Searchable Querical Data Networks

Farnoush Banaei-Kashani and Cyrus Shahabi

Computer Science Department,
University of Southern California,
Los Angeles, California 90089
{banaeika,shahabi}@usc.edu

Abstract. Recently, a new family of massive self-organizing networks has emerged that not only serve as a communication infrastructure, but also mainly as a distributed query processing system. We term these networks *Querical Data Networks (QDNs)*. Peer-to-peer networks are examples of QDN. In this paper, first we identify and characterize QDN as a new family of data networks with common characteristics and applications. Subsequently, as the first step toward realizing the evolved vision of QDN as a large-scale distributed query processing system, we propose an efficiently searchable QDN model based on a recently developed small-world model. We discuss in details how our QDN model enables effective location of the data relevant to a QDN query.

1 Introduction

Recently, a new family of massive self-organizing networks has emerged that not only serve as a communication infrastructure, but also mainly as a distributed query processing system. These networks are significantly different with classical engineered networks (e.g., the Internet) both in the way they are applied and in the characteristics of their components (nodes and links). We term these networks *Querical Data Networks (QDNs)*. Peer-to-peer networks and sensor networks [9, 2] are examples of QDN.

This paper is organized in two parts. In the first part, we provide our overview, where we 1) define and characterize QDN as a new family of data networks with common characteristics and applications (Section 2), and 2) review possible database-like architectures for QDNs as querying systems, and discuss design principles and implementation issues in realizing those architectures (Section 3). In the second part (Section 4), as the first step toward realizing the evolved vision of QDNs we focus on a specific problem, namely the problem of effective data location (or search) for efficient query processing in QDNs.

We believe that with QDNs, search optimization is not limited to optimal query routing on the unintelligently generated ad hoc network topology. Considering the flexibility (hence, controllability) of the topology of QDNs (see Section 2), a QDN can self-organize to a *search-efficient topology* that in combination with an intelligent query routing/forwarding mechanism can significantly improve the *searchability* of the QDN. Here, with our searchable QDN model, first

we propose a self-organization mechanism that generates a *small-world* QDN based on a recently developed small-world model [29]. Small-world networks are known to be extremely search-efficient [28]. Although DHTs [20, 27, 22, 31] are also successful in creating efficient topologies, we argue that as a model they are not compatible with characteristics/requirements of QDNs. Particularly, DHTs enforce certain data distribution and replication schemes and impose strict connectivity rules to QDN nodes. On the other hand, the small-world model is the *natural* model for QDNs that completely respects the autonomy of QDN nodes. Second, we complement the generated small-world network topology with query forwarding mechanisms that effectively route the query toward the QDN nodes that store target data relevant to the submitted query.

2 Querical Data Networks (QDNs)

Recently, a family of massive, self-organizing, engineered networks has emerged that bear componental characteristics (e.g., node lifetime) that are significantly different with those of the classical engineered networks such as the Internet, but similar to those of their natural counterparts, so-called *complex systems* such as social networks and biological networks. Besides componental differences, these networks are also *applied* differently as compared with the classical networks. A collection of interrelated data is distributed among the nodes of these networks, where the data are naturally created, collected (e.g., by sensing the physical world), and/or stored. Database-style queries are frequently posed (by nodes themselves, or by an outsider) to retrieve the data or information (i.e., the processed data) from this distributed network of data sources. Hence, the network not only serves as the communication infrastructure, but also mainly and more importantly as a distributed data source and a distributed query processing system. We term these emerging networks *Querical Data Networks (QDNs)*. Here, we enumerate the main componental characteristics and application features of a QDN.

2.1 Componental Characteristics

A network is an interconnection of nodes via links, usually modelled as a graph. Nodes of a QDN are often massive in number and bear the following characteristics:

- *Peer functionality*: All nodes are capable of performing a restricted but similar set of tasks in interaction with their peers and the environment, although they might be heterogeneous in terms of their physical resources. For example, joining the network and forwarding search queries are among the essential peer tasks of every node in a peer-to-peer network.
- *Autonomy*: Aside from the peer tasks mentioned above, QDN nodes are autonomous in their behavior. Nodes are either self-governing, or governed by out-of-control uncertainties. Therefore, to be efficacious and applicable the QDN engineering should avoid imposing requirements to and making

assumptions about the QDN nodes[1]. For example, strict regulation of connectivity (i.e., number of connections and/or target of connections) might be an undesirable feature for a QDN design.
- *Intermittent presence*: Nodes may frequently join and leave the network based on their autonomous decision, due to failures, etc.

