A Software Tool for Large-Scale Sharing and Querying of Clinical Documents Modeled Using HL7 Version 3 Standard

Praveen R. Rao
Computer Sci. Electrical Engg.
Univ. of Missouri-Kansas City,
Kansas City, MO, USA
raopr@umkc.edu

Tivakar Komara Swami
Computer Sci. Electrical Engg.
Univ. of Missouri-Kansas City,
Kansas City, MO, USA
tk27f@mail.umkc.edu

Deepthi S. Rao
Veterans Affairs Medical Center & Univ. of Missouri-Kansas City, MO, USA
dsrkv5@mail.umkc.edu

Michael J. Barnes
Computer Sci. Electrical Engg.
Univ. of Missouri-Kansas City,
Kansas City, MO, USA
mjb5a6@mail.umkc.edu

Swati G. Thorve
Computer Sci. Electrical Engg.
Univ. of Missouri-Kansas City,
Kansas City, MO, USA
sgtr4c@mail.umkc.edu

Prasad Natoo
Computer Sci. Electrical Engg.
Univ. of Missouri-Kansas City,
Kansas City, MO, USA
pynzpf@mail.umkc.edu

ABSTRACT

We present a novel software tool called `CDN` (`C`ollaborative `D`ata `N`etwork) for large-scale sharing and querying of clinical documents modeled using HL7 v3 standard (*e.g.*, Clinical Document Architecture (CDA), Continuity of Care Document (CCD)). Similar to the caBIG initiative, `CDN` aims to foster innovations in cancer treatment and diagnosis through large-scale, sharing of clinical data. We focus on cancer because it is the second leading cause of deaths in the US. `CDN` is based on the synergistic combination of peer-to-peer technology and the extensible markup language XML and XQuery. Using `CDN`, a user can pose both structured queries and keyword queries on the HL7 v3 documents hosted by data providers. `CDN` is unique in its design – it supports *location oblivious queries* in a large-scale, network wherein a user does not explicitly provide the location of the data for a query. A location service in `CDN` discovers data of interest in the network at query time. `CDN` uses standard cryptographic techniques to provide security to data providers and protect the privacy of patients. Using `CDN`, a user can pose clinical queries pertaining to cancer containing aggregations and joins across data hosted by multiple data providers. `CDN` is implemented with open-source software for web application development and XML query processing. We report the evaluation of `CDN` in a distributed environment (LAN) using a real dataset of discharge summaries available from the i2b2 project.

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences—*Medical Information Systems*

General Terms

Design

1. INTRODUCTION

Today, it is well agreed upon that through effective use of Information Technology (IT), health care costs can be reduced and better quality care can be delivered to patients. The US government is spending billions of dollars to promote the adoption of electronic health records and to develop Health Information Exchanges (HIEs) [4]. HIEs aim to enable "the electronic movement of health-related information across organizations according to nationally recognized standards" [4]. They are considered to be the building blocks for Nationwide Health Information Network (NHIN) initiative [7] and are designed to achieving Institute of Medicine's (IOM) vision of a learning healthcare system [14]. Some of the established HIEs such as HealthBridge, CareSpark, Indiana Health Information Exchange, and MedVirginia serve up to few million patients and few thousand physicians, thereby, hosting large volumes of patient data [8].

Recently, "data sharing and collaboration" and "large scale management of health care data" have been identified as the key IT challenges to advance the nation's healthcare system [40]. This is because vast amounts of health-related information remain untapped due to the lack of suitable IT solutions. Personal health information resides in digital silos and healthcare systems do not easily share information with each other. However, by tearing down these silos, health-related information can be utilized by medical practitioners and researchers to provide efficient, quality, timely, and cost-effective care to patients.

The National Cancer Institute's caBIG is a nation-wide initiative, whose vision is to advance research on cancer and improve clinical outcomes for patients by connecting the members of the cancer community to share knowledge and data [20]. The caBIG community has more than 190 organizations [10]. Today, there are 124 participating institutes connected to caGrid – the underlying network infrastructure of caBIG. The community has shown great interest in sharing large amounts for biospecimen annotations, microarray data, cancer genome data (e.g., tissue samples), and so forth [3]. Such large-scale sharing of biomedical and clinical data is the first step towards collaborative e-science in the 21st century.

Achieving interoperability among applications processing clinical data has been a topic of interest for several years. Many advances have been made in developing standards for clinical data with regard to exchange/messaging, terminology, application, architecture, and so forth [26]. The standards from Health Level Seven International (HL7) have become popular for the exchange, integration, sharing and retrieval of electronic health information. HL7 standards are used by 90% of the hospitals in the US.[1] More recently, HL7 Version 3 standard was developed to enable *semantic interoperability* in healthcare data interchange [32]. (XML is used to encode the data.) The documents in HL7 v3 are derived from the Reference Information Model (RIM) and use terminologies such as SNOMED CT, LOINC, CPT, and so forth. Software tools are available for modeling data using HL7 v3 standards (*e.g.*, Model-Driven Health Tools [12], caAdapter [2], HL7 Tooling [11]).

We present a software tool called `CDN` (**C**ollaborative **D**ata **N**etwork) for large-scale sharing and querying of clinical data modeled in HL7 v3 standard. (We discussed our vision of `CDN` in IHI '2010 [33].) Of particular interest to us are the HL7 CDA (Clinical Document Architecture) and CCD (Continuity of Care Document) standards. `CDN` is ideal tool for data providers (*e.g.*, clinic, hospital, research lab) who wish to selectively enable data sharing and querying of HL7 v3 documents on a large-scale. While `CDN` is not restricted to a particular health condition, the GUI of `CDN` is designed for posing clinical queries related to cancer diagnosis and treatment. Cancer is the second most leading cause of deaths in the US. `CDN` *differs from the aim of HIEs in the sense that it is not designed for the electronic movement of health-related information across organizations.*

The salient features of `CDN` and key contributions of our work are summarized below.

• The design of `CDN` is based the synergistic combination of the peer-to-peer (P2P) technology and the widely adopted XML standard and the XQuery language. The striking feature of `CDN` is the notion of a *location oblivious query*, wherein the locations of relevant data in the network (for a query) are discovered during query processing. A user simply issues a single location oblivious query (that is mapped to an XQuery query) across multiple data sources in the network to perform aggregations and joins.

