Design of a Distributed Digital Library for HealthCare (DLHC)

Chad M.S. Steel, *Member, IEEE*

***Abstract*—The Digital Library for HealthCare (DLHC) implements a hybrid peer-to-peer network for dissemination, storage, and querying of electronic patient records. By making functions available in a service oriented architecture using web services, the DLHC has a low cost of implementation and a high degree of extensibility. The DLHC has the ability to support all types and sizes of records currently available, and has a growth channel that will support future data needs alongside of historical record access.**

***Index Terms*—Digital Library, Healthcare, Electronic Patient Record, Peer to Peer, Service Oriented Architecture.**

I. Introduction

"The future for the application of computers in medicine is bright. With health care now considered a right rather than a privilege, the demands on physicians offer a unique opportunity to use the computer as a 'physician assistant.' Large computer files on patients will be kept, and decisions made from these files will assist the physician and health care provider... A computerized health care system is the answer to these new demands." *IEEE Computer Magazine, January 1975*[1].

Health care information management is a growing problem with direct impact on patient care. There is significant current interest in making patient medical records broadly available to two classes of individual – the patient and the provider[2]. Much of the research interest has focused on the role of the patient in managing their own healthcare information[3-5] and on the implementation of small-scale systems with a centralized storage component [6, 7].

The non-distributed implementations put forth to-date address the concerns of either the patient or the provider, but are not designed for access by both classes' stakeholders who have very different needs. Additionally, current implementations assume a homogenous, structured dataset with centralized records management by a single source or small number sources. In reality, there are numerous competing standards for health information and many records are in proprietary formats[8].

The DLHC provides a conceptual design which addresses the shortcomings of current implementations while providing a solution which maps better to distributed nature of healthcare information and the difficulties faced when working with an IT-phobic audience[9].

To make the DLHC future-resistant and ensure interoperability with the largest number of datasets, a Service Oriented Architecture (SOA) design is used. A set of primitive operations is defined for DLHC rather than requiring a single interface and/or platform implementation. The use of a unifying SOA allows individual requestors the ability to define their own fit-for-purpose client interfaces as well as the ability to retrofit the disparate, legacy sources of patient records. A peer-to-peer (P2P) design is implemented for communication and data transfer that allows for both rapid, distributed query fulfillment as well as redundant, resilient data storage.

II. Prior Art

There are multiple relevant peer-to-peer network designs that DLHC builds upon, and significant work has been done in making information available in a SOA. Additionally, there have been several theoretical implementations of healthcare record systems and, large scale implementations of non-healthcare libraries provide a basis for many of the architecture decisions for the DLHC. Key areas of prior art are detailed below.

A. Peer-to-Peer

Peer-to-Peer networks have been moved from simple sharing environments like Gnutella[10] to rich content-sharing and replication systems. In the medical space,

Chad M.S. Steel is a PhD student in the Computer Science Department, Virginia Polytechnic Institute and State University, Falls Church, VA 22043. This paper is being submitted as a requirement of the qualifying examination for 2007 in Information Retrieval.

Huang et al describe the Picture Archiving and Communication System, PACS[11]. Meant to handle tens of terabytes of medical data, PACS was designed primarily as a way to store records off-site rather than a collaborative tool. Hung-Chang et al describe a more general approach to content storage in peer-to-peer networks[12].

Vagelis et al describe a Communications Virtual Machine meant to mediate different health data, but without a focus on data redundancy[13]. For general data reliability, Reiche et al describe approaches to data reliability in general peer-to-peer networks[14], and Judge and Ammar document strategies to protect content[15].

B. Service Oriented Architectures

SOAs are a good fit for health information, given the limited number and repeatability of tasks a typical patient looks to perform in a standardized way across providers [16]. This same concept of operations was used by Roantree et al in a smaller-scale Common Object Request Broker Architecture (CORBA) implementation over a Health Level 7 (HL7) structured records system[17]. Beyer et al place a system on top of HL7 and other structured data, but take a use-case approach to design and add the idea of a consolidated Master Patient Index (MPI)[18] similar to the patient index used in DLHC. Both Turner et al and Omar et al use web services as the foundation for health information systems in an SOA architecture – Turner to make available health data as a service and Omar to build a network of health sensors[19-21].

Earlier work with distributed object technology in the area of consolidating health records was performed by Anderson Consulting and presented as part of a workshop on objects in healthcare[22]. Amendalia et al built a more robust system that uses the concept of both simple and complex queries as well as distributed query processing in an SOA, similar to DLHC for just mammogram data[23]. Possibly the closest in concept to DLHC is Integrated Healthcare for the Enterprise (IHE), noted above as providing an interoperability standard. Its IHE's intent to permit interoperability through an SOA based on web services[24]. DLHC uses a similar model with the additional benefit of leveraging peer-to-peer communications.