On the other hand, links in various QDNs stand for different forms of interaction and communication. Links may be physical or logical, and they are fairly inexpensive to rewire. Therefore, a QDN is "a large-scale federation of a dynamic set of peer autonomous nodes building a transient-form interconnection". Traditional modelling and analysis approaches used for classical networks are either too weak (oversimplifying) or too complicated (overcomplicated) to be effective with large-scale and topology-transient QDNs. The *complex system theory* [5], on the other hand, provides a set of conceptual, experimental, and analytical tools to contemplate, measure, and analyze systems such as QDNs[2].

2.2 Application Features

A QDN is applied as a distributed source of data (a *data network*) with nodes that are specialized for cooperative query processing and data retrieval. The node cooperation can be as trivial as forwarding the queries, or as complicated as in-network data analysis (see Section 3). In order to enable such an application, QDN should support the following features:

- *Data-centric naming, addressing, routing, and storage*: With a QDN, queries are declarative; i.e., query refers to the names of data items and is independent of the location of the data. The data may be replicated and located anywhere in the data network, the data holders are unknown to the querier and are only intermittently present, and the querier is interested in data itself rather than the location of the data. Therefore, naturally QDN nodes should be named and addressed by their data content rather than an identifier in a virtual name space such as the IP address space. Consequently, with data-centric naming and addressing of the QDN nodes [13], routing [16] and storage [19] in QDN are also based on the content. It is interesting to note that non-procedural query languages such as SQL also support declarative queries and are appropriate for querying data-centric QDNs.
- *Self-organization for efficient query processing*: QDNs should be organized for efficient query processing. A QDN can be considered as a database system with the data network as the database itself. QDN nodes cooperate in processing the queries by retrieving, communicating, and preferably on-the-fly processing of the data distributed across the data network (in Section 3,

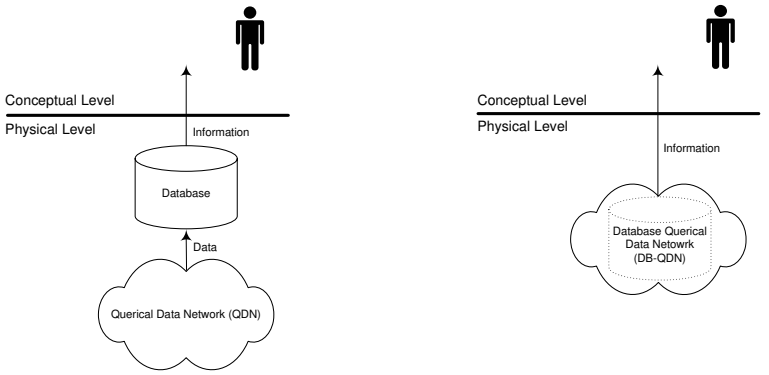
[1] One can consider peer tasks as *rules of federation*, which govern the QDN but do not violate autonomy of individual nodes.

[2] Further discussion about the complex system theory is beyond the scope of this paper. We refer the interested reader to our work on complex-system-based modelling of the peer-to-peer networks in [4].

we explain the benefits of adopting a database system framework to discuss and design query processing in QDNs). To achieve efficiency in query processing with high resource utilization and good performance (e.g., response time, query throughput, etc.), QDN should be *organized* appropriately. In Section 4, we discuss an example organization, where the topology of QDN is structured such that search for data items is performed efficiently. Other examples of organization are: intelligent partitioning of the query to a set of sub-queries to enable parallel processing, or collaborative maintenance of the data catalogue across the QDN nodes. However, the peer tasks of the QDN nodes should be defined such that they *self-organize* to the appropriate organization. In other words, organization must be a collective behavior that emerges from local interactions (defined by peer tasks) among nodes, otherwise the dynamic nature and large scale of QDN renders any centralized micro-management of the QDN unscalable and impractical.

