Inquiry and Introspection for Non-Deterministic Queries in Mobile Networks

Vasanth Rajamani
Christine Julien
Jamie Payton, CS Dept., UNC-Charlotte
Gruia-Catalin Roman, CSE Dept., Wash.U.-St. Louis

TR-UTEDGE-2008-014

Inquiry and Introspection for Non-Deterministic Queries in Mobile Networks

Vasanth Rajamani[1], Christine Julien[1], Jamie Payton[2], and Gruia-Catalin Roman[3]

[1] Department of Electrical and Computer Engineering
The University of Texas at Austin
{c.julien, vasanthrajamani}@mail.utexas.edu
[2] Department of Computer Science
University of North Carolina, Charlotte
payton@uncc.edu
[3] Department of Computer Science and Engineering
Washington University in Saint Louis
roman@wustl.edu

Abstract. This paper focuses on the information gathering support needs for enterprises that operate over wireless mobile ad hoc networks. While queries are a convenient way to obtain information, the highly dynamic nature of such networks makes it difficult to ensure a precise match between the results returned by a query and the actual state of the enterprise. However, decisions can be made based on the perceived quality of the information retrieved; specialized query support is needed to control and assess the accuracy of the query results. In this paper, we introduce the notion of inquiry mode to allow the user to exercise control over the query processing policy so as to match the level of accuracy to the requirements of the task. In addition, we describe the use of query introspection, a process for assessing the fitness of a particular inquiry mode. Both concepts are formalized, illustrated, and evaluated.

1 Introduction

Information drives the modern world. Everyday decisions depend on the quality of data decision makers have. With the introduction of mobile wireless devices, access to information has been extended to any individual carrying a phone or PDA. This, in turn, led to changes in the very nature of the enterprise structure by empowering mobile users and by facilitating more decentralized decision processes, faster reaction times, and more nimble data acquisition. A still more radical transformation is made possible by the emergence of mobile ad hoc networks (MANETs) that support application domains where a still more fluid organization is required to adapt to rapidly evolving circumstances. Emergency response units, wilderness exploration groups, and battlefield intelligence teams demand quality information gathered from distributed, cooperating sources.

The transformation will have far reaching implications, as enterprises that rely on connectivity to the wired infrastructure are likely to evolve to include

operations over local MANETs with only occasional access to the wired network. Construction sites are a representative example. Today, phones are used on the site to gather information about the status of jobs and to track developing situations. A chemical spill is likely to trigger a large volume of phone conversations to coordinate a response to the emergency and to assess the appropriateness of those actions. Enterprise level systems on the wired network, though well-suited to support logistics and planning, are not nimble enough to support these ad hoc peer-to-peer interactions that emerge unexpectedly. A more effective solution is required to acquire up-to-date data on-demand.

These new enterprises demand flexible and timely access to spatially correlated information about highly dynamic environments. Because the information is distributed, accessing it entails the evaluation of a distributed query over a rapidly evolving network, rendering any atomicity guarantees infeasible much of the time. The spatiotemporal dimension of the information encourages decision makers to pose questions in a manner sensitive to one's mental model of how space is organized and how the system evolves. Furthermore, one is likely to question the soundness of the decision process. Am I asking the right question? Am I looking in the right place? Am I getting an accurate enough answer? By knowing how a query is processed and how volatile the system state is, one can gain important semantic information about the data a query returns. This, combined with an ability to specify how queries are evaluated, can significantly impact the quality of the decision process. For example, a construction site supervisor can specify that a query will be evaluated by sampling across the entire site or by acquiring values from all devices within a smaller local area. Even though the queries may return identical results, a supervisor will interpret the data differently. Finally, changes in query results over time (e.g., rising chemical concentration readings) or the presence of unexpected values may offer insight into the adequacy of the query relative to the decision maker's goals.

This paper explores the semantic dimension of query processing over MANETs. Starting with the premise that the universe of discourse is the global state of a connected mobile ad hoc network that represents a distributed mobile enterprise, we seek to provide decision makers with a new set of query processing tools. These are meant to enable both flexible control over the spatiotemporal dimension of query processing and the ability to assess the fitness of the query processing for the specific task at hand. Our contribution is twofold:

- We propose an inquiry mode as the means to specify how a query is evaluated across a MANET; we identify several inquiry modes that relate to different semantic interpretations of results; and we offer both a general formalization of the concept and specific query processing protocols.
- Complementing the notion of inquiry mode is query introspection, which enables evaluation of the adequacy of an inquiry mode. We propose the use of adequacy metrics that compare query results against expectations or results obtained from previous correlated queries.