C. Theoretical Healthcare Management Systems

A theoretical model for healthcare records, TeleMed, was proposed by Forslund et al which has the unique capability of integrating the results of a patient record search from multiple sources[25]. TeleMed is the closest design to DLHC in that it is a distributed model and uses a public key authentication mechanism to ensure privacy. It also has a proposed data mining function for searching similar images that can be applied to rich media as in DLHC. Takeda proposes another theoretical system which incorporates typical hospital management functions (billing, scheduling, etc.) alongside a distributed object implementation of record sharing. The concept of extending the capabilities beyond individual patient access to institution and public health access is also used in DLHC[26]. Abiteboul et al further look at a peer-to-peer system which addresses current privacy concerns using XML[27]. Finally, Bliykh et al propose a grid-based approach to national health information management using a SOA approach, but, like others noted above, rely on records being in a common format[28].

D. Digital Libraries

Digital libraries address the same issues of rich content indexing as healthcare records[29], and recent concerns about digital rights management (DRM)[30] have corollaries to healthcare records protection. LOCKSS provides a raw replication framework, with a focus on securing data for extended period of time, building on other data replication work [31-33]. The replication strategy takes into account authorization for content access, but does not address multiple source aggregation. Additionally, replication is not done in a swarm fashion.

Recognizing the benefits of a hybrid peer-to-peer network, Lu et al use a region-based document retrieval system which serves as an inspiration for the hierarchical superstructure in DLHC[34].

III. DESIGN PRINCIPLES

A. Performance

The performance of the DLHC should be suitable for large scale implementation. For discussion purposes here and below, the United States healthcare system is used, but the ultimate design should scale to global levels. The performance principles are set forth as follows.

1) Availability

Any medical record held within DLHC should be available for query and access from any requestor at any

time.

2) Query Time

Any query request should be fulfilled within a sublinear search time plus the time to transfer the record contents. This includes system-wide queries, such as an epidemiologist looking for aggregate disease trends in a particular state, as well as directed queries, like an individual searching for their records at a specific provider. Querying should be optimized for both keyword and geographic searches.

3) Scalability

The DLHC should be sufficiently robust as to accommodate dozens of records for each member of the global population. Query time and performance must scale at a sublinear rate based on a linear increase in the number of records, number of providers, and the number of queries generated.

B. Interoperability

Because of the diverse nature of legacy health information systems presently used and the inability to predict future trends, the DLHC must be general enough to support interoperability with past, present, and future systems. To ensure the maximum interoperability, the DLHC defines a high level protocol, a set of metadata, and a group of operations necessary to achieve the design goals. The key interoperability design principles employed are noted below.

1) Legacy Support

The DLHC must support legacy platforms either natively or through the implementation of appropriate middleware.

2) Technology Independence

The design decisions should not constrain the implementation to a single platform or class of platforms. Additionally, the abstractions should not require any specific lower-layer network mechanism. Support for heterogeneous platforms to increase adoptions and ensure class-attack protection should be implemented[35].

Schema Independence

The architecture must not require a single schema and/or data storage format for records. Support for both structured and unstructured data must be permitted to prevent existing data stores from needing to either duplicate data or transform other applications such as billing systems to support a different format.

C. Security

The regulatory nature of health information globally requires a high degree of security and privacy to even be permissible to implement. In the United States, the Health Insurance Portability and Accessibility Act[36] strictly regulates the controls over data that must be in place on systems from covered entities, which include the providers connecting to DLHC. In other countries, laws like European Data Protection Act require both patient control over their records (regardless of location) as well as restrictions on sharing of personally identifiable information across national borders, further complicating implementation[37]. The system design must take into account these restrictions as well as future restrictions of a legal nature. The overall security design principles are detailed below.

1) Confidentiality

DLHC must maintain record confidentiality. Unlike many other P2P systems, DLHC requires user intervention and authorization for records release. This is further complicated by the requirement to transfer control over records release in cases of emergency and individual incapacity. The confidentiality requirement holds for aggregate data as well – the knowledge that a search hit for an individual name even exists at an abortion clinic, for example, would provide inappropriate insight into a record even without revealing the contents of that record. Maintaining the confidentiality of the data while at rest and in transit (to prevent eavesdropping or platform-level disclosure) is a platform-specific implementation decision outside the scope of the system architecture.

2) Integrity

Record contents must be maintained without alteration, either accidental or intentional. The patient and provider responsible for creation must be the ultimate arbiters of record contents. Completeness of the record must be guaranteed as well – the exclusion of key information such as a drug allergy could result in a loss of life.