3 Database-System Querying Framework for QDNs

The idea of adopting a DBS querying framework was originally discussed briefly and abstractly in two recent position papers for some specific cases of QDN [10, 12]. Here, we define a taxonomy of approaches to generalize this querying framework for the entire family of QDNs.



a. Centralized Design b. Decentralized Design

Fig. 1. Database System Framework for Querying Querical Data Networks (QDNs)

3.1 Taxonomy

Based on the two fundamentally distinct design choices for the physical level of the DBS framework (i.e., centralized and decentralized), we recognize two approaches to implement a DBS-based querying system for QDNs:

1. Database *for* QDN (Figure 1a): This approach corresponds to the querying systems with centralized query processing. These systems are similar to other centralized database applications, where data are collected from some data sources (depending on the host application, the data sources can be text documents, media files, and in this case, QDN data) to be separately and centrally processed.
2. Database-QDN (DB-QDN) (Figure 1b): The systems that are designed based on this approach strive to implement query processing in a decentralized fashion within the QDN; hence, in these systems "*QDN is the database*".

3.2 Designing a Database-QDN (DB-QDN)

By definition QDNs tend to be large-scale systems and their potential benefits increase as they grow in size. Therefore, between the two types of DBS-based querying systems for QDNs, the database-QDNs (DB-QDNs) are more promising because they are scalable and efficient. Here, we discuss the design issues with these systems.

Design Principles Among the design principles for distributed query processing at QDNs one can distinguish the following:

1. *In-network query processing*: In-network query processing is the main distinction of DB-QDNs. In-network query processing techniques should be implemented in a distributed fashion, ensuring minimal communication overhead and optimal load-balance.
2. *Transaction processing with relaxed semantics*: Due to the dynamic nature of QDNs, requiring ACID-like semantics for transaction processing in DB-QDNs is too costly to be practical and severely limits scalability of such processing technique. Hence, transaction processing semantics should be relaxed for DB-QDNs.
3. *Adaptive query optimization*: Since QDNs are inherently dynamic structures, optimizing query plans for distributed query execution in DB-QDNs should also be a dynamic/adaptive process. Adaptive query optimization techniques are previously studied in the context of central query processing systems [3].
4. *Progressive query processing*: Distributed query processing tends to be time-consuming. With real-time queries, user may prefer receiving a rough estimation of the query result quickly rather than waiting long for the final result. The rough estimation progressively enhances to the accurate and final result. This approach, termed *progressive* query processing [24], allows users to rapidly obtain a general understanding of the result, to observe the progress of the query, and to control its execution (e.g., by modifying the selection condition of the query) on the fly.
5. *Approximate query processing*: Approximation techniques such as *wavelet-based* query processing can effectively decrease the cost of the query, while producing highly accurate results [25]. Inherent uncertainty of the QDN data together with the relaxation of the query semantics justify application of approximation techniques to achieve efficiency.

Implementation Issues

Operators In DB-QDNs, the data is distributed among QDN nodes. An operator is executed by a set of selected QDN nodes, which receive the query (or sub-queries), and perform partial processing and communicate with each other to generate the final result. Therefore, regardless of the type of the operator, the mechanism that defines the implementation of an operator should address the following questions:

1. Which QDN nodes should receive the query (or sub-query)?
2. Which sub-query should each node receive?
3. How should the nodes collaborate to process the query? Or, what is the set of local computation and communication tasks (i.e., peer tasks) that every QDN node should perform such that QDN nodes cooperatively compute the result of the query with optimal cost and performance?

Query Plan Construction and Optimization With centralized database systems the metrics for query optimization are usually user-centric; e.g., throughput and response time. However, with QDNs physical resources of the system are seriously restricted. Therefore, besides user-centric metrics the query optimizer should consider "resource utilization" as a new optimization metric (probably with higher priority). With these systems, the query optimizer should also optimize the partitioning of the QDN processing power among several parallel operators in a query plan, and in case of multi-user systems, among multiple query plans.

4 QDN Searchability

Toward realizing a Database-QDN (DB-QDN), here we address the first implementation issue mentioned in Section 3.2, *"which QDN nodes should receive the query?"*. This problem arguably reduces to the problem of locating the nodes containing the data relevant to the query. The query declares the required data and the QDN must be *efficiently* searched to locate the corresponding nodes. We term a QDN that self-organizes for efficient search a *searchable* QDN.