• Though `CDN` employs a P2P model, it still provides the benefits of a federated database model such as ownership of data and ability to implement local access control policies. The HL7 v3 clinical documents are always stored with the owner and are never exchanged or transferred across the network. While caBIG is designed to enable sharing of a wide range of biomedical and clinical data, `CDN` focuses only on sharing and querying HL7 v3 clinical documents.

• To process a location oblivious query, `CDN` has a novel location service called *psi*X for quickly identifying the locations of data relevant to the query [35]. This location service uses a novel distributed XML indexing technique that allows processing of XPath queries in a P2P environment.

• `CDN` employs a hybrid shipping approach [28] to process an XQuery query. Parts of the query are shipped across the network and parts of the query are processed by the data provider where the query was issued. Not only is this better than pure data shipping or pure query shipping w.r.t resource utilization and performance, but also for enabling high levels of security and privacy during query processing.

• `CDN` employs standard cryptographic techniques (*e.g.*, RSA public key cryptography) to provide security to data providers and authenticate members in the network during query processing.

• `CDN` is built using open-source software tools. We have conducted a performance evaluation of `CDN` in a LAN environment using a real dataset of deidentified discharge summaries available from the i2b2 project. The results are reported in Section 4.

The remainder of the paper is organized as follows. Section 2 provides the background and motivations. Section 3 describes the novel architecture of `CDN`, the query processing approach, and security schemes in `CDN`. Section 4 describes the implementation and evaluation of `CDN`. We conclude in Section 5 and provide a plan for future work.

2. BACKGROUND AND MOTIVATIONS

XML and Distributed XQuery.

The extensible markup language XML has become the de facto standard for information representation and interchange on the Internet. It is widely adopted in a variety of domains ranging from ecommerce to health informatics. XQuery is a popular query language for XML and is recommended by the W3C. It is a functional language that subsumes XPath – a query language for selecting qualifying nodes in an XML document. XQuery allows for the creation of new elements and attributes and the specification of their contents and relationships.

Distributed XQuery processing [37, 21, 44, 22, 5, 18, 6] has been studied in recent years. The underlying principle is to ship portions of a query to remote servers which then execute them. Locations of remote servers are specified in the query. These previous solutions were not designed for a P2P network, where the locations of relevant data of interest may not be known apriori. In contrast, `CDN` differs from previous techniques as it supports *location oblivious queries.*

Due to the popularity of P2P systems, several approaches were developed to find/locate relevant XML documents and their publishers in a P2P environment [23, 27, 15, 16, 35]. Of particular interest to us is the *psi*X system [35], which is used in `CDN` to process location oblivious queries.

Systems Based on a Federated Database Model.

A federated database model is typically used in data integration systems. A survey has shown that many state-level HIEs are based on a federated database model [1]. BIRN [25] is one of the early initiatives for large-scale sharing and collaboration of biomedical data (*e.g.*, neuroimaging data). SHRINE [43] is a federated querying tool for aggregating data from multiple sites. Currently, SHRINE does not support joins. FURTHeR [30] is a federated querying tool for heterogeneous data sources owned by multiple organizations. It emphasizes on OSGi-based development (`http://www.osgi.org`). NCI's caBIG [20] also uses a federated database model. It's underlying network infrastructure, called caGrid [39], is a model-driven, service oriented archi-

[1]http://www.itl.nist.gov/div897/docs/Message_Maker.html

```
FOR  $gene IN service
("http://cabio.osu.edu/GeneService.wsdl")/Gene,
$go IN service
("http://cabio.osu.edu/GeneOntologyService.wsdl")/GeneOntology,
$microarray IN service
("http://caarray.duke.edu/caArrayService.wsdl")/Microarray
LET $subject := $microarray/experiment/subject
WHERE
  $go/term='vacuole' AND $gene/goAcc=$go/acc AND
  $gene/gbAcc=$microarray/data/geneId AND
  count($microarray/data[geneId=$gene/$gbAcc]/condition)>50
RETURN
<subject>
  <subjectId>{ $subject/lsid }</subjectId>
  <species>{ $subject/species }</species>
  <microarrayData>
    { $microarray/data }
  </microarrayData>
</subject>
```

Figure 1: An XQuery query in caGrid

tecture, and the data services are accessed via grid services that expose data sources in well-documented, interoperable form. To the best of our knowledge, among the aforementioned federated systems, only caGrid [39] supports XML queries over native XML data sources.

Motivations.

The design of `CDN` is motivated by the following limitations of connecting heterogeneous XML data sources via a service-oriented architecture (*e.g.*, as in caGrid): (a) the inability to express complex queries effectively using XQuery, and (b) the lack of fine-grained selection of data sources.

Consider a query in caGrid to *find all the expression data where there are at least 50 conditions for genes found in the vacuole* shown in Figure 1. (This example is taken from Summary and Initial Recommendations draft available on the website of caBIG.) The query performs joins across data exposed by three data services `Gene`, `GeneOntology`, and `Microarray`. Suppose we want a query to access `Gene` and `Microarray` data from all possible data providers to perform the join. Then multiple queries should be posed – each one for a particular combination of `Gene` and `Microarray` data service – and therefore, will lead to poor scalability and performance when the number of data services grow.

`CDN` overcomes this limitation by constructing a *location oblivious query*. This is done by replacing `service`("http:// ... GeneService.wsdl") with the phrase `collection`("CDN") and replacing `service`("http:// ... MicroarrayService.wsdl") with the phrase `collection`("CDN") in the original query. Now the query specifies a join over multiple data sources without specifying the locations of `Gene` and `Microarray` documents distributed across a network of participating data providers. From a user's perspective, a single query is posed, rather than a potentially large number of queries with location information (or data services). From a system's perspective, fewer queries need to be processed.