Our approach represents an important shift in the way one thinks about query processing. A query is intellectually decoupled from the traditional notions of

database processing and is promoted as a basic tool for exploring the surrounding world. Within this broad context, new user requirements for query processing emerge, which suggest that queries should be sensitive to the spatiotemporal nature of the environment and its evolutionary dynamics. In the remainder of this paper, we first define, formalize and demonstrate inquiry modes. Section 3 then defines and demonstrates query introspection. Section 4 presents our programming model and a case study application that employs it. We discuss related work in Section 5 and conclude in Section 6.

2 Defining Inquiry in Dynamic Networks

Our approach is based loosely on our previous work modeling change in mobile environments [1]. In our model, queries can retrieve information from the network using a variety of techniques, or *inquiry modes*, each of which entails its own costs and benefits. Informally, an inquiry mode specifies the subset of hosts that contribute to resolving the query. Different inquiry modes may generate vastly different results for the same query. In this section, we formalize the specification of queries, their inquiry modes, and their results and provide concrete examples of how real protocols can be expressed using this formalization.

2.1 Hosts, Configurations, and Configuration Change

A mobile ad hoc network is a closed system of hosts, each represented as a triple (ι, ζ, ν), where ι is the host's unique identifier, ζ is its context, and ν is its data value. In a simple model, the context can be simply a host's location. In more complicated models, the context may include a list of a host's neighbors, routing tables, and other system or network information. A snapshot of the global abstract state of a mobile ad hoc network, which we call a *configuration*, C, is simply the set of these host tuples, one for every host in the network.

To capture network connectivity, we define a binary logical connectivity relation, $\mathcal{K}$, to express the ability of one host to communicate with a neighboring host. Using the values of the fields of a host triple, we can derive physical and logical connectivity relations. As one example, if the host's context element, ζ, includes the host's location, we can define a physical connectivity relation based on communication range. $\mathcal{K}$ is not necessarily a symmetric relation; in the cases that it is symmetric, $\mathcal{K}$ specifies bi-directional communication links.

The environment evolves as the network topology changes, value assignments occur, and hosts exchange messages. We model network evolution as a state transition system where the state space is the set of possible system configurations, and transitions are *configuration changes*. Specifically, a single configuration change consists of one of the following:

- *neighbor change*: changes in hosts' states impact the connectivity relation.
- *value change*: a single host can change its stored data value.
- *message exchange*: a host can send a message that is received by one or more neighboring nodes.

To refer to the connectivity relation for a particular configuration in this evolution, we assign configurations subscripts (e.g., C_0, C_1, etc.) and use $\mathcal{K}_i$ to refer to the connectivity relation for configuration i.

We build on $\mathcal{K}$ to define *reachability* across configurations. The reachability relation, $\mathcal{R}_{(i,j)}$, is a binary relation on host tuples that indicates the potential of one-way multi-hop communication between them that starts no earlier than the i^{th} configuration and completes no later than the j^{th} configuration:

$$\langle \forall k : i \leq k \leq j :: h \in C_k \Rightarrow (h,h) \in \mathcal{R}_{(i,j)} \rangle$$
$$\langle \forall h_1, h_2, k : i \leq k \leq j :: (h_1,h_2) \in \mathcal{K}_k \Rightarrow (h_1,h_2) \in \mathcal{R}_{(i,j)} \rangle$$
$$\langle \forall h_1, h_2, h_3, k : i \leq k < j :: ((h_1,h_2) \in \mathcal{R}_{(i,k)} \wedge (h_2,h_3) \in \mathcal{K}_{k+1}) \Rightarrow (h_1,h_3) \in \mathcal{R}_{(i,j)} \rangle$$[4]

First, every host is always reachable from itself. Second, if one host (h_1) is connected to another (h_2) in any configuration between i and j inclusive, then h_2 is reachable from h_1. Finally, we recursively define reachability.

2.2 Queries, Inquiry Modes, and Query Results

We extend our definition of reachability to define *query reachability*, which, informally, determines whether it was possible to deliver a query to and receive a response from some host h within the sequence of configurations. Given the host who issues the query, $\overline{h}$, query reachability for query q, $\mathcal{R}^q$, is defined as:

$$(\overline{h}, h, i) \in \mathcal{R}^q \Leftrightarrow (\overline{h}, h) \in \mathcal{R}_{(0,i)} \wedge (h, \overline{h}) \in \mathcal{R}_{(i,m)}$$

It is not only necessary that h was reachable from the query issuer during the query, but also that, after h was able to receive the query, $\overline{h}$ was reachable from h, ensuring that it was possible for h's response to reach the query issuer.

In our model, the inquiry mode is the technique used to process the query over a set of configurations. An inquiry mode is defined by a *forward-function* and a *respond-function*, both of which use a host's state to make a decision about whether to propagate and/or respond to a query. From any host's perspective, an inquiry model is simply the application of two independent functions: $I \triangleq \langle f, r \rangle$. Each of f and r is a boolean function over a host tuple h. In the same atomic step in which a host receives a query, it evaluates both f and r given the particular query and the host's state. Since a message exchange constitutes a configuration change that takes the network from some configuration C_i to C_{i+1}, the two functions are evaluated on the receiving host's tuple in configuration C_{i+1}.

