional communication cost, although it requires minor additional memory space. Simulation results show that our adaptive scheme can track the changes well and keep a high success rate while maintaining a low communication cost [21].

3.6 Fixed-node Query

There are different application scenarios that require different communication models. We have considered the applications where a query can be initiated from any nodes in the

network. On the other hand, many applications may have a fixed sink node and information query is only generated by the sink node. The location of the sink node is fixed and known by all sensors. In such a case, the pure push-based strategy can be degenerated as follows: a sensor node sends the data to the sink node through a shortest path spanning tree rooted at the sink instead of broadcasting the information to the whole network. We assume that the tree is established during initialization. We call this fixed-node query where the communication cost of the pure push-based strategy is significantly reduced. In such a scenario, the proposed comb-needle structure can still be applied to reduce communication cost under certain conditions, discussed as follows.

We first analyze the communication cost of the fixed-node query. We assume that all nodes in the network are aware of the location of the fusion center. Such information can be obtained through broadcast in the bootstrap phase. Note that in the reverse-comb case, i.e., when $f_q \geq f_e$, instead of building a comb-structure to spread the data to the whole network, the reverse comb can send the data directly to the fusion center, which is the same as the pure-push based scheme. Thus, we focus on the case where $f_q < f_e$. The communication cost of pure push-based strategy is independent of the frequency of the query process. The communication cost per unit time is

$$c_d \approx f_e n \alpha. \quad (9)$$

Again, the value of α reflects the location of the fusion node, αn is the average cost to report the event to the fusion center, and f_e is the event frequency. Recall that the per query cost $C_{optimal}$ is given in Eq. 8. The comb-needle is more efficient when $f_q C_{optimal} < c_d$. Thus, we have the following conclusion. If

$$\frac{f_q}{f_e} + \sqrt{\frac{f_q}{f_e}} \leq \frac{\alpha(1 - f_q \tau)}{2}, \quad (10)$$

then the comb-needle structure is more efficient than the pure push-based fixed-node query. The intuition here is clear: if $f_q\tau$ is small enough, the comb structure is desirable because it reduces unnecessary reporting. Otherwise, we should use pure push-based scheme in the case of fixed-node query. Similar analysis can be extended to the cases where there are a few fixed query entry points and their locations are known by all sensors in the network.

In Figure 6, we compare the performance of the comb with that of the fixed-node query. We focus on the case where $f_q \leq f_e$ because the (reversed) comb and the fixed-node query are the same when $f_q > f_e$. We have $f_q = 1$, $\alpha = 0.5$, $n = 10$, and $\tau = 0.1, 0.3$. In the figure, the straight dash-line is the performance of the fixed-node query. It is not surprising that the communication cost is linearly proportional to the event frequency. The other two lines are the performance of comb with $\tau = 0.1$ and $\tau = 0.3$, respectively, where the lower one corresponds to $\tau = 0.1$. It shows that when the event frequency is relatively high, comb is more energy efficient, especially when τ is relatively small.

Fixed-node query approach can also be extended to the case where the entry points of queries can be any sensor. We can assume that there is an elected head (e.g., virtual fusion center) in the field. All information is reported to the head and all queries are directly reported to the head. This approach is usually more efficient than broadcasting information to the whole network. The disadvantage is that the “head” is a single point of failure. Furthermore, nodes close to the head consume more energy and may die quickly. The second problem can be partially solved by rotations where each sensor node takes turns to be the head. However, this remedy requires global synchronization and update which indeed incur a large amount of signaling cost and complexity.

Figure 6. Compare comb with fixed-node query.

Figure 7. Compare the width of regular comb and sequential comb.

3.7 Binary Query

We have focused on queries where the objective is to find all the relevant data. There exist queries that only demand a binary “Yes-or-No” type of answers. An example of such a query can be whether or not there are tanks in the field. To obtain the information, we can perform a sequential comb query when the delay is not important. We query the first horizontal line. If the answer is Yes, then the query stops. If the answer is No, we continue to query the second horizontal line, and so on.