Consider another query that performs aggregation over all or multiple data sources. Suppose the query has multiple selection predicates with a Boolean AND operation (*e.g.*, gender="female" AND smoker="yes" AND age>35). In a service-oriented environment like caGrid, the query will be shipped to each data source (assuming the data service name is known), but only a few may contain data that satisfies all the selection predicates. It is, therefore, effective to identify

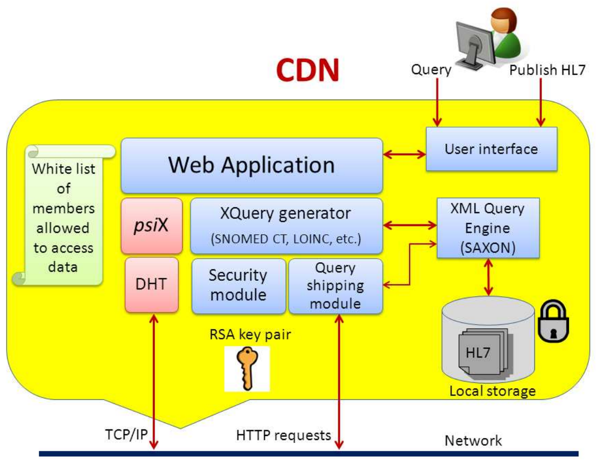


Figure 2: Key components of `CDN`

those data sources that contain matching data for all the selection predicates and to ship the query to only those data providers. `CDN` aims to achieve such *fine-grained selection of data sources* through the indexing power of *psi*X. The benefit is clear: the number of queries issued in the network are reduced and critical resources such as network bandwidth are saved.

3. THE ARCHITECTURE OF CDN

In this section, we present the novel architecture of `CDN`. The design of `CDN` is inspired by the success of P2P computing in the last decade – specifically, the concept of Distributed Hash Tables (*e.g.*, Chord [41], Pastry [38], CAN [36], Tapestry [45], Kademlia [31]), that has had major success. Today, production quality key-value stores such as Dynamo [17], Cassandra [29], and Voldemort [9] employ the principles of a DHT. `CDN` has two key design goals: (a) to enable scalable data sharing and querying of HL7 v3 clinical documents, and (b) to provide high level of security to the data providers and protect the privacy of patients for HIPAA compliance. In the following sections, we discuss how `CDN` achieves these goals. Occasionally, we use the terms "**data provider**", "**publisher**" and "**participant**" interchangeably to mean the same.

3.1 System Overview

Given a network of data providers, each data provider runs a copy of the `CDN` software much like an Internet user who installs and runs softwares such as Kazaa and Skype. The data providers are connected through a network such as the Internet or a Virtual Private Network (VPN). Each `CDN` software communicates with other `CDN`s in the network to process a user's request.

The key components of the `CDN` software are shown in Figure 2. The user interface accepts requests from a user (authorized by the data provider) to either publish HL7 v3 documents or pose queries. Documents that are made available for sharing with other data providers are stored in a local database, which may be encrypted for security reasons. At the heart of the `CDN` software, is a web application containing the XQuery generator, the Query Shipping module, the Security Module, and the location service called *psi*X built atop a DHT. CDN employs a restricted form of the *hybrid shipping approach* [28] for processing queries. The Query Shipping Module is responsible for shipping the subqueries to relevant data providers and storing the returned results. An open source XQuery processor executes the queries. These queries are either subqueries shipped from other data providers or queries generated by the XQuery generator to process the results obtained from other data providers (*e.g.*, join processing). The actual HL7 v3 documents are never transferred across the network. Each CDN maintains a RSA public/private key pair, which is used by the Security Module for authentication and secure communication during query processing. Each CDN maintains a white list of data providers/participants in the network that are allowed to access its local data. (One way is to maintain the public keys of allowed data providers in the white list.)

3.2 Publishing HL7 V3 Clinical Documents

HL7 v3 standard (*e.g.*, CDA R2 [19]) aims to provide incremental semantic interoperability and therefore, HL7 v3 documents can evolve over time. An adopter can start with minimal structure in the HL7 v3 documents and over time add more structure to the documents and code content using standard terminologies such as SNOMED CT and LOINC. Different XML schemas can be designed by the data providers to model their data as long as the schemas are derived from the Reference Information Model (RIM). For example, a data provider may decide to share only the "Past Medical History" and "Physical Examination" from deidentified discharge summaries of some patients.

CDN allows any valid HL7 v3 clinical document to be published by a data provider and therefore, a data provider is expected to do minimal standardization of the documents. (By using HL7 v3, we are in fact enforcing some standardization.) By "publishing a document", we mean that the data provider stores the document in its local database and the document becomes ready to be queried by other data providers. *How do other data providers become aware of this document?* The answer is through the location service called *psi*X [35, 34], which is based on a novel, distributed XML indexing technique for DHT-based P2P networks. It is important to note that a document owned by a data provider resides locally and is never exchanged or transferred through the network. The data provider has full ownership and control of its data and can implement local access control policies similar to a federated system.

In the interest of space, we provide a brief description of *psi*X – an Internet-scale location service for XML documents. (A reader is referred to previously published articles [35, 34] for complete details.) Using *psi*X, participants in the network can index XML documents in a distributed fashion; any participant can issue an XPath query and *psi*X will locate all participants/publishers that host XML documents containing a match for the XPath query. The *psi*X system indexes a *signature* of an XML document. The signature essentially captures the summary of the XML document and includes both the structural summary and the value/content summary [35]. The original document is not stored by *psi*X. This works well within CDN because we wish to protect the privacy of patient records and provide complete control to the owner of the data. Because *psi*X is built over a DHT, it inherits the scalability, fault-tolerance, and load balancing properties of the DHT. (The *psi*X system has been tested on more than 200 computing nodes in an Internet-scale environment [13].)

Algorithm 1: Publishing a HL7 v3 document

```
proc publishDocument(document $d$)
1: Store the document $d$ in the local database
2: Compute the signature $s$ for $d$ as described in psiX [35]
3: Construct the docid for $d$ by concatenating the
   hostname of the data provider, the local id of $d$, and
   the data provider's public key
4: Index $(s, docid)$ using $s$ as the key by invoking psiX
endproc
```

Algorithm 1 shows the sequence of steps involved in publishing a HL7 v3 document. First, the document is stored in the local database. Then the signature of the document is generated. The signature is indexed by invoking *psi*X and along with it the hostname of the data provider, the local id of the document, and the data provider's public key is stored. By knowing the hostname of a data provider and the local id of a document owned by that data provider, a participant in the network can ship a query to it for execution. The public key is necessary for secure communication during query processing. (The discussion of the security schemes employed by CDN is deferred until Section 3.4.)