Next, we define a trio of relations over host tuples that allow us to formalize a query's result. Briefly, the *forwards* relation, Φ_q, specifies pairs of senders and receivers of the query and the configuration of reception; the *receptions* relation, Γ_q, specifies hosts that received the query and the relevant configuration; and the *responses* relation, Ψ_q, specifies whether a host qualified to generate a response.

[4] In the three-part notation: $\langle$ `op` *quantified_variables* : *range* :: *expression* $\rangle$, the variables from *quantified_variables* take on all possible values permitted by *range*. Each instantiation of the variables is substituted in *expression*, producing a multiset of values to which `op` is applied. If no instantiation of the variables satisfies *range*, the value of the three-part expression is the identity element for `op`, e.g., *true* if `op` is $\forall$.

The forwards relation is a ternary relation over host tuples and configuration numbers. Specifically, $(h_1, h_2, i) \in \Phi_q$ if the host identified by $h_1.\iota$ forwarded the query q in a configuration in which h_1 captured its state, and the host identified by $h_2.\iota$ received the query in a configuration in which h_2 captured its state. In addition, the host that forwarded the query (h_1) must also have satisfied the query's *forward-function*, $q.f$ in the state that it received the query. Formally:

$$(h_1, h_2, i) \in \Phi_q \Rightarrow$$
$$\langle \exists j, h_1' : i < j \wedge h_1'.\iota = h_1.\iota :: (h_1', i) \in \Gamma_q \wedge q.f(h_1') \wedge (h_1, h_2) \in \mathcal{K}_j \rangle$$

In this formalization, i and j identify configurations; i is the configuration in which the host h_1' received the query, and j is the configuration in which the forwarding occured. Forwarding the query caused the transition $C_j \rightarrow C_{j+1}$.

We next define the receptions relation, Γ_q. A host is in the receptions relation if it was the target of a forward or if it is the query issuer:

$$(h, i) \in \Gamma_q \Rightarrow (h = \overline{h} \wedge i = 0) \vee \langle \exists h' :: (h', h, i-1) \in \Phi_q \rangle$$

This requires that for a host other than the query issuer to receive a query in C_i, the query must have been forwarded by a neighboring host in C_{i-1}.

Finally, the responses relation, Ψ_q, defines the hosts that received the query and generated a response to it. To generate a response, the host must receive the query and satisfy the *respond-function*. Formally, Ψ_q is:

$$h \in \Psi_q \Rightarrow \langle \exists i :: (h, i) \in \Gamma_q \wedge q.r(h)) \rangle$$

A query's result, ρ, is a subset of a configuration: it is a collection of host tuples that constitute responses to the query; no host in the network is represented more than once in ρ, though it is possible that a host is not represented at all (e.g., because it was never reachable from the query issuer). The constraints defining a query's result stem from the concepts derived above. First, a result present in the query must come from a host that responded to the query:

$$h \in \rho \Rightarrow h \in \Psi_q \tag{1}$$

Second, any result must have satisfied the aforementioned property of query reachability. This ensures that both forward and reverse paths exist for query propagation and response collection. Formally, given the query issuer, $\overline{h}$:

$$h \in \rho \Rightarrow \langle \exists i, j : i \leq j :: (h, i) \in \Gamma_q \wedge (\overline{h}, h, j) \in \mathcal{R}^q \rangle \tag{2}$$

2.3 Examples of Application Protocols

The ability to specify the inquiry mode with which a query executes gives an application fine-grained control. Consider an application that has some requirement for query quality. Given the availability of forward and respond function definitions, the application can construct a variety of possible query processing protocols, honing in on the implementation best suited to the combination of

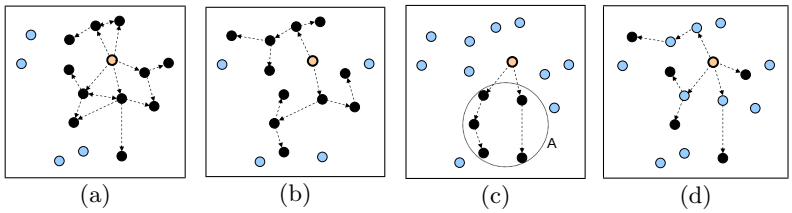

Fig. 1. Potential inquiry modes. Solid lines indicate available connections, dashed lines indicate sent messages. (a) Flooding. Every node within the constraint (2 hops) retransmits the query. (b) Probabilistic. Every node within the constraint (3 hops) that receives the packet retransmits it to 2 random neighbors. (c) Location Based. The query reaches the nodes in region A. (d) Random. The query reaches any 5 nodes.

the application's requirements and the query environment. In this section we show how query processing protocols that are commonly used in mobile ad hoc networks can be easily represented using inquiry modes and their components.