To measure searchability, we introduce the following metrics:

1. *Precision and Recall*: Considering *Ret* as the set of QDN nodes that receive the query during the search for data, and *Rel* as the set of nodes that are intended to receive the query (because they store relevant/target data required to process the query), precision (**P**) and recall (**R**) of a search are defined according to the classical definitions of precision and recall:

$$\mathbf{P} = \frac{|Rel \cap Ret|}{|Ret|} \times 100 \qquad \mathbf{R} = \frac{|Rel \cap Ret|}{|Rel|} \times 100$$

Assuming uniform distribution of the target data among the relevant QDN nodes, **P** and **R** are correlated with the precision and recall of the retrieved

data, respectively. The limited amount of data stored at each node of a typical QDN (e.g., sensor network) justifies such a uniform distribution.
2. *Hop-count*: We use hop-count (**H**) as an abstract measure of the search time. Nodes forward the query to their neighbors once at each hop (or time slot). The number of hops a search takes (or is allowed to take, otherwise it will be terminated) to execute is the hop-count of the search.

The combination of these two metrics capture both resource-efficiency and user-centric performance (e.g., response time) of the search. With a fixed hop-count (i.e., limited time for search), higher precision and recall demonstrates better use of the resources; on the other hand, the fewer hops required by a search to achieve a target precision and/or recall, the better is the response time of the search. For example, the flooding-based search is expected to have low precision but high recall in low hop-count. In contrast, a search based on random walk may result in higher precision but lower recall, and requires more hops to achieve high recall.

Here, first we discuss how to organize a QDN for searchability and compare two different approaches to introduce such organization into QDN. Thereafter, we propose a searchable QDN model compatible with one of these approaches.

4.1 Organizing QDNs for Searchability

Without organization, a QDN that is generated by random interconnection of nodes is not efficiently searchable. With such a QDN, since the required data can be located anywhere in the network, nodes lack a sense of direction to forward the query toward the node(s) that potentially hold the required data. All neighbors of a node may equally lead the path to the target data. Therefore, nodes either pick a neighbor randomly and forward the query (i.e., random walk), or forward the query to all neighbors (i.e., flooding). In both cases, search is brute-force.

As we mentioned in Section 2, unlike classical networks, with QDNs the topology is flexible and rewiring the links is fairly inexpensive. Therefore, search optimization is not limited to optimal routing on the available network topology. A QDN can self-organize to a *search-efficient topology*, which in combination with an intelligent and data-centric query routing/forwarding mechanism can significantly improve the searchability of the QDN. To create a search-efficient topology, interconnection between nodes is correlated with "similarity" of the data content of the nodes: the more similar the data content of two nodes, the more likelihood of interconnection. Consequently, with sufficient correlation, nodes with similar data content are clustered to generate highly connected components that correspond to data localities, and the topology is a connected graph of interwoven data localities. Such a topology enhances the searchability in two ways:

1. Directed forwarding: Assuming that a node has (local) knowledge about the data content of the nodes in its neighborhood, the node can selectively forward the query to the neighbor with most similar data to the target data (i.e., data required to process the query). Due to the correlation between

data similarity and node connectivity, it is more likely that the selected neighbor lead to the data locality corresponding to the target data. Therefore, the query is effectively routed toward the target data locality and avoids irrelevant paths. Hence, the precision **P** of the search increases.

2. Batch data access: When the query approaches the target data locality, most probably a large collection of the target data can be located all together within a few hops. Hence, the recall **R** of the search is improved too.

Approaches There are two philosophically different approaches to organize a search-efficient topology: *fabricated* organization, and *natural* organization (i.e., an organization approach that respects natural characteristics of the system components)[3]. Distributed Hash Table (DHT) [20, 27, 22, 31], which is proposed as a model for peer-to-peer networks, is an example of fabricated organization. To generate data locality, with DHT both nodes and data items are randomly assigned unique identifiers from a virtual identifer space. Each data item (or pointer to the data item) is stored at the node with the closest identifier to the identifier of the data item. Finally, nodes are interconnected via a *regular* topology, where nodes that are close in the identifier space are highly interconnected. DHT topology is regulated: all nodes have the same connectivity (same number of neighbors), and selection of the neighbors is strictly dictated by the semantics of the DHT interconnections.