3.3 Processing XQuery Queries

Next, we describe the steps involved in processing an XQuery query. We use the term "query initiator" to refer to the data provider where query is posed by a user. There are three well-known approaches to processing a distributed query [28], namely, *pure data shipping*, *pure query shipping*, and *hybrid shipping*. Neither pure data shipping nor pure query shipping are the best choices in all scenarios in a distributed setting and a hybrid approach has shown to perform better [28]. In the context of sharing clinical data, we have developed a restricted form of hybrid shipping approach to ensure that effective security and privacy policies can be implemented for HIPAA compliance. There are some limitations of pure data shipping and pure query shipping. If pure data shipping were employed, then an entire HL7 v3 document would have to be transferred across the network to the query initiator and the query initiator would have complete access to the document. If pure query shipping were employed, then the participating data providers would have to exchange results of shipped queries amongst each other (*e.g.*, in case of join operations). This may not be desirable. In the hybrid approach adopted by CDN, joins are always executed locally by the query initiator. The selection and projection operations in the query on a single document are always executed by the data provider owning the document. (Which parts of the data can be projected, will depend on what the data provider wishes to expose.) Aggregation and duplicate elimination can be done either by the query initiator or remotely by a data provider depending on the query.

Algorithm 2 shows the steps taken by the query initiator to process a location oblivious XQuery query. First, *maximal XPath expressions* are extracted from the query by examining the XPath expressions in the FOR, WHERE, and RETURN clauses (Line 1). *We define a maximal XPath expression as the longest XPath expression that should be matched in an XML document to generate correct results.*

Algorithm 2: Query processing at the query initiator

```
proc processXQuery(location oblivious XQuery query $q$)
1: Compute the maximal XPath expressions in $q$ by
   analyzing the XPath expressions in the FOR, WHERE,
   and RETURN clauses of $q$
2: foreach maximal XPath expression $p$ in $q$ do
3:     Send $p$ to psiX to get the (docid, publisher) pairs
       for all documents that contain a match for $p$
4:     foreach publisher returned by psiX do
5:         Create one XQuery query per matching
           document to do selections and projections and
           ship the entire list of queries to the publisher
6:     Merge the results from the publishers (after
       decryption) and store it in a single temporary XML
       document locally
7: Construct an XQuery query to operate on the
   temporary XML documents to perform operations such
   as joins, aggregation, and duplicate elimination
8: Return results to the user
endproc
```

Algorithm 3: Processing of a shipped query

```
proc processShippedQueries(list of queries $\overline{q}$)
1: Authenticate the query initiator by checking the white
   list of allowed data providers
2: if query initiator is authorized then
3:     Execute the queries $\overline{q}$ on the local HL7 database
4:     Encrypt the results and return the results
   else
5:     Do not execute $\overline{q}$ and reject further processing
endproc
```

For each maximal XPath expression, *psi*X is invoked to obtain the (docid, publisher) pairs for further processing (Line 4). Next, for each publisher identified by *psi*X, an XQuery query is created on one matching document owned by that publisher/data provider to do selections and projections. The entire list of such queries is sent all at once to that publisher (Line 5). The returned results are stored in temporary XML files (Line 6) and finally, local processing is done (Line 7).

Algorithm 3 shows the steps taken to process queries shipped to a data provider. First, the receiving data provider authenticates the query initiator using its white list containing public keys of authorized query initiators and public key cryptography. If the authentication succeeds, then the shipped queries are executed and the results are encrypted and returned to the query initiator. (The details of encryption and decryption steps during query processing is discussed in Section 3.4.)

3.3.1 Basic Aggregation Queries

We present a few examples of location oblivious XQuery queries with basic aggregation operations. These queries examine both coded content as well as textual content in the HL7 v3 documents. In the following discussions, we assume that the HL7 documents are modeled using Clinical Document Architecture (CDA) R2 over deidentified patient data. (In our evaluation, we used deidentified discharge summaries from the NLP research data sets available from the i2b2 project [42] and modeled them as CDA documents with sections, namely, History of Present Illness, Physical Examination, Past Medical History, Past Surgical History, Allergies, Hospital Course, Discharge Date, Discharge Diagnosis, and Discharge Disposition.)

Figure 3(a) shows a location oblivious XQuery query for a clinical query Q1 *to count the number of male patients who have had colon cancer.* Two maximal XPath expressions are extracted from the query and are shown in Figure 3(b). Based on these expressions, two different query templates are used to generate the queries shipped to the data providers containing matching documents for each of the maximal XPath expressions. These are shown in Figure 3(c). (As an optimization, we ship a list of queries at once to a data provider, one each on a matching document from that data provider.) Finally, counting and duplicate elimination is applied at the query initiator on the results of the shipped queries. This is shown in Figure 3(d).

A few more examples of clinical queries in `CDN` are shown in Figure 4. Because HL7 v3 is designed for incremental semantic interoperability, it is necessary to have queries process both coded content as well as textual content for better coverage of the data. For instance, in the query shown in Figure 4(a), the term "alopecia" is searched using the SNOMED CT code within `observation` as well as within the textual content under "Physical Examination."

`CDN` also supports keyword queries over HL7 v3 documents. Figure 5 shows a query that finds the number of patients who had past medical history of "anemia."

3.3.2 Aggregation Queries with Join Operations

Next, we present an aggregation query with a join operation that `CDN` can execute. Consider the query Q5 shown in Figure 6(a) to *find the number of patients who were given medications during hospital course that have caused an allergy in one or more patients.* The join operation is on the attribute `ApplicationNumber` of the medications coded under "Allergies" and "Hospital Course" in different discharge summaries. The two maximal XPath expressions for the query are shown in Figure 6(b). Two templates for the queries shipped by the query initiator to the data providers containing matching documents are shown in Figure 6(c). (Query template A is for the first maximal XPath expression and template B is for the second maximal XPath expression.) The partial results from the shipped queries (generated from query template A) are concatenated and stored locally in a temporary file say `A.xml`. The partial results from the shipped queries (generated from query template B) are concatenated and stored locally in a temporary file say `B.xml`. Finally, the query initiator (locally) performs the join operation on the attribute `ApplicationNumber` in `A.xml` and `B.xml`, followed by duplicate elimination and counting. This is shown by the query in Figure 6(d).