The communication cost for the sequential comb query is calculated as follows. Let comb width be s . Let p_s be the probability that there is an event in an area of size $n \times s$. For simplicity, we set $\tau = 1$. We have $p_s = 1 - (1 - p_0)^{ns}$, where $p_0 = \frac{f_e}{f_q n^2}$, which is the probability that an event happens at a node during the query interval.

Thus, the average query cost per query is

$$c_q \approx \sum_{i=1}^{n/s-1} i n p_s (1 - p_s)^{i-1} + (1 - p_s)^{n/s} \frac{n^2}{s} = \frac{1 - (1 - p_0)^{n^2}}{1 - (1 - p_0)^{ns}} n. \quad (11)$$

In the equation, $p_s(1 - p_s)^{i-1}$ is the probability that an event is not discovered by the $(i - 1)$ th spike, but by the i th spike on the comb, and $(1 - p_s)^{n/s}$ is the probability that no event is discovered. In the equation, the cost of the vertical line is ignored.

The data duplication cost per query is $c_d = \frac{f_e}{f_q} s$. We have

$$C_{total} = c_q + c_d + c_r,$$

where c_r is the event report cost which is independent of s , as discussed in Section 3.3. Take a derivative with respect to s and set it to zero, we have

$$\frac{a^{ns}}{(1 - a^{ns})^2} = \frac{f_d}{f_q \ln \frac{1}{a} (1 - a^{n^2})},$$

where $a = 1 - \frac{f_d}{f_q}$. The optimal value of s is the solution to the above equation. It is interesting that the optimal value of s in the sequential comb is actually smaller than that of the regular comb as shown in Figure 7. The intuition is that in a sequential comb, the query might stop earlier than traversing the whole network. In other words, sequential comb rewards early information discovery.

Depending on applications, delay can be a concern in the sequential comb. When delay is important to the query process, we might need to be more aggressive. For example, one or more sequential query lines can progress at the same time. In fact, the regular comb structure also encounters longer delay than push-based strategy due to the delay caused by querying the network. In summary, push-based strategy results in the lowest query delay, which is followed by comb and then pull-based strategy. Sequential comb results in the longest delay.

4 Simulation

In this section, we verify the theoretical result given in the previous section via simulations. While in the previous section we make simplified assumption about the network model, in the simulation we adopt a more realistic one, including unreliable links and irregular grids.

4.1 The Simulation Environment

Prowler [22] is used for the simulation of comb-needle query model. Prowler is a probabilistic wireless network simulator based on the event-driven structure. Prowler provides various radio models. In our experiments, we use the default model. The transmission model is

given by:

$$P_{rec,ideal}(d) \leftarrow P_{transmit} \frac{1}{1 + d^\gamma}, \text{ where } 2 \leq \gamma \leq 4 \quad (12)$$

$$P_{rec}(i, j) \leftarrow P_{rec,ideal}(d_{i,j})(1 + \alpha(i, j))(1 + \beta(t)) \quad (13)$$

where $P_{transmit}$ is the signal strength at the transmitter and $P_{rec,ideal}(d)$ is the *ideal* received signal strength at distance d , α and β are random variables with normal distributions $N(0, \sigma_\alpha)$ and $N(0, \sigma_\beta)$, respectively. A network is asymmetric if $\sigma_\alpha > 0$ or $\sigma_\beta > 0$. A node j can receive a packet from node i if $P_{rec}(i, j) > \Delta$ where Δ is the threshold. Furthermore, an additional parameter p_{error} models the probability of a transmission error by any unmodeled effects. We have used $P_{transmit} = 1$, $\gamma = 2$, $\sigma_\alpha = 0.45$, $\sigma_\beta = 0.02$, $p_{error} = 0.05$ and $\Delta = 0.1$ in simulations in this section unless otherwise specified. Prowler also provides the MAC layer that simulates the Berkeley notes' CSMA protocol, including random waiting and backoffs. We assume each packet is 960-bit long and the transmission rate is 40 kbps.

While our analysis in the previous section is based on a regular grid network, in simulation we use a random grid topology model which is more realistic in some deployment scenarios. An instance of the random grid network is generated as follows, first the nodes are put on a regular grid, then the x and y coordinates of each node is shifted independently via a random offset generated from a uniform distribution. Given the size of random offsets, one can generate networks with regular or random node distributions. Figures 8 and 9 show 10x10 networks with small and large random offsets, respectively.