3) Revocability

An individual under European Union law has the right to revoke access to their information from any provider at any time. This requires a temporal nature be included in patient authorizations (to avoid a central revocation list). This includes the need for a patient to request deletion of their record from a specific provider as well as delete information from the entire DLHC.

4) Redundancy

Sufficient redundancy must be present in the DLHC to prevent the loss of data when a single provider and/or cluster of providers are lost. This must extend to the loss of a location through a location failure or loss of

network connectivity.

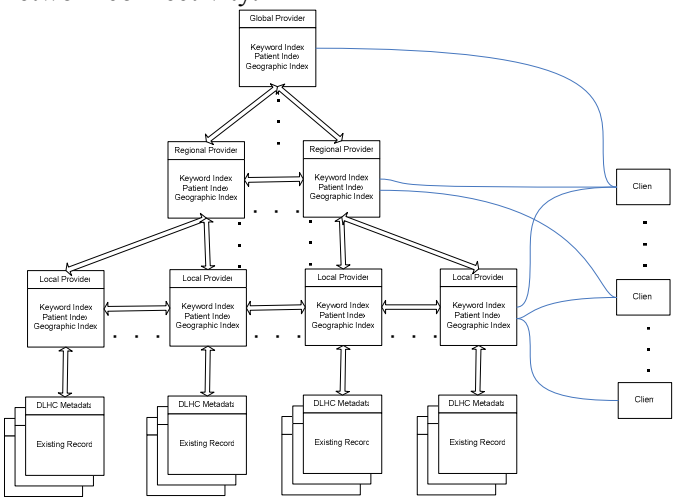

Fig 1. DLHC Conceptual Architecture.

IV. Architecture

The DLHC makes use of a P2P architecture with a superimposed geographic hierarchy. The system consists of two node types – providers and consumers. Providers are persistent nodes which contain healthcare records for individuals, whereas consumers are clients which make queries on providers. In many cases, a provider is also a client – a doctor's office may be a provider that makes available its patient's records, while simultaneously acting as a client that queries and obtains the records for new patients. Fig. 1 shows the conceptual architecture, with clients shown querying multiple providers. The providers are shown in a hierarchical fashion, but this is for conceptual purposes only – each provider has the same implementation and maintains patient information and is only hierarchically arranged for indexing purposes.

Data on patients stored on provider systems in indexed in three separate ways. First, the provider hierarchy itself is indexed by geographic region. Second, patient records are indexed by patient identifier. Third, key terms from the individual records are extracted and held in a third, inverted index.

Though the use of separate indices may be suboptimal for performance compared to hybrid structures[38], they are more appropriate for the DLHC implementation for three reasons. First, the indices are segmented differently – the geographic index serves double duty as a node list, while the patient indices and keyword indices are split and localized. Second, the use of the geographic index is a local search whereas the other searches are distributed. Third, by using the geographic index first to identify an appropriate keyword or patient-record subindex location, hybrid structure performance is achieved without the added complexity of a more complicated data structure.

The patient data itself is stored in a hybrid structure consisting of raw data and superimposed metadata. There are two main choices for existing data – allow it to remain in its existing form or transform it into a defined schema[39]. Choosing to allow the raw data to remain in its existing form allows for both easy integration with existing records systems and support for future systems. Choosing to migrate to a defined schema enhances the ability to query the information to approach that of a data retrieval solution. In order to speed adoption while still allowing for some metadata being stored, the DLHC defines a metaschema which coexists with existing structures. The metaschema defines only three mandatory fields – the patient identifier, the patient public key and the record identifier – with other fields added as optional enhancements depending on the record contents. The remainder of the record is full-text indexed in its current format without requiring semantic information. For already-structured data, the metaschema is extensible enough to include common patient record fields which can be populated but are not mandatory. Additionally, fields for multimedia content such as the transcript from a video, thumbnails of x-rays, and format information for MRI scan results are supported to allow for rich client interfaces and rapid browsing[40].

A. Geographic Index

The individual providers are stored in a hierarchical index, called the geographic index, maintained in a basic R-Tree data structure for quick searching. The R-Tree is used to provide fast querying for areas with both evenly distributed providers (the East Coast of the United States) as well as sparse areas (Eastern Siberia)[41]. Though not optimized for all situations, the real world data is not expected to approach the worst case scenarios, thereby warranting a more complex structure [42]. By providing a geographic hierarchy, query localization can be performed. Examples of geographical queries may include a patient attempting to identify online providers in their local area or a county health department compiling disease statistics. Providing a geographic index structure additionally permits the localization of searches for patient information where a priori information is known about the record locations.