3.4 Security Module

`CDN` provides high level of security to data providers and protects the privacy of patient data to ensure HIPAA compliance. `CDN` uses standard cryptographic techniques to achieve this. Similar to a federated database model, each data provider has complete control of its data and exposes only those that it wishes to share (*e.g.*, portions of deidentified discharge summaries of certain patients). Each data provider running

```
Q1: How many male patients had colon cancer in the target population?

count (
 for $x in collection("CDN")/ClinicalDocument
 where $x/RecordTarget/PatientRole/Patient = "M" and
  ($x//observation/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"] or
   $x//procedure/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"])
 return $x/RecordTarget/PatientRole/ID
)
```

(a) XQuery query for Q1 over coded content in the HL7 v3 documents

```
/ClinicalDocument[RecordTarget/PatientRole/Patient = "M"][RecordTarget/PatientRole/ID]
             //observation/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"]

/ClinicalDocument[RecordTarget/PatientRole/Patient = "M"][RecordTarget/PatientRole/ID]
             //procedure/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"]
```

(b) Two maximal XPath expressions for Q1

```
(: ********** Query template A ********* :)
for $x in doc("...")/ClinicalDocument
   where $x/RecordTarget/PatientRole/Patient = "M" and
         $x//observation/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"]
  return <res> {$x/RecordTarget/PatientRole/ID} </res>

(: *********** Query template B ********* :)
for $x in doc("...")/ClinicalDocument
   where $x/RecordTarget/PatientRole/Patient = "M" and
         $x//procedure/code[@codeSystem="2.16.840.1.113883.6.69"][@code = "315058005"]
  return <res> {$x/RecordTarget/PatientRole/ID} </res>
```

(c) Templates of queries shipped to data providers with matching documents

```
count ( distinct-values ( for $x in doc("results.xml")//ID return $x ) )
```

(d) Counting and duplicate elimination performed locally

Figure 3: Step-by-step evaluation of a location oblivious XQuery query

```
Q2: How many patients developed alopecia as a side-effect of chemotherapy in the target population?

count (
 for $x in collection("CDN")/ClinicalDocument
 where $x//procedure/code[@code="150415003"][@codeSystem="2.16.840.1.113883.6.69"] and
  ($x//observation/code[@code="270504008"][@codeSystem="2.16.840.1.113883.6.69"] or
   $x//section[code/@code="29545-1"][code/@codeSystem="2.16.840.1.113883.6.1"]/text[contains(.,"alopecia")])
 return $x/RecordTarget/PatientRole/ID
)
```

(a)

```
Q3: How many cases of small cell lung cancer are noted among smoking females in the target population?

count (
 for $x in collection("CDN")/ClinicalDocument[RecordTarget/PatientRole/Patient = "F"]
 where ($x//procedure/code[@code="254632001"][@codeSystem="2.16.840.1.113883.6.69"] or
   $x//observation/code[@code="254632001"][@codeSystem="2.16.840.1.113883.6.69"]) and
   $x//section[code/@code="10164-2"][code/@codeSystem="2.16.840.1.113883.6.1"]/text[contains(.,"smoker")]
 return $x/RecordTarget/PatientRole/ID
)
```

(b)

Figure 4: Examples of queries that process both coded content as well as textual content

```
Q4: How many patients have had past medical history of ‘‘anemia’’?

(: *********** Keyword query *********** :)
count (
 for $x in collection("CDN")/ClinicalDocument
 where $x//section[code/@code="11348-0"][code/@codeSystem="2.16.840.1.113883.6.1"]/text[contains(.,"anemia")]
 return $x/RecordTarget/PatientRole/ID
)
```

Figure 5: Keyword query in CDN

```
Q5: Find the number of patients who were given medications during hospital course that have caused an
allergy in one or more patients.

count ( distinct-values (
  for $e in collection("CDN")/ClinicalDocument,
      $f in collection("CDN")//section[code/@code="45675-6"][code/@codeSystem="2.16.840.1.113883.6.1"]
  where
    $e/structuredBody/section[code/@code="8648-8"][code/@codeSystem="2.16.840.1.113883.6.1"]/
     manufacturedMaterial/@ApplicationNumber = $f/manufacturedMaterial/@ApplicationNumber
  return $e/RecordTarget/PatientRole/ID
))
```

(a) Join query in CDN

```
/ClinicalDocument[RecordTarget/PatientRole/ID]/structuredBody/section[code/@code="8648-8"]
                 [code/@codeSystem="2.16.840.1.113883.6.1"]/manufacturedMaterial/@ApplicationNumber


//section[code/@code="45675-6"][code/@codeSystem="2.16.840.1.113883.6.1"]/manufacturedMaterial/
          @ApplicationNumber
```

(b) Maximal XPath expressions extracted from the query

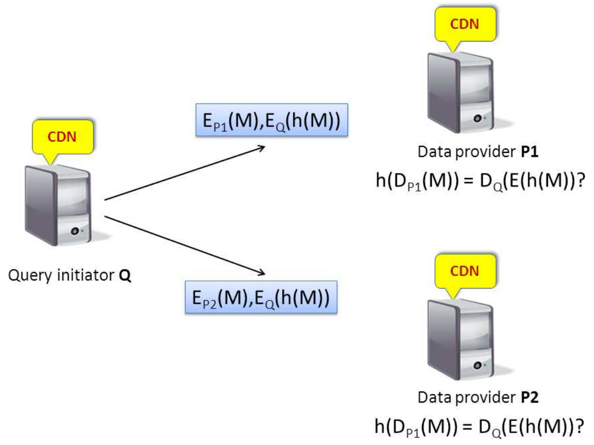
```
(: ********** Query template A ********* :)
for $e in doc("...")/ClinicalDocument
  where
    $e/structuredBody/section[code/@code="8648-8"][code/@codeSystem="2.16.840.1.113883.6.1"]/
     manufacturedMaterial[@ApplicationNumber]
  return
    <res>
      <arg1>{$e/structuredBody/section[code/@code="8648-8"][code/@codeSystem="2.16.840.1.113883.6.1"]/
              manufacturedMaterial}</arg1>
      <arg2>{$e/RecordTarget/PatientRole/ID}</arg2>
    </res>

(: *********** Query template B ********* :)
for $f in doc("...")//section[code/@code="45675-6"][code/@codeSystem="2.16.840.1.113883.6.1"]/
                      manufacturedMaterial[@ApplicationNumber]
  return
    <res> <arg1>{$f}</arg1> </res>
```