Figure 8. Small random offset 0.1.

Figure 9. Large random offset 0.5.

4.2 Routing Protocol

Unlike in a regular grid network, building the comb-needle querying in a random grid network needs extra attention, since the “left-right-up-down” in the random network is no longer well-defined, and the combs and the needles can no longer be straight lines. To overcome this problem, we build approximations of the combs and needles using a Constrained Geographical Flooding (CGF) method, similar to the illustration in Figure 1(b). The needles and the spikes on the comb have a width of w , besides their lengths.

Schemes to build trajectories based on location-information have been discussed in the literature, e.g., [16, 19, 20]. Such schemes can be adopted to build the comb and needles. The CGF discussed here is an alternative which is simple to implement at the cost of reliability and communication overhead.

In CGF, whenever a new packet arrives, each node will decide if it should rebroadcast the packet according to the geographical constraints for the type of packet. More specifically, for a query packet, a node will broadcast only if it is at the comb branch; for an event packet, a node will broadcast only if it is vertically within the duplication distance to the source; and for a report packet, a node will broadcast only if it is at the comb branch and closer to the query node. If the radio range is larger than the grid distance, duplicated packets are received at each node while the total number of packets are still bounded by the size of comb or needle. This increases the coverage and thus the robustness. The pseudo code of the protocol, including subroutines, is shown as follows.

```
received query  $q$  at  $v$  do  
  if  $new(q)$  and  $atComb(q.x, q.y, v.x, v.y, s, w)$  then  
    broadcast  $q$ ;  
     $events = getEvents(q)$ ;  
    if  $events \neq \emptyset$  then
```

```

    broadcast  $r(events, q.x, q.y, v.x, v.y)$ ;
  end
end
end

received event  $e$  at  $v$  do
  if  $new(e)$  and  $atNeedle(e.x, e.y, v.x, v.y, l, w)$  then
    copy  $e$  to memory
    broadcast  $e$ ;
  end
end

received report  $r(events, q_x, q_y, s_x, s_y)$  at  $v$  do
  if  $new(r)$  and  $q_x = v.x$  and  $q_y = v.y$  do
    return events;
  end
  if  $new(r)$  and  $atPath(q_x, q_y, s_x, s_y, v.x, v.y, w)$  then
    broadcast  $r$ ;
  end
end

function  $out = atComb(q_x, q_y, v_x, v_y, s, w)$ 
 $d = \text{mod}(|q_y - v_y|, s)$ ;
 $out = |q_x - v_x| < w$  or  $\min(d, s - d) < W$ ;

function  $out = atNeedle(e_x, e_y, v_x, v_y, l, w)$ 
 $out = |e_x - v_x| < w$  and  $|e_y - v_y| < l + w$ ;

function  $out = atPath(q_x, q_y, s_x, s_y, v_x, v_y, w)$ 
 $out = |s_y - v_y| < w$  and  $(v_x - s_x)(q_x - v_x) > 0$  or
 $|q_x - v_x| < w$  and  $(v_y - s_y)(q_y - v_y) > 0$ 

```

4.3 Performance Evaluations

Two separate tests are done in this environment. The first test is to discover what is the best spacing for the comb for a grid network with small random offset (Figure 8), in order to verify the theoretical result in the paper. The second test is to show the robustness of the protocol with a varying comb width w for a grid network with large random offset (Figure 9). In both tests, query rate is set to be one query every 10 seconds ($f_q = 0.1$). We note that broadcast is assumed in this section.

For the first test, we set network size to be 7x7, 10x10, 12x12, and 15x15, respectively. Let the comb width s be 1, 3, 5, and 7, and let $l = s$. We first let event rate be 1 packet per second, i.e., $f_q/f_e = 0.1$, and then reduce the event rate to 0.1, i.e., $f_q/f_e = 1$. Figure 10 and Figure 11 show the number of packets transmitted in the network per query. In the first case, $s = 3$ results in the minimum number of packets. In the second case, $s = 7$ is the best. These results match well with the result in Eq. (7). The error bar in the figures indicates the standard deviation, corresponding to a confidence interval of 68%.