Each geographic index is replicated in its entirety to

every provider in the DLHC. This is done so that every peer does not need to respond to every query on the network – a geographic search for encephalitis cases in Georgia does not need to engage peers in California.

ProviderID	Description	MBR Min Lat/Long	MBR Max Lat/Long	Last Update	Last Known IP	Parent	Child 1..8
4 Bytes	64 Bytes	20 Bits	20 Bits	4 Bytes	4 Bytes	32 Bytes	256 Bytes

Fig 2. Geographic index entry details.

There are approximately 360,000 provider entities in the United States at the present time (hospitals, clinics, doctors offices, podiatrists, etc.) and the number is decreasing with industry consolidation[43] and growth in provider size (there are fewer individual doctors – most work in practices). Assuming each provider has a description of 64 single byte characters[1] plus an IP address (4 bytes) and an associated minimum bounding rectangle (MBR) (5 bytes to store latitude and longitude to the second for the bounds), a 32 bit unique identifier, and a 4 byte date entry for the last update as shown in Figure 2, each entry can be stored in 81 bytes uncompressed. Including 288 bytes for child and parent node pointers, we have a size of 369 bytes per entry, giving a total size of 127MB for the entire directory. Assuming a 50% compression ratio (better compression is possible with fit-for-purpose algorithms, but DLHC implements standard Lempel-Ziv for compatibility[44]), the total size of the directory would be 63.5MB, not an unreasonable size for a one time transfer plus updates at current bandwidths.

Each of the providers is responsible for defining its own scope in the MBR. A local doctor's office would have a scope consisting of a few miles, whereas a regional hospital might have a scope that covers dozens of miles. At the higher levels, a state health association might cover hundreds of miles and a national organization the entire US. The MBR should be defined so as to cover the majority of the patient base for an organization – a few outlier patients should not increase the scope of the MBR. In addition to their own records, regional providers maintain metaindices for patient records and keywords as described in those index sections below.

Updates to the geographic database can be performed by a client or peer request (pull.) Push updates are not permitted to avoid overloading of the peers by a malicious and/or misconfigured provider. The details of the update operations are described below.

For optimized searching of provider names, a client may further index the 360,000 provider descriptions in a simple inverted index, but this is not an interaction of the system and is purely client use dependent.

[1] To support double-byte and quad-byte languages, as well as more detailed descriptions, additional description information can be stored in the metadata associated with the node itself.

B. Patient Index

The patient index is a simple, sorted list mapping patients' identifiers to providers which hold information about that patient. Each provider must maintain an index of all patients on their local system, as well as peer providers under the same regional provider. For regional providers, an index of all patients of the provider and its peers as well as an index of patient records held by sub-providers must be maintained. Due to the expected sparseness of consolidated regional metaindices, a reverse mapping of the patients at a particular subprovider is not maintained.

There are two different types of index entries for patients – internal and external. Internal entries correspond to records maintained by the provider itself (or replicated to the provider from peers). The internal entries indicate to the provider that it must search its local system – how the local search is conducted is dependent on the underlying software and is not relevant to the DLHC architecture.

External entries indicate patient records exist within a sub-region of the current index. In the case of state and national-level providers, an exhaustive search of all sub-regions may be required to find the one that actually contains the patient record. While this may be less efficient overall, it distributes the search request between multiple peers, it only requires one peer per sub-region to perform the search, and it allows the patient record to be a smaller size as it does not require the storage of specific sub-provider identifier.

PatientID	Record Location	Last Update	Secure Identifier
8 Bytes	1 Byte	4 Bytes	64 Bytes

Fig 3. Patient index record structure.

Unlike the geographic index, the patient index is expected to grow linearly as time progresses to coincide the growth in population[45]. As such, the structures must be searchable in a reasonably fast manner for both current and historical data. At this point, no provisions for data expiration (for deceased patients) are planned, but such provisions could be easily implemented using the operations provided in the future as needed.

The patient index is stored as a simple sorted list of patient identifier records, as shown in Figure 3. Since the searches on the list are dependent on local resources, the individual providers can determine how to best

organize local lists, but a B-tree would be a reasonable structure[46]. Each internal identifier consists of a unique patient ID (64 bits – enough to handle the growth in patients for the next several millennia), a last updated date, a 512-bit public key for the patient, and a record location indicator.

The use of the public key in authenticating queries is further detailed below. In the event future implementations require a larger key (due to advances in cryptanalysis), the patient identifier record can be expanded with new record fields without a requirement for major architectural revisions.