Flooding Inquiry Mode: Flooding based queries are most common in mobile applications [2, 3]. The sending node broadcasts the query to all of its one-hop neighbors, who in turn propagate the message to their neighbors. A query is very likely to reach every node in the network. However, this approach can be very expensive in terms of the message overhead [4]. Approaches also exist that constrain the flooding to some region of the network [5]. Because a basic flooding protocol is deterministic, it cannot be parameterized, so no protocol parameters are included in the inquiry mode. However, a constrained flood can be parameterized by specifying the number of hops across which the query is flooded. Fig. 1(b) shows the messages sent by and the nodes responding to a flooding inquiry constrained to nodes within two hops of the query issuer.

Expressing the flooding inquiry mode using our formulation is trivial:

$$\mathcal{I}_{flooding} = \langle true, true \rangle$$

This query reaches all hosts but at a significant communication overhead.

Probabilistic Inquiry Mode: Probabilistic techniques distribute the query with a lower message overhead by reducing the number of nodes involved in query propagation. Fig. 1(c) depicts an example, where each node receiving a query retransmits it to two randomly selected neighbors. A more common variant is to use a probability function to determine whether a particular node should rebroadcast a received message [4]. Additional parameters can be used to determine how many times to retransmit a message [6].

Here, the respond function is identical to flooding's respond function. The forward function ensures that only a fraction of messages propagate. Every host generates a random number ($rand$) and passes it as an argument to the $f_{probabilistic}$ function, which is used to determine satisfiability:

$$f_{probabilistic}(\theta) \triangleq (\theta < p)$$

$$\mathcal{I}_{probabilistic} = \langle f_{probabilistic}(rand), true \rangle$$

In this inquiry mode, the query reaches only a probabilistically selected set of hosts, which comes with added complexity but reduced overhead.

Location based Inquiry Mode: If location information is available, it can direct queries to particular regions. Fig. 1(d) demonstrates a location based query targeting region A. In this inquiry mode, it is important to be able to compute a logical function that determines whether a given node satisfies the query's location requirements. This is accomplished by writing forward and respond functions that use the cartesian distance to evaluate satisfiability:

$$f_{location}(x_1, y_1) \triangleq ((x_1 - x_2)^2 + (x_2 - y_2)^2 < maxD)$$

$$r_{location}(x_1, y_1) \triangleq ((x_1 - x_2)^2 + (x_2 - y_2)^2 < maxD)$$

$$\mathcal{I}_{location} = \langle f_{location}(h.\zeta.\lambda.x, h.\zeta.\lambda.y), r_{location}(h.\zeta.\lambda.x, h.\zeta.\lambda.y)\rangle$$

The parameters come from the host's location (λ) stored in its context ($h.\zeta$). The query targets hosts in a specific location, reducing the communication overhead but requiring significantly increased computation and resource demands.

Random Inquiry Mode: At the far end of the spectrum, a random sampling algorithm may just randomly select a fraction of hosts to reply to the query, as depicted in Fig. 1(e). These protocols can be parameterized by specifying how many or what fraction of nodes should take part in the query.

The forward function is the same as in flooding. However, the respond function needs to ensure that only $k\%$ of the query receptions are replied to:

$$r_{random}(\theta) \triangleq (\theta < k)$$

$$\mathcal{I}_{random} = \langle true, r_{random}(rand)\rangle$$

The value of k is used as decision criteria and a randomly generated number, *rand*, is passed in as a parameter. The random inquiry mode reaches only randomly selected hosts, radically reducing communication complexity.

3 Inquiry Introspection

Using different inquiry modes can yield different results for the same query. The suitability of an inquiry mode is determined by the needs of the querying application and may depend on the dynamics of the environment. We define *query introspection* as the use of information about a query's result to determine if the associated inquiry mode meets the application's needs. In this section, we discuss the use of adequacy metrics, which compare query results against application expectations, to support query introspection.

3.1 An Informal Introduction to Query Introspection

Different inquiry modes provide different sets of tradeoffs as they collect information from a distributed network. The inquiry mode selected depends on an application's needs; how well a particular inquiry mode meets the needs of the application is dependent, in part, on the environment during query execution.

For example, a query issued by a construction supervisor may determine the concentration of a dangerous compound on a construction site. If it is important to minimize the query's overhead, the application may use a random inquiry mode to collect concentration readings from randomly selected hosts across the site. However, given the environmental conditions in which the query operates, the random inquiry mode may not provide results that are accurate enough. In a dense network of spatially-correlated values, a random inquiry mode would likely provide a representative result; the same query may not provide an accurate enough view of a sparse network to enable a decision about site safety.