(c) Templates of queries shipped to publishers with matching documents to enable local joins

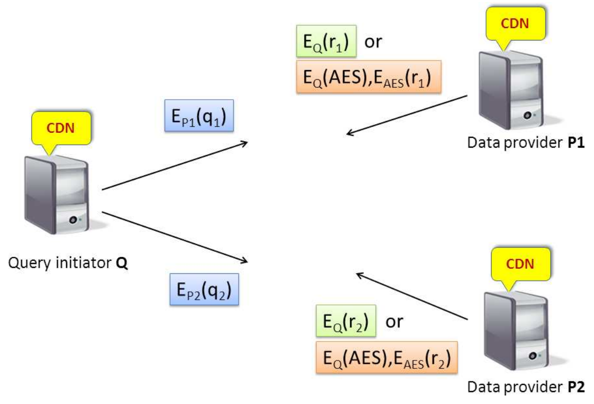
```
count( distinct-values(
   for $e in doc("A.xml")//res,
       $f in doc("B.xml")//res
   where $e/arg1/manufacturedMaterial/@ApplicationNumber = $f/arg1/manufacturedMaterial/@ApplicationNumber
   return $e/arg2/ID
))
```

(d) Join operation, duplicate elimination, and aggregation performed locally at the query initiator

Figure 6: Step-by-step evaluation of a location oblivious XQuery query with a join



(a) Authentication



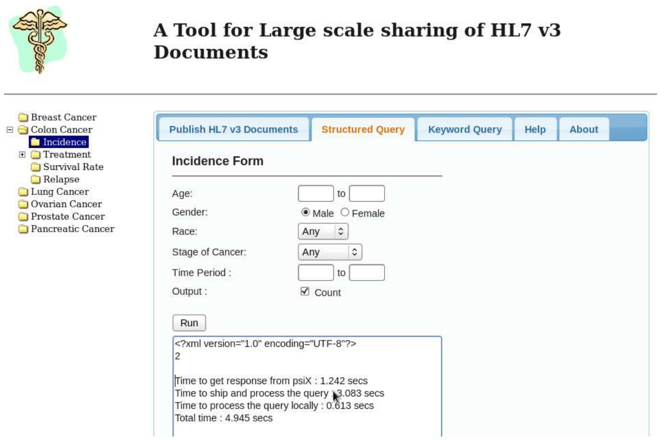
(b) Query processing

Figure 7: Security schemes in CDN

CDN generates a RSA public/private key pair. Each data provider also maintains a white list of data providers that can access its data (*e.g.*, the white list can contain public keys of those data providers).

When *psi*X is invoked to first identify relevant data providers and their documents, the public keys of the data providers are also returned. (Recall that along with the document's signature, the public key of the data provider is stored by *psi*X (Algorithm 1).) When the query initiator contacts a data provider that owns the matching documents, the data provider first verifies the identity of the query initiator. Essentially, the query initiator computes the hash of a message and encrypts it using its private key. Both the message, which can be encrypted using the public key of the data provider, and the encrypted hash are sent to the data provider. The data provider uses the public key of the query initiator (or checks its white list) and decrypts the encrypted hash and also generates a hash of the message (after decrypting it). When both match, the data provider considers the verification to be successful. If the verification fails, then the query initiator is refused further processing. We illustrate this process in Figure 7(a).

When the verification succeeds, the data provider executes the shipped queries on its local database. The query is encrypted by the query initiator (using the public key of the data provider) and the results from the data provider are encrypted (using the public key of the query initiator). This prevents malicious attacks and eavesdropping. When the



A Tool for Large scale sharing of HL7 v3 Documents

Incidence Form

Age:	to
Gender:	[✓ Male] Female
Race:	Any
Stage of Cancer:	Any
Time Period :	to
Output :	[✓ Count]

```
<?xml version="1.0" encoding="UTF-8"?>
2

Time to get response from psiX : 1.242 secs
Time to ship and process the query : 3.083 secs
Time to process the query locally : 0.613 secs
Total time : 4.945 secs
```

Figure 8: A screenshot of the user interface in CDN

size of the results returned by a data provider is large, the data providers generates an AES key and uses it to encrypt the results. The AES key itself is encrypted using the public key of the query initiator. We illustrate the entire process in Figure 7(b).

Note that never is an actual HL7 document exchanged or transferred through the network and it always resides with the owner. Moreover, through our hybrid shipping approach and the above security schemes, only an authorized query initiator and the data providers owning the relevant data for a query deal with unencrypted data.

3.5 User Interface

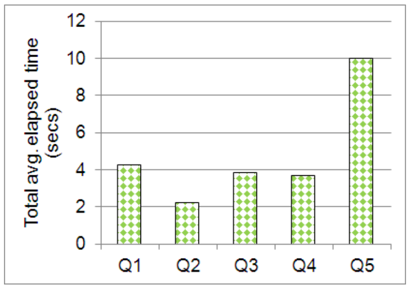
The user interface of CDN is designed to allow a clinician or researcher to easily publish a HL7 v3 document and pose structured queries and keyword queries related to cancer diagnosis and treatment. For structured queries, a browse hierarchy is provided to simplify the input process for structured queries. A form for posing incidence related queries on colon cancer is shown in Figure 8. (The user interface will be enhanced in the future to allow the entry of join queries.)

3.6 Joining and Leaving CDN

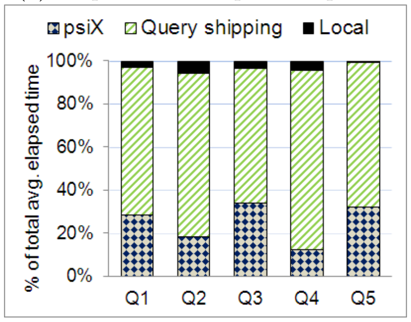
CDN can be setup to have any data provider join or leave the network of data providers at any time with little administration. Furthermore, if only authorized data providers should be permitted to join the network, then one data provider that usually acts as the bootstrap node, can maintain the public key of each authorized data provider and only allow a data provider to connect to the network after authentication using public key cryptography. In CDN, we expect low degree of churn unlike in Internet-scale P2P applications. We leave it up to the data provider to implement the desired authentication scheme for users in the institution that can interact with CDN using the GUI. For example, the data provider can rely on a single sign-on (SSO) approach used in many institutions.