With a per-patient field size of 77 bytes, the size of the patient list can grow to be fairly large. As an example, a national-level provider that maintained the list of all patients in the United States would need approximately 23 Gigabytes of storage to keep the list. This allows for efficient identification of the existence of a record within a given scope while providing efficient local searching of much smaller provider lists for regional and local entities.

C. *Keyword Index.*

The keyword index maintains a list of unique keywords in an inverted index structure. Each provider maintains an index of the keywords present in all the patient records on its system. Because the patient records for each branch of the geographical index are replicated to other peers at the same tree level on that branch, each of these peers will also have the same index. Because of this, there is no need for an index synchronization routine – the indices can be updated locally after the patient records have been synchronized.

As with the patient index, all regional providers must maintain a keyword index for all peers underneath them as well as their own local index. Unlike the leaf indices, these indices will contain only the words that are present on the lower-geographical scope peers, not pointers to the specific documents or even providers (for large-scope indexes) that contain the words.

Based on the above structure, the index for a given provider's internal records would be expected to grow at linear rate and be approximately 15% of the total record size after compression using standard coding techniques[47]. Given an average patient record size of 17KB[48] per visit, and a 2.67 visits per patient per year[49], there would be a total of 13.6 terabytes of text health data generated annually in the United States. An average data size of 38GB per provider per year would be generated based on the above numbers[2]. Given the 15% above, the average index would grow linearly at a rate of 5.7GB per year.

Since the metaindices held on regional providers do not contain pointers to data (other than their own internal patient data), they are not expected to grow at the same rate as the individual provider indices. Because of this, the metaindices would be expected to grow at a sublinear rate based on Zipfs Law[50]. Given the search efficiency of a B-tree as an example index structure, this would mean a log(log(N)) annual increase in search time, much slower than the expected growth in processing speed and memory availability[51], indicating search times would improve faster than data growth would inhibit their speed.

For local provider keyword indices, a mapping of keywords to individual patient records is maintained as another simple inverted index structure. These are maintained as an inverted index with a pointer into a list of unique record identifiers which identify the actual patient records.

To reduce the size of the index (and more importantly, to limit the time spent on fruitless querying for this application) key stopwords are removed. Because the structured nature of patient records is such that certain words will likely appear in every record (e.g. name, address, sex, etc.), DLHC uses a custom stopword list as well[52].

D. *Patient Records*

With competing standards for the implementation of Healthcare records, frequently referred to as Electronic Health Records (EHRs), the DLHC must retain broad compatibility with legacy, emerging, and future implementations. Existing standards include Health Level 7's Clinical Document Architecture, the OpenEHR initiative, the Integrated the Healthcare Enterprise's Cross-Enterprise Document Sharing (IHE-XDS), and newer systems built upon Medical Markup Language[8]. Because of the competing standards, the DLHC is built to sit on top of existing records systems and allows individual providers the ability to leverage the semantic structure of these libraries while still retaining compatibility with unstructured and proprietary records systems.

DLHC defines a metaschema, a schema that sits on top of existing schemas. The metaschema has three required components, a 64-bit patient identifier, a 512-

[2] No studies indicating the distribution of patients per facility on the above provider statistics are available, so a simple mean calculation is used.

bit patient public key and a second 64-bit record identifier.

The 64-bit patient identifier is unique to the entire DLHC network. When a patient is added to the system, a random 64-bit identifier is generated. This identifier is then searched for on the DLHC – if it does not exist, it is assigned to the new patient. If it does exist (a one in $3x10^9$ likelihood), a new random identifier is generated until a unique identifier is found. Once an identifier is generated, the patient generates a public and a private key using an elliptical curve algorithm. The use of elliptical curve cryptography (ECC) over systems like the RSA suite of algorithms or el Gamal was done to minimize the key length. ECC requires a key size of approximately twice that used in traditionally symmetric algorithms, instead of ten to twenty times the size as required by other public key approaches[53]. Given the storage of uncompressable public keys in a large index structure, the key length is an important consideration.

The public/private key pair can be generated either by the provider's system or patient generated. The provider generated key provides simplicity, but the private key will be available to the provider, if only briefly. Additionally, a mechanism to transmit the private portion to the patient either through email or locally on a smartcard or USB key is needed. A patient-generated key would require the patient to give the public key to the provider after generation. This has the advantage of allowing the patient to generate their own key offline, and ensure they are the only ones with access to the private portion.

Though both approaches are supported, DLHC recommends the former approach to enable an added benefit – key escrow. When a patient selects a primary care physician, that physician can become an escrow agent for the patient's private key. In the event of patient incapacity or patient death, the physician can authorize the disclosure of patient information on their behalf.