This kind of analysis can be supported through query introspection, using feedback about query execution to determine if an inquiry mode is appropriate. The ideal approach to evaluating an inquiry mode's tradeoffs is to determine how well results reflect the ground truth of the environment during the query's execution, but this is impractical. Instead, query introspection examines query results directly, using a history that may include results for this query and for queries recently executed using the same inquiry mode. Since each host's contribution to a query result contains information about its context, we can consider properties of the execution environment relevant to the application's query execution needs. For example, when using a random sampling inquiry mode to collect a sample of spatially correlated sensor readings, query introspection can be based on context information that describes the density of the network around each responding node to determine if the results are representative.

Using a history of query results, we can approximate a view of the world that can be used to determine if an inquiry mode is appropriate in the current environment given the application's needs. Query introspection, then, can be achieved by applying an adequacy metric over similar queries' results. In the next section, we formalize query introspection and provide examples of adequacy metrics that can be used in the introspection process.

3.2 Formalizing Query Introspection

Query introspection is the process of determining if an inquiry mode is suitable. This decision is often related to a tradeoff between the desired properties of the result and the cost of query execution. A desired property may be that the results are representative samples; another is that the results are relatively stable and do not change rapidly over time. Variability in the network topology, the query's execution context, and randomness introduced by the inquiry mode can impact how well results delivered by a query reflect the desired properties.

To support quantifiable introspection, we apply adequacy metrics to query results. An adequacy metric, d, measures the logical distance between the desired property of a query's execution and the actual properties of the achieved query result. For each distance function, an associated threshold (δ) can be defined by the application to aid in evaluation. This simple construction supports expression of arbitrary adequacy metrics that can enrich decision-making processes.

We identify two categories of adequacy metrics. The first measures the quality of the data captured by a query based on the variability between successive

queries. The second compares an idealized property of the execution environment to the environment's measured state during query execution. This set of metrics is not exhaustive; rather, we intend to illustrate applications' needs and to provide a framework for identifying and expressing adequacy metrics.

Data Capture Quality Metrics. Often, decisions regarding the appropriateness of an inquiry strategy are related to the desired quality of the result. One way to measure the quality of query results is to define an adequacy metric based on a measurement of the changes in the captured data due to network variability during query execution. This type of adequacy metric can be supported by constructing a baseline for comparison using the results of a previously executed query. In general, this kind of distance metric can be expressed as: $d(\mathcal{P}_j(\rho_j), \mathcal{P}_k(\rho_k))$, where ρ_j is the result for the j_{th} query issued in a sequence of queries that employ the same inquiry mode, $\mathcal{P}_j$ maps a desirable property of the result to a numerical value, and $j \leq k$.

Set Difference. Consider a construction site supervisor that uses a probabilistic inquiry mode to return an inventory of available materials; however, the supervisor realizes that when the network is highly dynamic, this type of inquiry mode is not sufficient to achieve a high-quality result that presents an accurate view of the inventory. To support this kind of introspection, the quality of results can be assessed by measuring variability between different collections of results. The set difference metric quantifies the percentage of items returned by a query that are newly available, have been modified, or are no longer reachable in comparison to previously collected results. We can express a distance metric that determines the number of new results as:

$$d = \frac{\mid \rho_k \mid}{\mid \rho_k - \rho_j \mid}$$

where the k^{th} query is the most recently issued in a sequence of queries using the same inquiry mode, and $j < k$. The threshold δ associated with this metric depends on application needs.

The set difference operator can similarly be used to describe how many result elements have departed between the submission of a previous query and the current query. In our construction site scenario, the amount of bricks available on-site is likely to remain steady until a job consuming them begins, so the construction site supervisor may initially use a random sampling inquiry mode to check the inventory of bricks. The supervisor can use the departure distance function to determine when a job on the site that consumes bricks has begun, and can alter the inquiry mode if necessary to more closely track brick use.

We can also define an adequacy metric based on a cumulative view across an entire sequence of returned results. For example, we can define an adequacy metric based on transitive departures, which counts the number of hosts that departed between the beginning of the execution of query i and the conclusion of query k, even if the hosts later return. This can be captured as:

$$d = \frac{\mid \rho_k \mid}{\mid (\bigcup_{i=1}^{k} \rho_i) - \rho_0 \mid}$$

Transitive additions can be specified similarly. These transitive distance metrics allow a construction site supervisor, for example, to make a decision regarding the use of the inquiry mode to more closely monitor inventory after an influx of supplies due to a delivery or the departure of supplies to another site.