Figure 10. $f_q = 0.1, f_e = 1$.

Figure 11. $f_q = 0.1, f_e = 0.1$.

Note that $s = 1$ corresponds to pure pull. Figure 12 shows the performance of $s = 1$ vs. the optimal s when $f_q = 0.1, f_e = 1$. The figure clearly shows that the cost of pure pull increases with the network size linearly and the cost of an optimal comb-needle query increases sub-linearly.

Figure 12. A full pull vs. optimal comb-needle.

The second test was run on the 10x10 network with larger random offsets (Figure 9). In this case, we let $f_q = 0.1, f_e = 1, l = s = 3$, and the comb width w be 0.5, 0.6, 0.7, 0.8, and

0.9. Figure 13 shows the delivery ratio (the total number of events received over those sent) with respect to different w . We can see that the wider the w is, the higher the delivery ratio. Figure 14 shows the corresponding cost. Clearly there is a trade-off between the delivery ratio and the communication cost. Note that when w goes to 1.5, the combing is similar to flooding.

5 Discussion

While in the paper we only discussed supporting discovery and query in a whole network, the application of comb-needle model is not limited to that. The idea is applicable for supporting geographic queries like “how many tanks are there in region x?”. For supporting such geographic queries, one traditional strategy is to geocast the query to the region, i.e., unicast the query to the region, and then broadcast the query in the region. Similarly, we can choose to unicast the query to the region, then comb the region with the desired granularity.

We have presented and analyzed the comb-needle structure in a regular grid network. The main ideas are 1) the trajectory of query and event duplications should cross each other to guarantee event discovery; and 2) the structure should adapt based on query and event frequencies. In a network with a random topology, we can build approximations of the combs and needles using a Constrained Geographical Flooding (CGF) method, similar to the illustration in Figure 1(b). The needles and the spikes on the comb has a width, as discussed in detail in Section 4. Alternatively, we can use trajectory-based routing schemes, such as in [16, 19, 18, 20], to develop trajectories for query and data duplication.

Query reliability is an important issue and is inherent in all wireless sensor systems. There are many proposed schemes to improve communication reliability by exploring local

reinforcement (such as retransmission and mesh-based transmission) and spatial diversity (such as disjoint path). They can be applied to the comb-needle structure. We can also explore the broadcasting nature of wireless media to enhance the reliability. For instance, in the simulation, the radio range can normally cover two to three nodes in any direction. We can also vary width of spikes to make it more robust to failures. In addition, we also note that the data push component not only helps to reduce the query cost in the comb-needle scheme, but also introduces data redundancy which is desirable in an unreliable network. We have considered the issue of query coverage in unreliable networks and investigated how redundancy can improve the coverage via both theoretical analysis and simulation in [21].

Data aggregation can be easily applied to the comb-needle structure due to the synchronization effect of the query process. In other words, when a query is sent to the network, nodes on the query path are awake and expecting (multiple) replies in a short period of time. Such replies can be aggregated and compressed to further reduce communication cost.

In this paper, we simply count the number of packet transmissions as an indication of the communication cost. We assume certain sleep-awake patterns are being applied in the underlining layer. Most nodes are asleep most of the time. The query process will wake up the nodes in the query path and wait for replies, in a similar process as the one described in [23]. In addition, data push will also wake up certain nodes, but these nodes can go back to the sleep state as soon as the data duplication process is completed locally. Because of this structured process, the comb-needle structure can fully exploit the benefit of low-duty cycle provided by sleep-awake patterns.

When the entry points of the query vary, vertical and horizontal lines in the comb-structure vary, which balances the power consumption of different nodes. In the case of a fixed entry point, when the comb structure is desired (based on Eq.(10)), vertical and

horizontal rotations can be used to balance power consumption.

We should note that comb-needle is not the only possible structure and we cannot prove that comb-needle is an optimal one. It is of great theoretical interest to discover the optimal structure for information dissemination and discovery, in particular in random networks. Our conjecture is that the shape of the optimal structure may be determined by the particular network topology.