The 64-bit record identifier is used to provide a unique ID for each record in the DLHC. The ID consists of a sequential 32-bit identifier generated by the provider, prefixed with the provider's own unique ID.

The remainder of the metadata is optional and is stored in two sections – one for personally identifiable information (PII) and the second for non-PII medical information. The PII information is protected (and can be encrypted with the patient's private key if supported by a particular implementation). The non-PII section is not encrypted and can be searched using a parser for providers that support a richer query language. If the medical records are already in a markup language like MML[54], the entire record becomes the second half of the metaschema. If the record is in a proprietary format, the data can be translated to MML and/or the raw text extracted and used for basic keyword searches on the second portion of the metaschema. A sample record is displayed in Figure 4.

Preamble	**Record ID**	**Patient ID**	**Public Key**
	64 Bits	64 Bits	512 Bits
PII	Personally Identifiable Information in MML		
Non-PII	Non Personally Identifiable Information in MML		
Original Record	Note: This may be transformed to the above metaschema format and the original removed or archived. There is no prescribed format for original records.		

Fig. 4. Metaschema for medical records.

E. Provider

The provider architecture is intended to be platform and language independent. In fact, having multiple providers developed using different languages for different platforms provides protection against platform or language specific flaws that may arise in the future[55].

The high level structures to be implemented by the provider are noted above. To achieve interoperability with clients and each other, both a common set of routines and a unifying communication mechanism must be implemented. The routines to be used are detailed below, and the communication mechanism is detailed here.

Each provider consists conceptually of a communicator, an indexer, a search agent, and a synchronization agent. The communicator provides a mechanism for contacting and receiving contact with external entities. The indexer maintains the individual indices noted above and provides indexing on new documents as they are added. The search agent performs internal searches in response to queries. The synchronization agent is responsible for ensuring the integrity of individual patient records. Note these are

logical functions of the provider, not actual modular implementations. The specific modular breakdown is determined by the implementer.

The communicator will operate using web services as the communications model. The underlying transport will utilize the Simple Object Access Protocol (SOAP)[56] to interact with both clients and other providers. SOAP was chosen both for its simplicity and for the availability of existing libraries to draw upon for new provider development. As an added benefit, extensive security extensions are available to ensure transport-level security without requiring additional design[57]. WSDL[58] is not needed as the operations are already predefined, though it may be used for individual providers to make available a more extensive search capability. Additionally, UDDI[59] is not needed since there is no centralized directory of these services (it is distributed).

The indexer structures are defined above. The specific tokenization and parsing tools implemented by the indexer will be specific to the existing record formats used by the underlying record structures. As an example, an indexer for video transcriptions would need to perform speech-to-text conversion, whereas an index for MML information would need to understand XML. As an additional task, the indexer is responsible for populating and updating all of the components of the metaschema as records change, except for the preamble. As an added security feature, the indexer must remove any references to patient name or other personally identifiable information from the parsed data so it is not included in the indices. To respond to rich-media requests, the indexer is generates any non-textual media abstractions supported, including thumbnails and summaries[40].

The search agent provides a link between the SOAP requests and the underlying data. When a search request is received, the search agent queries the appropriate index. In the case of the patient index, the search agent first verifies the validity of the request by decrypting a nonce provided by the communicator to the requestor using the appropriate patient public key. If it matches, the hits are returned. If it does not match, no data is returned – the same result as if there is no record available. The details of this transaction are shown in Figure 5.

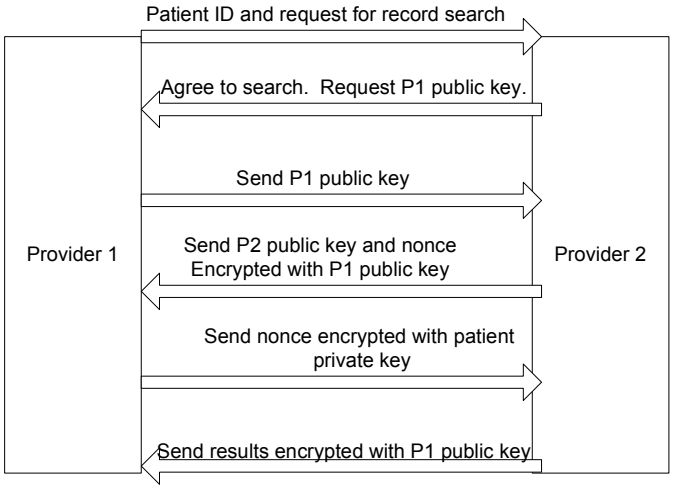

Fig. 5. Patient authentication for searching.