Although DHTs (as the most important realization of the fabricated organization approach) are very successful in creating efficient topologies, we argue that as a model DHT fails to respect natural characteristics/requirements of QDNs. For example, data-to-node assignment via the virtual identifier space violates the natural distribution and replication of the data, where each node autonomously maintains its own and only its own data. Also, regular topology of the DHT imposes strict connectivity rules to autonomous nodes. While similarity of data content itself can be used to generate data localities, using virtual identifier space to introduce virtual similarity is not only unnecessary but also problematic (as mentioned above). Moreover, strict regulation of the topology must be relaxed to allow more flexible connectivity rules that enable trading off efficiency against natural flexibility. In Section 4.2, we provide an example of such a natural organization, which improves searchability of QDNs while respecting their characteristics.

4.2 Our Searchable QDN Model

To develop a *natural* model for searchable QDNs, we adopt and generalize a social network model [29] that is proposed recently to explain the small-world phenomenon in social networks[4]. Here, assuming a relational data model for

[3] In continuation of our QDN analogy to social networks, one can compare socialistic and capitalistic societies with fabricated and natural QDNs, respectively!

[4] In social networks, where individuals are represented as nodes and the acquaintance relationship between them as links, the average length of acquaintance chain

DB-QDN, we use the following query template to explain the basics of the QDN model:

```
SELECT   *
FROM     r
WHERE    AND_{i∈[1,d]}  (A_i = v_i)
```

where $R(A_1, A_2, ..., A_d)$ is a schema of degree d for the relation r stored at DB-QDN, and v_i is a constant. Later in this section, we generalize this model to other query templates with join clauses and disjunctive predicates.

Basics We define our QDN model by explaining the following concepts in sequence:

1. Node identity: We define an *identity* for each node based on its data content;
2. Identity-based linking to create the network: We explain how nodes join the QDN by linking to some other QDN nodes, selecting the nodes of "similar" identity with higher probability; and
3. Identity-based routing to forward the queries: We describe how nodes, which know about the identity of their neighbors, route/forward the query to the neighbor that has the most similar identity to the target data content.

Node Identity The identity of a node is defined such that it represents the data content of the node, just as the cognitive identity of a person which stands for the knowledge-set of the person. For node n, identity I_n is a d-tuple $< a_1, a_2, ..., a_i, ..., a_d >$ defined as follows:

$$I_n = < a_1, a_2, ..., a_i, ..., a_d > = \mathcal{F}(\{t_k | \textit{ tuple } t_k \in r \textit{ is stored at } n\}) \quad (1)$$

where $\mathcal{F}$ is a tuple-aggregation function defined based on a set of d attribute-aggregation functions f_i such that[5]:

$$< a_1, ..., a_i, ..., a_d > = < f_1(\{t_k[A_1]\}), ..., f_i(\{t_k[A_i]\}), ..., f_d(\{t_k[A_d]\}) >$$

The selection of the aggregation functions f_i is application dependent. However, since QDN nodes often have limited storage that stores few tuples, a simple aggregation function such as AVG is sufficiently accurate in generating an identity that represents the data content of the node. Otherwise, we can avoid aggregation altogether, and represent each node as multiple *virtual* nodes by taking

between two individuals is logarithmic to the size of the network. The fact that individuals, which only have local knowledge about the network (i.e., they know their close acquaintances), can effectively and cooperatively find these short paths from a source to a target individual is termed the small-world phenomenon [28]. In addition to the small-world model we adopt here, there are other (more aged) models for this phenomenon [6, 1, 15]; however, we found the model proposed at [29] the most compatible model with the characteristics of QDNs.

[5] Notation: $t_k[A_i]$ is the value of the i-th attribute at tuple t_k. $\{t_k[A_i]\}$ is the set of these values for a set of tuples.

each tuple t_k as a virtual identity. The latter approach increases the size of the QDN, but it is more accurate. Also, with dynamic data content, the identities of the nodes is expected to change. However, since with QDN nodes the data content is often highly autocorrelated, frequent modification of the identity is unnecessary.

In subsequent sections, we explain how we use node identities to 1) organize the network topology to localities of nodes with similar identities (i.e., data content), and 2) route the queries to the node localities that store data content similar to the target data of the query.