6 Conclusion

In this paper we proposed the comb-needle, a simple yet efficient data discovery scheme for supporting queries in large-scale sensor networks. We also used the comb-needle model as a substrate to study the benefit of balancing push and pull in data gathering and dissemination in large-scale wireless networks. The comb-needle scheme (including the reverse one) covers a spectrum of push and pull strategies, with the pure push-based and pure pull-based schemes in two extremes. The intuition is that the communication strategy should adapt to the relative frequency between queries and events. In particular, the lower the (relative) frequency of the query/event, the larger the number of nodes it propagates to. When the query frequency is low, global pull (query) plus local push (data) is more efficient. As the query frequency increases, combs become coarser. We have demonstrated that in general, the comb-needle scheme performs better than both pure push-based and pure pull-based schemes. We note that the proposed scheme can be used in the hierarchical structure as well. Furthermore, data aggregation and compression can be integrated into the comb-needle strategy to reduce the communication cost. In addition, optimal structures for information dissemination and discovery, in particular in random networks, are unknown. These are interesting directions

to explore in the future.

References

- [1] W. Peng and X. Lu, “On the reduction of broadcast redundancy in mobile ad hoc networks,” in *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2000.
- [2] A. Qayyum, L. Viennot, and A. Laouiti, “Multipoint relaying: An efficient technique for flooding in mobile wireless networks,” INRIA, Tech. Rep. Research Report RR-3898, Feb. 2000. [Online]. Available: citeseer.nj.nec.com/qayyum00multipoint.html
- [3] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 1999, pp. 152–162.
- [4] D. Braginsky and D. Estrin, “Rumor routing algorithm for sensor networks,” in *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, October 2002.
- [5] S. Shakkottai, “Asymptotics of query strategies over a sensor network,” in *Proceedings of the IEEE INFOCOM*, Hongkong, China, March 2004.
- [6] J. Heidemann, F. Silva, and D. Estrin, “Matching data dissemination algorithms to application requirements,” in *Proceedings of the first international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 218–229.
- [7] B. Krishnamachari and J. Heidemann, “Application-specific modelling of information routing in sensor networks,” in *Proceedings of the IEEE International on Performance, Computing, and Communications Conference*. Phoenix, Arizona, USA: IEEE, April 2004, pp. 717–722.

- [8] N. Sadagopan, B. Krishnamachari, and A. Helmy, "Active query forwarding in sensor networks," *Elsevier Journal of Ad Hoc Networks*, 2003.
- [9] S. Kapadia and B. Krishnamachari, "Comparative analysis of push-pull query strategies for wireless sensor networks," in *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2006.
- [10] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *International Journal of High Performance Computing Applications*, vol. 16, no. 3, 2002.
- [11] J. Ahn and B. Krishnamachari, "Fundamental scaling laws for energy-efficient storage and querying in wireless sensor networks," in *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM Press, 2006, pp. 334–343.
- [12] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data*. ACM Press, 2003, pp. 491–502.
- [13] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proceedings of the first international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 63–75.
- [14] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mob. Netw. Appl.*, vol. 8, no. 4, pp. 427–442, 2003.

- [15] D. Ganesan, D. Estrin, and J. Heidemann, "DIMENSIONS: Why do we need a new data handling architecture for sensor networks?" in *Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I)*, Princeton, NJ, October 2002.
- [16] R. Sivakumar, B. Das, and V. Bharghavan, "Spine routing in ad hoc networks," *CM/Baltzer Publications Cluster Computing Journal, Special Issue on Mobile Computing*, 1998.
- [17] I. Stojmenovic, P. Eduardo, and V. Pena, "A scalable quorum based location update scheme for routing in ad hoc wireless networks," SITE, University of Ottawa, Tech. Rep., 1999.
- [18] I. Aydin and C.-C. Shen, "Facilitating Match-Making Service in Ad hoc and Sensor Networks Using Pseudo Quorum," in *11th International Conference on Computer Communications and Networks (ICCCN 2002)*, Miami, Florida, October 14–16 2002.
- [19] B. Nath and D. Niculescu, "Routing on a curve," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 155–160, 2003.
- [20] J. Tchakarov and N. Vaidya, "Efficient content location in wireless ad hoc networks," in *IEEE International Conference on Mobile Data Management (MDM)*. IEEE, January 2004.
- [21] X. Liu, Q. Huang, and Y. Zhang, "Combs, needles, haystacks: Balancing push and pull for discovery in large-scale sensor networks," in *ACM Sensys 2004*. Baltimore, MD: ACM, November 3-5 2004.
- [22] G. Simon, "Probabilistic wireless network simulator," 2003, <http://www.isis.vanderbilt.edu/projects/nest/prowler/>.
- [23] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *ACM International Conference on Mobile Computing and Networking (MOBICOM)*. ACM, 2004.