Aggregate distances. Another way to describe the quality of a query's result is to measure the distance between the aggregated numerical values of a previous result and the aggregated values for the current result. Such aggregate distance measures can define adequacy metrics based on trends in the reported results. For example, a construction supervisor may initially use a probabilistic inquiry mode to monitor the total level of hazardous chemicals. If the hazardous chemical level rises at a rate that implies a leak, the probabilistic inquiry mode may no longer be suitable; given the potential danger, an inquiry mode which gives more accurate view of the world is needed regardless of the associated cost. To make this kind of assessment, an aggregate distance metric can compute the distance of an aggregate (e.g., a total) between two queries:

$$d = |\langle \mathbf{sum}\ p : p \in \rho_{i+1} :: p \rangle - \langle \mathbf{sum}\ p' : p' \in \rho_i :: p' \rangle|$$

Variations on this metric can be applied to other aggregates such as count, minimum, maximum, and average.

Environmental Property Metrics. The previous class of adequacy metrics are concerned with the quality of a query's result. These metrics defined the distance between the most recent query result and an approximate view of the ground truth. Here, we describe adequacy metrics that evaluate the distance between a query result and a desired property of the execution environment during query processing. These metrics can be defined in terms of the distance between an ideal environment and the conditions under which the query executed, as captured by the context field ζ for each host tuple in the result.

Network Coverage. In some cases, a query's network coverage is important in determining the suitability of its inquiry mode. A query covers an area if the elements of the result form a connected graph that spans the associated geographic region. In regions of the network where the data is dense, an area can be covered by an inquiry strategy that selects a small subset of hosts. This would yield a query result that is representative of the data in the region while reducing the associated overhead. In sparse regions of the network, a query result may cover a region only if all hosts in the region report results. The query issuer can use query introspection to determine if results obtained using a particular inquiry mode provide a reasonable representation of the surrounding area.

While finding the minimum set of nodes that cover a geographic region is an NP-hard problem, it is still possible to provide an adequacy metric based on whether the results approximately satisfy a network coverage constraint using information about the density of results. We can get a rough estimate of this density through average numbers of neighboring results; a relatively high average results suggests that the results are dense, while a low average suggests that results are sparse. To provide an example based on location, we first define a connectivity relation over host locations that defines the existence of communication links between hosts. A physical connectivity relation that represents a

connectivity model with a circular, uniform communication range can be defined using the location variable λ from a host's context ζ:

$$(h_{\iota_0}, h_{\iota_1}) \in \mathcal{K} \Leftrightarrow |(h_{\iota_0}.\zeta.\lambda) - (h_{\iota_1}.\zeta.\lambda)| \leq b$$

where b refers to a bound on the distance between two hosts to consider them connected. We can roughly determine the density of the query results by averaging across the number of one-hop neighbors, which we compute by applying the connectivity relation to hosts in the query result. This characterization of network density can be expressed as:

$$n = \frac{\langle \mathbf{sum}\ h_{\iota_0}, h_{\iota_1} : h_{\iota_0} \in \rho_i \wedge h_{\iota_1} \in \rho_i :: 1 \rangle}{\langle \mathbf{sum}\ h : h \in \rho_i :: 1 \rangle}$$

The distance metric can be expressed simply as the difference between n and the ideal number of neighbors (i.e., $d = |ideal - n|$). The application can dictate how the average connectedness measure relates to ideal network density and define a threshold that determines adequacy.

Semantic Discovery. It may also be desirable to determine inquiry strategy suitability based on the presence of a particular value:

$$n = \langle \mathbf{sum}\ p : p \in \rho_i \wedge p = v :: 1 \rangle$$

where v is the desired value. A distance metric can be specified as $d = 1 - n$. For an application that wishes to be alerted of the presence of a single value, the associated threshold is specified as $\delta = 0$.

The value of interest may not be directly related to a query's reported result. Instead, causal relationships between different values may be the basis for adaptation. For example, the discovery of smoke on a construction site implies the presence of fire. If smoke is detected, then a inquiry mode which trades accuracy for overhead should be abandoned in favor of one that provides a higher accuracy in a search for a fire. These causal relationships can be captured in an adequacy metric by collecting and evaluating a metric over causally related values in the context field ζ of responding hosts.

4 Implementing Inquiry Modes and Introspection

We have implemented our model using Java[5]; we use the public interface presented below to demonstrate how our model can be realized in practice in a real application example. Fig. 2 shows this public interface, which includes a definition of a `Query` and its `InquiryMode`. Implementations of inquiry modes like the ones described in Section 2 extend these abstract classes with concrete functionality. For this discussion, consider a scenario where a construction site supervisor has deployed sensors across the site over which he issues queries to monitor the concentrations of hazardous airborne materials. The manner in which queries

[5] The source code and settings used are available at `http://mpc.ece.utexas.edu/InquiryMode/index.html`

are processed should differ depending on the conditions on the site; our model captures this in a changing inquiry mode.