4. IMPLEMENTATION AND EVALUATION

We implemented CDN in Java using Eclipse and the open-source JSP and Servlet Container called Apache Tomcat (version 6). The open-source XSLT and XQuery processor called SAXON [24] was the XML query engine in CDN. Available security libraries in Java were used for implementing the



(a) Elapsed time to process queries



(b) Breakup of time spent during query processing

Figure 9: Performance evaluation

security schemes in CDN. The *psi*X codebase was written in C++ and was implemented using the Chord DHT package. We tested and evaluated CDN in a local area network running five Pentium 4 machines with dual-core processors (3.4 GHz) running Fedora Linux. Each machine had 2 GB main memory and 80 GB disk drive.

4.1 Dataset of HL7 CDA Documents

We obtained deidentified discharge summaries from the NLP research datasets available from the i2b2 project [42]. From these discharge summaries, we created 325 HL7 CDA documents. These documents contained both coded content as well as textual content and had the following sections: History of Present Illness, Physical Examination, Past Medical History, Past Surgical History, Allergies, Hospital Course, Discharge Date, Discharge Diagnosis, and Discharge Disposition. The codes were drawn from LOINC, SNOMED CT, and FDA NDC (National Drug Code Directory). Clinical findings, observations, procedures, and manufactured materials in the discharge summaries were assigned appropriate codes. Human intervention was necessary due to the unstructured nature of textual content in the discharge summaries. For example, abbreviations were used in the discharge summaries such as B.C. for breast cancer and A. Fib. for atrial fibrillation.

4.2 Performance Evaluation

CDN was setup on 5 machines in a LAN and each machine represented a data provider. The data providers on four machines published 100, 100, 75, and 50 CDA documents, respectively. The data provider on the fifth machine issued the queries Q1, Q2, Q3, Q4, and Q5 shown in Figures 3, 4, 5, and 6. We measured the total elapsed time for each query once it was chosen to run through the GUI and report the average elapsed time over 5 runs.

Figure 9(a) shows the total (average) elapsed time per query. Query Q5 is a join query and also returned more results than other queries, and hence took more time to complete its execution. Figure 9(b) shows how much time was spent in the three phases of query processing, namely, (a) to locate relevant XML documents using *psi*X, (b) to ship queries to relevant data providers and receive the results, and (c) to perform local processing such as joins, duplicate elimination, and aggregation. We observed that the first phase where *psi*X was used, consumed under 35% of the total elapsed time and the final phase of local processing was less than 10% of the total time. The phase involving query shipping consumed most of the time. One reason is that we did not use multithreading to ship queries in parallel to relevant data providers that were identified by *psi*X. We plan to enhance our current implementation with multithreading. Also, we did not build any indexes on the CDA documents during local XQuery processing with SAXON. We plan to use tools with XML indexing support (*e.g.*, Oracle Berkeley DB XML) to facilitate faster local XQuery processing.

5. CONCLUSIONS AND FUTURE WORK

We presented a software tool called CDN for large-scale sharing and querying of HL7 v3 clinical documents. CDN is based on the synergistic combination of the P2P model of computing and XML and XQuery technologies. The key feature of CDN is the notion of location oblivious queries. CDN provides complete ownership of data to a data provider similar to a federated database model. For HIPAA compliance, CDN uses standard cryptographic techniques for providing security to data providers and protecting the privacy of patient records. CDN has a simple user interface for posing structured as well as keyword queries on both coded as well as textual content in HL7 documents. In the future, we plan to study the quality of results returned by CDN and run CDN in a cloud environment to test its scalability and performance. We plan to recruit users from hospitals to test CDN and obtain their feedback to improve its design. We would like to release CDN under the Open Health Tools Initiative (`http://www.openhealthtools.org/`). It may also be possible to release CDN as a service in a private cloud.

Acknowledgements.

Deidentified clinical records used in this research were provided by the i2b2 National Center for Biomedical Computing funded by U54LM008748 and were originally prepared for the “Shared Tasks for Challenges in NLP for Clinical Data” organized by Dr. Ozlem Uzuner, i2b2 and SUNY. We thank the anonymous reviewers for their insightful comments.

Praveen R. Rao would like to acknowledge the funding support from National Science Foundation (IIS-1115871), University of Missouri Research Board, and IBM Smarter Planet Faculty Innovation Award.

6. REFERENCES

[1] http://www.hoise.com/vmw/07/articles/vmw/LV-VM-01-07-29.html.

[2] caAdapter. https://cabig.nci.nih.gov/tools/caAdapter/.

[3] caBIG Annual Report 2009. http://cabig.cancer.gov/resources/reports/2009ar/.

[4] Defining Health Information Exchange. http://www.himss.org/content/files/2009DefiningHIE.pdf.

[5] DXQP - Distributed XQuery Processor. http://sig.biostr.washington.edu/projects/dxqp/.

[6] Galax: An Implementation of XQuery. http://galax.sourceforge.net/.

[7] HIMSS Health Information Exchange. http://www.himss.org/asp/topics_hie.asp.

[8] Overview of Health Information Exchange (HIE). http://www.himss.org/content/files/RHIO/RHIO_HIE_GeneralPresentation.pdf.

[9] Project Voldemort. http://project-voldemort.com/.

[10] The caBIG Pilot Phase Report Executive Summary. https://cabig.nci.nih.gov/overview/pilotreport_ExSum.

[11] The HL7 Tooling Project. https://www.projects.openhealthtools.org/sf/projects/hl7tooling/.

[12] The Model-Driven Health Tools Project. https://www.projects.openhealthtools.org/sf/projects/mdht/.

[13] The *psi*X Project. http://vortex.sce.umkc.edu/psix.

[14] Crossing the Quality Chasm: A New Health System for the 21st Century. *The National Academies Press, Washington D.C.*, 2005.