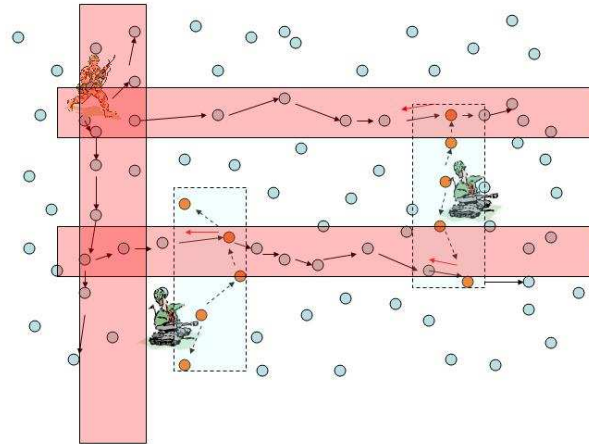


Figure 1: A Comb-Needle Example.

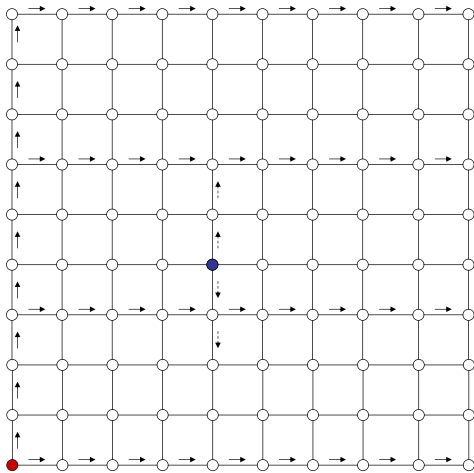


Figure 2: A comb-needle structure where $l = 5$ and $s = 3$.

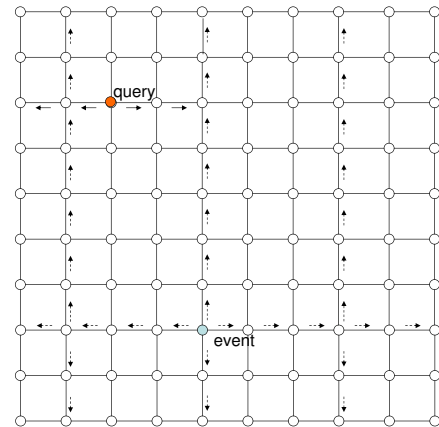


Figure 3: Reverse comb



Figure 4: A Spectrum of Push-Pull Strategies

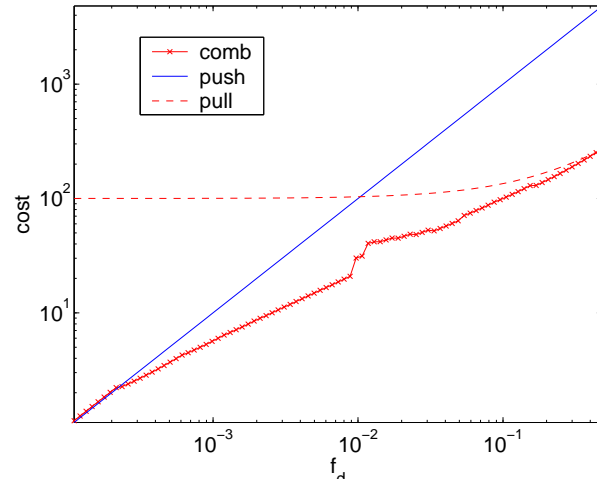


Figure 5: Compare Comb with Push and Pull



Figure 6: Compare comb with fixed-node query.