The synchronization agent is responsible for keeping the indices replicated with same-tier peers and in the case of regional providers for lower-tier peers. Additionally, the synchronizer is responsible for maintaining record accuracy with peer record sets. For index updates, the synchronization agent requests all peer entries which have changed in the past 24 hours according to the last updated date. To maintain date accuracy, all dates are kept based on Coordinated Universal Time (UTC)[3]. For deleted items, the synchronization agent maintains a simple list of deleted entries with the time of deletion. These are discarded every thirty days on a rolling basis. If a system is offline for a longer period, it must perform a full replication with a peer system to come back online.

For patient records themselves, the synchronization agent is sent a torrent-like file with links to all of the information on that file. Using the standard BitTorrent protocol, the patient record is then downloaded from all peers which maintain that record, reducing the download impact on any one peer[60]. In addition to the use of torrents, only those portions of records that have changed are replicated. Using the Rsync algorithm and a series of hash values, DLHC is able to only send changed and/or added portions of text and not entire records where only a portion of a record has changed[61].

F. Client

The client architecture is largely undefined in DLHC

[3] Keeping the resolution to the specific date means that clients must make requests from a specific provider if they need results before 24 hours have elapsed. If more frequent updates are required by a provider, it can synchronize more frequently and assume any entries with the current date require updating. The additional overhead is expected to be minimal in terms of transfer bandwidth.

to provide for a large array of client needs. A simple web interface which allows an individual to bring back their own patient record after providing their private key might be the simplest form of interface. A more complex interface may be used by a national health system which needs to obtain aggregate statistics on a full geographic region. These statistics might require a complex search of both MML and non-MML formatted records.

Because of the diverse needs, the clients can be considered providers without peer-to-peer replication responsibilities (since providers can also be clients). Because this DLHC uses a service oriented architecture whereby the clients are merely electronic consumers that treat the entire DLHC system as one large provider.

V. Operations

There are several key operations which DLHC implements to provide the functionality detailed above, all of which are made accessible as web services. The operations can be separated into two categories - updates and queries.

A. Updates

There are three specific update types that are supported by DLHC – peer updates, parent->child updates, and child->parent updates. Each type is addressed individually below.

1) Peer Updates

Peer updates represent the synchronization of data between peers. There are three specific peer update operations – updating the patient index, updating the geographic index, and updating a patient record. Additionally, there is a rebuild operation defined for peers which are damaged and/or new peers being added.

Peer operations are performed between members that share the same parent node. There will be a maximum of eight peers in any particular level, and each peer agrees to maintain an amount of storage capable of replicating data from each of the other seven nodes. In the event there is a substantially uneven distribution of data, the largest of the peers will have their geographic scopes expanded and be placed in a parent position.

The geographic index update occurs on a daily basis at a minimum. Because the addition and removal of providers is an infrequent event (at the local level – system-wide it is still infrequent when compared to patient index updates), and the index structure has the ability to grow within a given level (up to eight peers), major changes to the overall geographic index are uncommon. As with the other updates, scheduling on a daily basis permits the scheduling of updates during times of low system usage.

To update the geographic index, the synchronization agent on a given peer sends a request to all of the other peers listed in its current geographic index. The only information passed in the request (other than node identification) is the time of last update. The peer receiving the request then sends back any changes in its geographic tree which have occurred since the last update time provided in a single set of records, prefixed as being adds, updates, or deletes, along with the associated changes to the record. The records are sent in an XML document update with a simple MD5 hash value appended to ensure accurate transmission.

If the last update time is longer than thirty days prior, the peer receiving the request sends a “request denied” response. Additionally, if a malformed request is received or a request is received from a node listed as “deleted”, a “request denied” is sent. If there are no updates, a “no update” response is received.

The sender of the request waits until all of the peer updates have been received or a timeout has been reached. All of the peer updates received before the timeout are then merged, with discrepancies settled by using the most current last update time.

If a node does not respond by the timeout period, it is added to a “to delete” list which is kept locally. If the node does not respond after thirty days, it is deleted from the geographic index.

The update of patient indices proceeds in exactly the same fashion as the geographic index. Because there is a potential for a large number of patient index updates over the course of a day, these updates are likewise scheduled for daily update at low bandwidth utilization periods.

Each of the updates noted above has up to eight different requests. Because the update is a one-way synchronization, there are 192 maximum updates which would occur daily on a given level within the geographic tree. While this appears substantial, the amount of data transferred for these updates is likely to be small, with the patient index being the larger of the two.