Identity-based Linking We assume the domain D_i of the attribute A_i is a metric space with the similarity measure S_i as the distance function of the space; for example, Hamming distance, Euclidean distance, or even the absolute difference $S(a,b) = |a-b|$ as the similarity measure for numbers, and Edit distance as that of strings. We also assume that node n is potentially able to (or desired to) connect only to nodes from a set of possible nodes Q (e.g., a range of nodes geographically close to n). Node n selects each of its M neighbors as follows: 1) randomly select a dimension i from $[1..d]$, 2) probabilistically select a distance x based on the probability distribution function $p_i(x) = c_i \exp(-\alpha_i x)$, where α_i is a tunable parameter[6] and c_i is the normalization constant, and finally 3) select the node m from Q such that $|x - S_i(I_n[A_i], I_m[A_i])|$ is minimum. The number of neighbors M depends on the local connectivity policy of n and is not enforced by the linking algorithm, although with higher connectivity diameter of the network decreases, and consequently, searchability of the QDN is expected to improve.

Identity-based Routing According to the searchability metrics, when a query is originated at some node, the QDN nodes should forward/route the query, one hop at a time, such that the query: 1) traverses (as many as possible) the nodes with tuples satisfying the selection condition of the query, and 2) preferably avoids other nodes. With the searchable QDN model, each node is aware of the identity of its neighbors. Assume that the selection condition of the query is represented by the tuple $t < v_1, v_2, ..., v_d >$, where $v_i = NULL$ if the attribute A_i is not conditioned. To forward a query, the node n uses the distance measure U to estimate the *semantic* distance of its neighbor m from the target data content (hence, target node)[7]:

$$U(m,t) = \min_{\{i|v_i <> NULL\}} S_i(I_m[A_i], t[A_i]) \tag{2}$$

Since with homophilic linking, the network distance is correlated with the semantic distance, among all neighbors, the node n selectively chooses to forward the query to the neighbor with minimum semantic distance to the target

[6] The parameter α_i can be considered as the measure of *homophily* at dimension i, i.e., tendency of associating identities that are similar at dimension i to each other.

[7] The semantic distance measure U is *ultrametric*, i.e., it may violate the triangle inequality.

data content. We term this routing algorithm *selective walk*. There are three conditions that terminate a selective walk:

1. When the node n receives a query for the first time, it creates and maintains a soft state (which expires, if not refreshed) for the query. If the query is received for the second time, this time n forwards the query to the neighbor with the second to minimum semantic distance to the target, and so on. The $(M+1)$-th time the query is received, it is terminated (i.e., discarded).
2. If the selection condition t of the query (see above) conditions a key of the relation r, the first node that receives the query and locates the target data locally terminates the query, because the target data is expected to be unique.
3. The query carries a TTL (Time-To-Live) value, which is decreased by one each time the query is forwarded. To avoid everlasting looping of the queries, the query is terminated when its TTL equals zero.

Improvements There are many ways to improve the basic searchable QDN model. Here, we discuss two examples:

Weighted Linking With normal linking, at step 1 of the neighbor selection procedure (see above) node n selects a dimension *randomly*, to link to a node with similar identity in that dimension. If the distribution of queries conditioned on each dimension/attribute is not uniform across all dimensions, instead of random dimension selection, node n selects the dimension according to the query distribution. This linking approach, termed *weighted linking*, strengthens the correlation between network connectivity and data similarity in those dimensions that are more frequently used at queries, and consequently, improves the searchability of the QDN in those dimensions.

Wooding Existence of the data localities in the searchable QDN model encourages a new routing algorithm that we term *wooding*[8]. Wooding starts with selective walk, which leads the query toward the target data locality. The first time a target data item is located signifies the approach to the target locality, where nodes with similar data content are clustered. Therefore, to take advantage of the high recall of the flooding in minimum hop-count (while high precision is ensured at the target data locality), it is reasonable that the routing algorithm is switched from selective walk to flooding. Thus, with wooding the node that receives the first hit during selective walk, marks the query for scope-limited flooding and continues forwarding the query by originating the flooding.