Phase 1: Initially, the supervisor may use a query with these characteristics:

- *Forward Function:* Probabilistic (Parameters: $p = 0.7$)
- *Respond Function:* Random Sampling (Parameters: $k = 0.5$)
- *Introspection:* Aggregate Data Capture quality (Parameters: $\delta = 0.1$)

To execute this query, concrete implementations of the probabilistic forward and random respond functions (such as the one shown in Fig. 3) need to be provided.

When a node receives a query, it executes the query's forward and respond functions. If the forward function returns true given the host's current context, the host forwards the query. If the respond function returns true, the node processes the query at the application level. This entails updating the query with its data value(s) and the cost as defined by the introspection type. In this phase, the introspection cost is simply an aggregate on data quality, so no metadata outside the query result is required.

This query is adequate for baseline monitoring. The forward and respond functions ensure that only a fraction of all the devices are involved in query processing, reducing resource usage. By choosing data quality as the introspection strategy, the supervisor can keep issuing such low cost queries until there is an indication of variance in data quality. In our example, a 10% change in the concentration indicates a chemical leak.

Phase 2. When a leak is detected, more serious monitoring is warranted. The first step is detecting where the leak emanates from.

- *Forward Function:* Flooding (Parameters: None)
- *Store Function:* Flooding (Parameters: None)
- *Introspection:* Spatial Coverage (Parameters: 10 units)

By issuing such a query, the supervisor spends more resources by employing a flooding based inquiry mode to detect the location of the leak. By correlating

```
class Query {
    public Query(InquiryMode inquiry, Introspection metadata);
}
class InquiryMode {
    public InquiryMode(ForwardFunction f, RespondFunction s);
}
abstract class ForwardFunction {
   abstract public boolean Execute(Context nodeContext);
}
abstract class RespondFunction {
   abstract public boolean Execute(Context nodeContext);
}
```

Fig. 2. The Inquiry and Introspection API

```
class FowardProbabilistic extends ForwardFunction {
    double p;
    FowardProbabilistic(Conext c) {
        p = c.probThreshold;
    }
    public boolean Execute(Context nodeContext) {
       return (nodeContext.getRandomNumber() < p);
    }
}
```

Fig. 3. Defining a probabilistic forward function

	$\mathbf{P1_1}$	$\mathbf{P1_2}$	$\mathbf{P1_3}$	$\mathbf{P2_1}$	$\mathbf{P3_1}$	$\mathbf{P3_2}$
Data Response	484	504	559	604	609	598
Spatial Coverage	$\emptyset$	$\emptyset$	$\emptyset$	$(900, 915), (805, 311), ...(700, 232)$	$\emptyset$	$\emptyset$
Cost	19	10	23	175	45	51

Table 1. Case Study query results by phases

those regions (obtained from the metadata information) with the response values, the application can establish regions of leakage.

Phase 3. Once the leak has been localized, the application can adapt the query once again to focus on the area of the leak:

- *Forward Function:* Location (Parameters: Area enclosing quadrant of site)
- *Store Function:* Location (Parameters: Area enclosing quadrant of site)
- *Introspection:* Data Quality (Parameters: $\delta = 0.2$)

Only devices in the vicinity of the leak respond to the query, and the user can get more detailed data from that region. In addition, this data can be collected more quickly since the network is focusing on a smaller amount of communication.

Results. Table 1 highlights the values for responses, metadata, and cost observed during the execution of this case study; in the table, $P1_2$ refers to the second query issued in Phase 1. When the first query is issued, the average concentration of chemicals is determined. Subsequent queries show a jump from an average that exceeds the threshold of 10% stipulated by the application. The application switches to Phase 2's flooding query, and the introspection strategy determines the query's spatial coverage. The supervisor identifies the location coordinates that have a high value for the amount of chemicals sensed. Once the areas of interest are located, he issues queries to get data only from those locations. At every step, he can evaluate the data against the cost of obtaining it. The cost expressed here is the number of messages to obtain the query result and its associated metadata. When the supervisor is obtaining just the data elements and using a low cost inquiry mode like random sampling, he incurs a lower cost. However, the cost increases to a higher value (175) when a more expensive inquiry mode like flooding is used. However, once this information is

used to locate the area of interest, the supervisor can revert to an inquiry mode that restricts the overhead by scoping the query to a particular region of interest. This example demonstrates that introspection can provide great flexibility to the application developer and has the potential to save resources. Introspection thus provides an important mechanism in analyzing feedback, which is critical to developing intelligent adaptive systems.

5 Related Work

Our approach takes a novel perspective on modeling queries in mobile ad hoc networks by defining two new concepts: inquiry modes and query introspection. In this section, we examine related approaches with respect to modeling queries in these dynamic environments and adapting querying techniques.