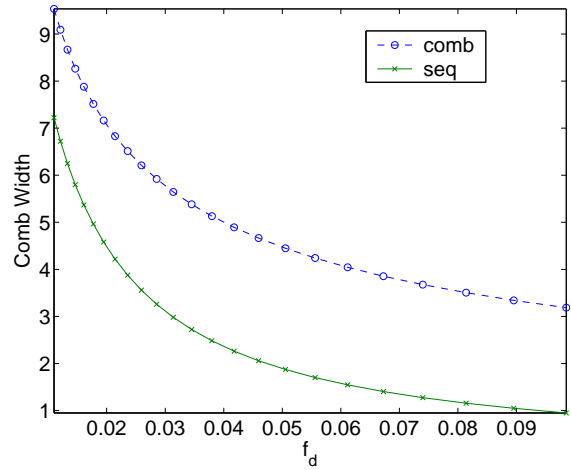


Figure 7: Compare the width of regular comb and sequential comb

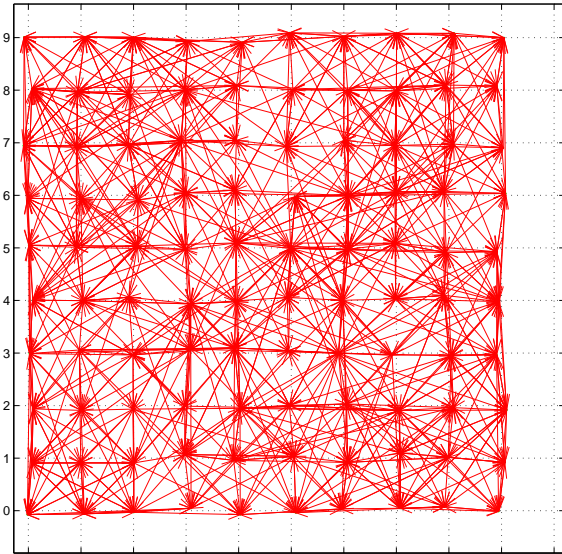


Figure 8: Small random offset 0.1

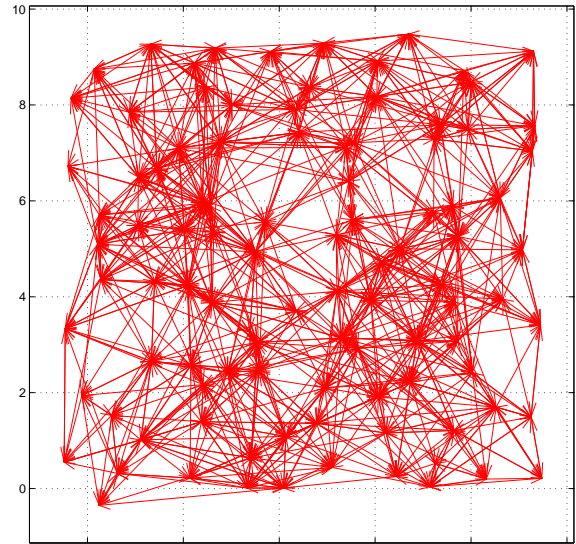


Figure 9: Large random offset 0.5

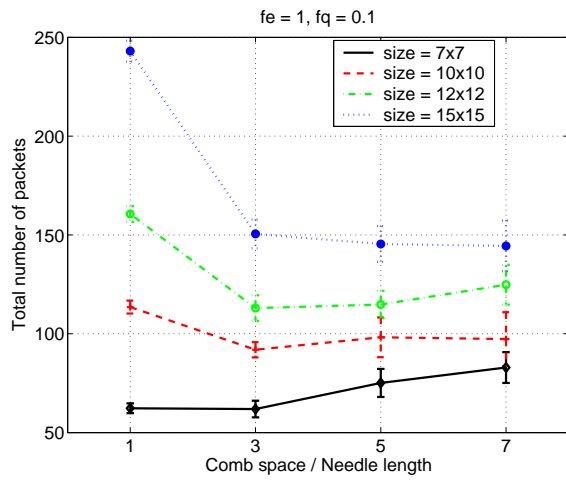


Figure 10: $f_q = 0.1, f_e = 1$