[15] S. Abiteboul, I. Manolescu, N. Polyzotis, N. Preda, and C. Sun. XML Processing in DHT Networks. In *Proc. of the 24th IEEE ICDE*, Cancun, Apr. 2008.

[16] E. Curtmola, A. Deutsch, D. Logothetis, K. K. Ramakrishnan, D. Srivastava, and K. Yocum. XTreeNet: democratic community search. In *Proc. of the 34st VLDB Conference*, pages 1448–1451, Auckland, 2008.

[17] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *Proc. of 21st Symposium on Operating Systems Principles*, pages 205–220, Stevenson, WA, 2007.

[18] L. T. Detwiler, D. Suciu, J. D. Franklin, E. B. Moore, A. V. Poliakov, E. S. Lee, D. P. Corina, G. A. Ojemann, and J. F. Brinkley. Distributed XQuery-based integration and visualization of multimodality brain mapping data. *Frontiers in Neuroinformatics*, 3(0), 2009.

[19] R. H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. M. Behlen, P. V. Biron, and A. Shabo Shvo. HL7 Clinical Document Architecture, Release 2. *Journal of the American Medical Informatics Association*, 13(1):30–39, 2006.

[20] D. Fenstermacher, C. Street, T. McSherry, V. Nayak, C. Overby, and M. Feldman. The Cancer Biomedical Informatics Grid (caBIG). In *Proceedings of IEEE Engineering in Medicine and Biology Society*, pages 743–746, Shanghai, China, 2005.

[21] M. Fernandez, T. Jim, K. Morton, N. Onose, and J. Simeon. DXQ: A Distributed XQuery Scripting Language. In *4th International Workshop on XQuery Implementation Experience and Perspectives*, 2007.

[22] M. F. Fernàndez, T. Jim, K. Morton, N. Onose, and J. Siméon. Highly Distributed XQuery with DXQ. In *Proc. of SIGMOD 2007*, pages 1159–1161, 2007.

[23] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Locating Data Sources in Large Distributed Systems. In *Proc. of the 29th VLDB Conference*, Berlin, 2003.

[24] M. Kay. SAXON: The XSLT and XQuery Processor. Available from http://saxon.sourceforge.net/.

[25] D. B. Keator, D. Wei, S. Gadde, H. J. Bockholt, J. S. Grethe, D. Marcus, N. Aucoin, and I. B. Ozyurt. Derived data storage and exchange workflow for large-scale neuroimaging analyses on the BIRN grid. *Frontiers in Neuroinformatics*, 3(0), 2009.

[26] K. Kim. Clinical Data Standards in Health Care: Five Case Studies. http://www.chcf.org/publications/2005/07/clinical-data-standards-in-health-care-five-case-studies.

[27] G. Koloniari and E. Pitoura. Peer-to-Peer Management of XML Data: Issues and Research Challenges. *SIGMOD Record*, 34(2):6–17, June 2005.

[28] D. Kossmann. The State of the Art in Distributed Query Processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.

[29] A. Lakshman and P. Malik. Cassandra: A Structured Storage System on a P2P network. In *Proc. of the 2008 ACM-SIGMOD Conference*, Vancouver, Canada, 2008.

[30] O. E. Livne, N. D. Schultz, and S. P. Narus. Federated Querying Architecture For Clinical And Translational Health IT. In *Proc. of the 1st ACM International Health Informatics Symposium*, pages 250–256, Arlington, Virginia, USA, 2010.

[31] P. Maymounkov and D. Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Proceedings of First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002.

[32] C. Mead. Data Interchange Standards In Healthcare IT – Computable Semantic Interoperability: Now Possible But Still Difficult, Do We Really Need A Better Mousetrap? *Journal of Healthcare Information Management*, 20(1):71–8, 2006.

[33] P. Rao, S. Edlavitch, J. Hackman, T. Hickman, D. McNair, and D. Rao. Towards large-scale sharing of electronic health records of cancer patients. In *Proc. of 1st ACM International Health Informatics Symposium*, pages 545–549, Arlington, VA, 2010.

[34] P. Rao and B. Moon. An Internet-Scale Service for Publishing and Locating XML Documents. In *Proc. of the 25th IEEE Intl. Conference on Data Engineering*, pages 1459–1462, Shanghai, China, March 2009.

[35] P. Rao and B. Moon. Locating XML Documents in a Peer-to-Peer Network using Distributed Hash Tables. *IEEE Transactions on Knowledge and Data Engineering*, 21(12):1737–1752, December 2009.

[36] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. of the 2001 ACM-SIGCOMM Conference*, pages 161–172, 2001.

[37] C. Re, J. Brinkley, K. Hinshaw, and D. Suciu. Distributed XQuery. In *Proc. of the Workshop on Information Integration on the Web*, pages 116–121, 2004.

[38] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of the IFIP/ACM Intl. Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, Nov. 2001.

[39] J. Saltz, S. Oster, S. Hastings, S. Langella, T. Kurc, W. Sanchez, M. Kher, A. Manisundaram, K. Shanbhag, and P. Covitz. caGrid: Design and Implementation of the Core Architecture of the Cancer Biomedical Informatics Grid . *Bioinformatics*, 22(15):1910–1916, 2006.

[40] W. W. Stead and H. S. Lin. Computational Technology for Effective Health Care: Immediate Steps and Strategic Directions. *The National Academies Press, Washington D.C.*, 2009.

[41] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of the 2001 ACM-SIGCOMM Conference*, pages 149–160, San Diego, 2001.

[42] Özlem. Uzuner, I. Goldstein, Y. Luo, and I. Kohane. Identifying patient smoking status from medical discharge records. *Journal of the American Medical Informatics Association*, 15(1):14–24, 2008.

[43] G. M. Weber, S. N. Murphy, A. J. McMurry, D. MacFadden, D. J. Nigrin, S. Churchill, and I. S. Kohane. The Shared Health Research Information Network (SHRINE): A Prototype Federated Query Tool for Clinical Data Repositories. *Journal of the American Medical Informatics Association*, 16(5):624–630, Sept. 2009.

[44] Y. Zhang and P. A. Boncz. XRPC: Interoperable and Efficient Distributed XQuery. In *Proc of Very Large Data Bases*, Vienna, Austria, September 2007.

[45] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41 – 53, Jan. 2004.