Several related approaches to modeling dynamic environments rely on process calculi [7, 8] or petri nets [9]. The former tend to focus on evolutionary changes (like our configuration changes), but make it difficult to capture the impact of time, space, and other constraints on query processing. The latter focus on low-level aspects of the environment such as packet transmission and energy consumption, lacking constructs to capture query processing behavior. More closely related work on coordination techniques for mobile and disconnected environments defined a concept similar to our reachability [10]: *disconnected routes*, which allow decoupling of mobile communication in space and time. The model, however, focuses on using the availability of motion profiles to plan nodes' interactions over time. Existing work in applying query processing to these dynamic environments focuses primarily on mobile distributed databases and the ability (or inability) of a system to provide traditional strong semantics (e.g., [11]). Our approach explicitly separates dissemination (through our forward function) from result generation (through our respond function). This approach in essence treats the entire network as a global virtual data structure, which is more in line with approaches targeted to database abstractions for sensor networks (e.g., [12]).

Our work also has some similarities to what can be broadly categorized as stream processing systems. Some of this work has explored model-driven query processing [13] where each node constructs a local model of the data available. If the estimated error of the model is below a threshold, a node processes a query over the local data model to avoid consuming resources. A model driven approach is less suitable for mobile environments because of the inherent unpredictability of movement. Our formalization of introspection provides a more systematic approach to exposing relevant adequacy metrics (both data and network related) to facilitate adaptivity. By evaluating these metrics over values, informed decisions can be made on the trade-off between the cost of executing a particular query against application needs and switch inquiry modes as application requirements change. Similarly, *reflection* is common in mobile computing middleware and models [14]; our work recognizes the importance of reflection to the adaptivity of mobile applications and provides a formal foundation for

exposing information about query results to applications through a principled use of introspection.

6 Conclusions

In this paper, we have presented a new perspective on query processing in mobile and pervasive networks. We presented a formalism for the concept of an *inquiry mode* that defines which devices participate in query resolution. In addition, we also formally defined the notion of *introspection* that helps evaluate the tradeoff between the cost of a chosen inquiry mode and its effectiveness by exposing relevant metadata in the form of adequacy measures. We showed how our model can be used to specify a variety of real world inquiry modes and adequacy measures. In addition, we provided a Java implementation that helps realize our model for practical application development and demonstrated its effectiveness.

References

1. Payton, J., Julien, C., Roman, G.C.: Automatic consistency assessment for query results in dynamic environments. In: Proc. of FSE. (2007)
2. Johnson, D.B., Maltz, D.A., Broch, J.: Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. Ad Hoc Networking **1** (2001) 139–172
3. Perkins, C., Royer, E.: Ad hoc on-demand distance vector routing. In: Proc. of WMCSA. (February 1999)
4. Ni, S.Y., Tseng, Y.C., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. In: Proc. of MobiCom. (1999) 151–162
5. Roman, G.C., Julien, C., Huang, Q.: Network abstractions for context-aware mobile computing. In: Proc. of ICSE. (2002) 363–373
6. Kyasanur, P., Choudhury, R., Gupta, I.: Smart gossip: An adaptive gossip-based broadcasting service for sensor networks. In: Proc. of MASS. (October 2006)
7. Braione, P., Picco, G.P.: On Calculi for Context-Aware Coordination. In: Proc. of Coordination. (2004) 38–54
8. Lopes, L., Martins, F., Silva, M., Barros, J.: A process calculus approach to sensor network programming. In: Proc. of Sensorcomm. (2007) 451–456
9. Xiong, C., Murata, T., Tsai, J.: Modeling and simulation of routing protocols for mobile ad hoc networks using colored petri nets. In: Proc. of Wkshp. on Formal Methods Applied to Defense Systems. (2002) 145–153
10. Roman, G.C., Handorean, R., Sen, R.: Tuple space coordination across space and time. In: Proc. of Coordination. (2006) 266–280
11. Dunham, M., Helal, A., Balakrishnan, S.: A mobile transaction model that captures both the data and movement behavior. ACM-Baltzer Journal on Mobile Networks and Applications **2**(2) (October 1997) 149–161
12. Madden, S., Franklin, M., Hellerstein, J., Hong, W.: The design of an acquisitional query processor for sensor networks. In: Proc. of the 2003 ACM SIGMOD Int'l. Conf. on Management of Data, ACM Press (2003) 491–502
13. Deshpande, A., Guestrin, C., Madden, S., Hellersetin, J., Hong, W.: Model-driven data acquisition in sensor networks. In: Proc. of VLDB. (2004)
14. Capra, L., Blair, G.S., Mascolo, C., Emmerich, W., Grace, P.: Exploiting reflection in mobile computing middleware. ACM SIGMOBILE Mobile Computing and Communications Review **6**(4) (October 2002) 34–44