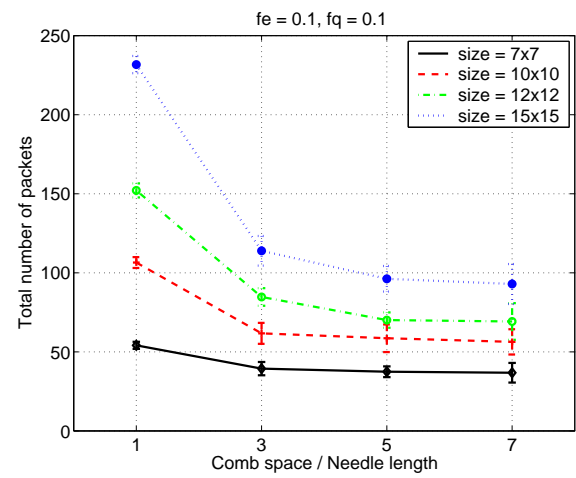


Figure 11: $f_q = 0.1, f_e = 0.1$

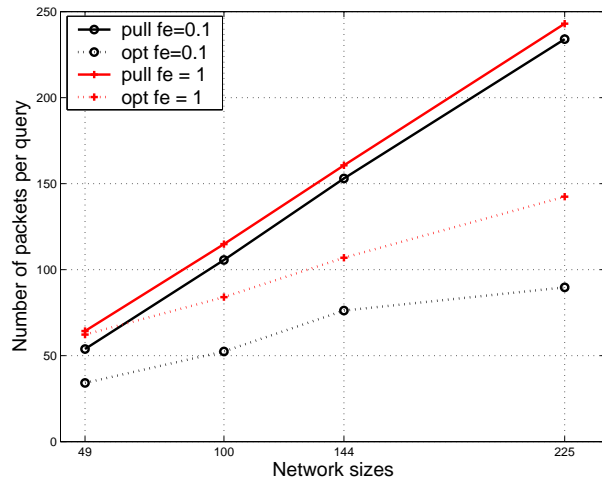


Figure 12: A full pull vs. optimal comb-needle

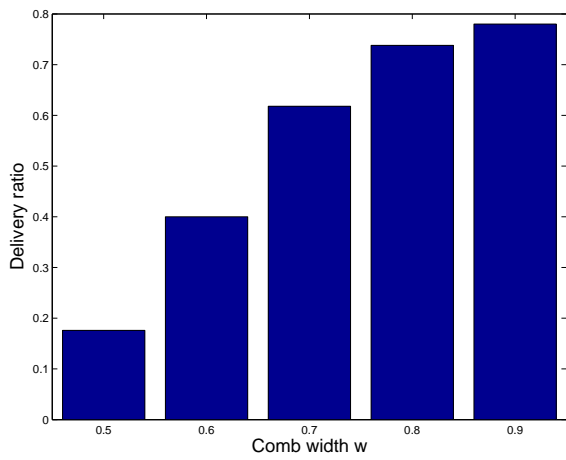


Figure 13: Delivery ratio for various widths

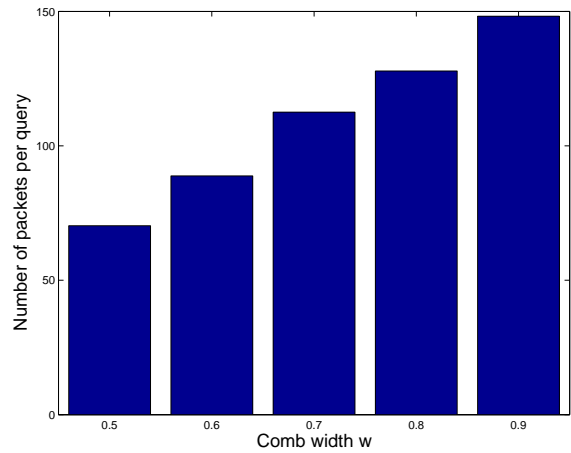


Figure 14: Cost for various widths