As an alternative (more controlled) wooding algorithm, nodes can fork successively more selective walkers as the semantic distance between the local node

[8] "Wooding" is an abbreviation for a hybrid routing algorithm of selective **w**alk and fl**ooding**. The verb "to wood" also means "to rage suddenly", which signifies the switching from walking to flooding to rapidly embrace the target data locality when located.

and the target data decreases. With this approach, starting with a single selective walker, as the query approaches the target data locality the number of query forwarding branches gradually increase such that close to the target the forwarding algorithm resembles flooding. It is important to note that both wooding algorithms are enabled by the linking policy that generates the data localities in the QDN topology, otherwise wooding in random topologies is ineffective.

Other Queries We used a selection-query template to explain our searchable QDN model. Leveraging on the main property of the search-efficient topology, i.e., existence of distinct data localities, the model is expandable to support other query templates. The expansion mainly involves customizing the basic routing and applying a different routing method for each particular query template. Also, for application-specific QDNs, the linking method can be customized to optimize the topology for processing the most frequent query templates. Here, we discuss two examples:

Selection The selection condition in our template query is a simple conjunction of equality literals. More generally, the selection condition is a statement in disjunctive normal form, which consists one or more disjuncts each of which is a conjunction of one or more equality and/or inequality literals. First, to support conjunction of both equality and inequalities literals, the semantic distance function used with selective random walk should be customized as follows:

$$U(m,t) = \min_{\{i|v_i<>NULL\}} \begin{cases} S_i(I_m[A_i], t[A_i]) & (A_i = v_i) \\ (1 - S_i(I_m[A_i], t[A_i])) & (A_i <> v_i) \end{cases}$$

where the distance function S_i is normalized. To process a selection query with disjunctive condition, the source of the query partitions the query to several sub-queries, each as a selection query conditioned by one of the disjuncts, and originates a basic selection query for each sub-query.

Join With a multi-relation DB-QDN, each node has multiple virtual identities, one per each relation. Consider the following join query:

```
SELECT  *
FROM    r, s
WHERE   r.A_i = s.B_j
```

To process such a query, we adopt an approach similar to the nested-loop join. The source initially floods the network with the query. At every node n that receives the query (in parallel with others), for each tuple t of r stored locally a selection query is originated with $s.B_j = t[A_i]$ as the selection condition. Subsequently, node n joins the results of the selection queries with the corresponding local tuples, and forwards the final result to the source. To optimize the query processing for resource efficiency, it remains open to study how nodes at the same data locality (i.e., cluster of nodes that store tuples with similar/identical A_i values) can cooperate to share selection queries for the same A_i value. An

intelligent cooperation scheme can effectively eliminate the redundant queries. This latter problem is an instance of the third-type implementation issues mentioned in Section 3.2.

4.3 Related Work

In [30] and [18], distributed query processing is discussed in the context of sensor networks, but the effect of the network topology in efficiency of the process is not considered. Also, [1] and [17] propose efficient routing schemes (without any particular linking scheme) for search in peer-to-peer networks assuming an ad hoc organization for the network topology; efficient routing mechanisms in combination with the linking mechanisms that create search-efficient topologies are expected to outperform those search schemes significantly. In [14], [21], and [11] DHTs are adopted as the mechanism to organize the network for efficient data access and retrieval. As we discussed in Section 4.1, DHT is an example of fabricated organization, which is not sufficiently compatible with the characteristics and requirements of QDNs. Similarly, [26] and [9] employ the traditional hierarchy scheme, and [23] uses a hypercube structure to organize the network. Here, we propose an organization scheme based on a small-world model, which as a natural organization model completely respects componental characteristics of the QDN nodes and allows highly efficient query processing as exemplified by the well-known small-world phenomenon. Finally, with "semantic overlays" [8, 7], network nodes are clustered into distinct overlays based on the semantic similarity of their data content in order to create data localities. However, this work does not introduce any routing and intra-linking scheme for semantic overlays.

5 Future Work and Conclusion

In this paper, we identified *Querical Data Networks (QDNs)* as a recently emerging family of networks (such as peer-to-peer networks and sensor networks) that not only serve as a communication infrastructure, but also mainly and more importantly as a distributed query processing system. We discussed the design issues with a Database-QDN (i.e., a QDN that implements distributed query processing in network), and as a first step toward realizing a Database-QDN, we focused on the problem of effective data location for query processing. We proposed a searchable QDN model that self-organizes to a small-world topology and allows effective forwarding of the queries to the nodes containing the target data required for query processing.

We intend to continue studying QDNs extensively. In short terms, we focus on the searchability problem by 1) experimental evaluation of our searchable QDN model and behavioral study of the model across the parameter space (e.g., number of data dimensions, homophilic parameter of the linking scheme, etc.), and 2) developing routing (and/or linking) schemes to support other query templates. In long terms, we turn our attention to other implementation problems discussed in Section 3.2, beginning with the study of cooperative query processing schemes that take advantage of the correlation between data placement and

network interconnection in the searchable QDN model to optimize the resource-efficiency of the distributed query processing.

Acknowledgment

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC), IIS-0082826 (ITR), IIS-0238560 (CAREER) and IIS-0307908, and unrestricted cash gifts from Okawa Foundation and Microsoft. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. L. Adamic, R.M. Lukose, A.R. Puniyani, and B.A. Huberman. Search in power-law networks. *Physics Review Letters*, 64(46135), 2001.
2. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8), August 2002.
3. R. Avnur and J.M. Hellerstein. Eddies: continuously adaptive query processing. In *In Proceedings of ACM International Conference on Management of Data (SIGMOD'00)*, May 2000.
4. F. Banaei-Kashani and Cyrus Shahabi. Criticality-based analysis and design of unstructured peer-to-peer networks as complex systems. In *Third International Workshop on Global and Peer-to-Peer Computing (GP2PC) in conjunction with CCGrid'03*, May 2003.
5. Y. Bar-Yam. *Dynamics of Complex Systems*. Westview Press, 1997.
6. A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
7. M. Bawa, G. Manku, and P. Raghavan. SETS: Search enhanced by topic-segmentation. In *Proceedings of the 26th Annual International Conference on Research and Development in Informaion Retrieval (SIGIR'03)*, 2003.
8. A. Crespo and H. Garcia-Molina. Semantic overlay networks. In submission.
9. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceesings of International Conference on Mobile Computing and Networks (MobiCOM'99)*, 1999.
10. R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. The sensor network as a database. Technical Report 02-771, University of Southern California, 2002.
11. A. Gupta, D. Agrawal, and A. El Abbadi. Approximate range selection queries in peer-to-peer systems. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research*, January 2003.
12. M. Harren, J.M. Hellerstein, R. Huebsch, B.T. Loo, S. Shenker, and I. Stoica. Complex queries in DHT-based peer-to-peer networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, 2002.
13. J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operating Systems Principles*, October 2001.

14. R. Huebsch, N. Lanham, B.T. Loo, J.M. Hellerstein, S. Shenker, and I. Stoica. Querying the inernet with PIER. In *Proceedings of 29th International Conference on Very Large Data Bases (VLDB'03)*, 2003.
15. J. Kleinberg. The small-world phenomenon: an algorithmic perspective. Technical Report 99-1776, Cornell University, 2000.
16. B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *Proceedings of INFOCOM'02*, June 2002.
17. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on supercomputing (ICS '02)*, June 2002.
18. S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. In *Proceedings of 5th Annual Symposium on Operating Systems Design and Implementation (OSDI)*, December 2002.
19. S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin, and F. Yu. Data-centric storage in sensornets. In *Proceedings of SIGCOMM'02*, 2002.
20. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM '01*, August 2001.
21. S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for data-centric storage in sensornets. In *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, 2002.
22. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.
23. M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. A scalable and ontology-based p2p infrastructure for semantic web services. In *Proceedings of the 2nd International Workshop on Agents and Peer-to-Peer Computing*, 2002.
24. R.R. Schmidt and C. Shahabi. How to evaluate multiple range-sum queries progressively. In *21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02)*, June 2002.
25. R.R. Schmidt and C. Shahabi. Propolyne: A fast wavelet-based algorithm for progressive evaluation of polynomial range-sum queries (extended version). In *VIII. Conference on Extending Database Technology (EDBT'02)*, May 2002.
26. C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor information networking architecture and applications. *IEEE Personel Communication Magazine*, 8(4):52–59, August 2001.
27. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM '01*, August 2001.
28. J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, December 1969.
29. D.J. Watts, P.S. Dodds, and M.E.J. Newman. Identity and search in social networks. *Science*, 296:1302–1305, 2002.
30. Yong Yao and Johannes Gehrke. Query processing in sensor networks. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, January 2003.
31. B.Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UCB, April 2001.