A large provider, such as the University of Chicago Hospital, may see an average of 1,095 patients per day[62]. Assuming 8 peers are all very large providers, this would mean 1,095 x 7 = 7,665 updates per peer per update in terms of actual updates. In reality, because duplicate updates are sent, the last peer might actually have 7,665 x 7 = 53,665 as a worst-case update received. Given the size of the patient record, this would result in

a worst-case update of 4.1 MB per day[4], well within acceptable bandwidth limits.

The most costly peer operation is the patient record synchronization. Given the numbers above, a high volume hospital may have 1,095 updates at 17KB per update. This yields 1,095 x 17KB = 19MB per peer, for a maximum of 19MB x 7 = 177 MB total at a given level generated daily. If the same synchronization routine as above were used, the impact may be signification. As such, a different synchronization operation is performed.

The patient record synchronization occurs after updating the patient index. For each of the new patient index records processed (not received as there are potential duplicates), a different request is generated based on the type of record change – either an addition or an update request.

Addition requests consist of two messages. First, a request is sent to all peers consisting of the record identifier. Each of the peers responds back with either a "record available" or a "record not available" response. The record available response consists of the record number, the last update date and time (from the record itself), the record size, an MD5 hash of the record contents, and a list of the available sections in the format of section name/section start byte/section end byte. Segment size is determined as MAX[record size/1024, 1KB].

The requestor evaluates each of the record available responses and compares the hash values. The most recently updated record is considered the master record. The requestor uses the MD5 hash of the master record to identify other records with the same content.

Once the additional records with the same content are identified, the requestor sends a series of additional messages to each of these providers requesting segments in a round-robin fashion. Because segments are requested in parallel round-robin, slower connections will automatically send less data. As with all of the messages, a timeout will identify non-responding providers.

Updates are more complex than additions. For each updated record, the same initial message used for additions is sent to each peer and the same algorithm for determining which peers to request from is used. Next, a decision is made on which update to use as follows:

```
If (RecordSize<SizeThresholdLow) Then
    Update Entire File
Else
    If (OldRecordSize/NewRecordSize<PercentThreshold)
        Update Entire File
    Else
        Run Rsync on a segment-by-segment basis
```

[4] This does not include envelope overhead and protocol overhead. Even at a 50% overhead rate, a size of 6.15MB is still acceptable.

The two thresholds are used to avoid the overhead associated with Rsync on very small changes and very large changes. For very small files, where the size of the file is less than a low threshold (determined by each provider based on their bandwidth availability), the file is simply retransmitted as a whole. Similarly, for very large files where there is a very large growth in file size (through the addition of an endoscopy video to an existing text record, for example) the entire record is treated as an addition. A reasonable threshold might be to use Rsync when the previous record is larger than 10% of the size of the new record.

For those records for which an update is warranted, the same round robin algorithm is used as for additions with two changes. First, the segment size is determined by the requestor based on their bandwidth availability. Larger segments are used to make Rsync more efficient. Second, for each segment used, the full Rsync operation is performed with one of the peers to only send the updates to that segment in a back-and-forth series of messages[63].

The final update operation for a peer is a bootstrap. The bootstrap is run in one of three situations – the peer is irreparably damaged, the peer represents a new node, or the peer has been out of sync for more than thirty days. A bootstrap may be performed offline (by copying the indices and records from an existing provider to tape) or online (through the bootstrap operation).

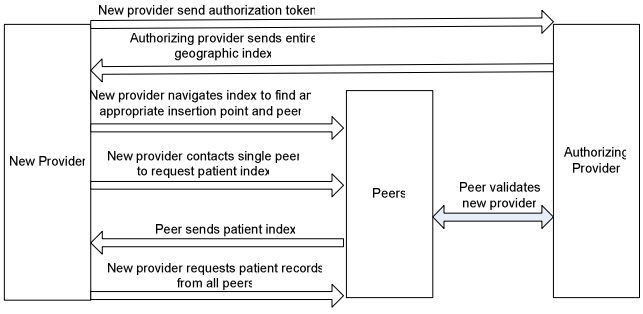

Fig. 6. Bootstrap process.

To bootstrap a new node, an out of band communication is made to identify at least one node in the DLHC. That node then gives authorization to the new node to bootstrap (the authorization process is outside the scope of this paper and may be either an online process of key exchange or an offline process – the end result being an authentication and authorization token that is given to the new peer and a new provider

ID). The peer bootstrap process is shown in Figure 6.

2) Parent->Child Updates

Parent->Child updates are updates where a parent requests updates from a child node. These updates are more frequent than peer updates, and are substantially smaller in size. In addition to the peer index updates, a keyword index update is performed. Patient records are not updated to parent nodes.

To update its geographic index, a parent node (defined as any node that has children) first sends a simple polling request to each of its child nodes. This poll request consists of a single "are you alive" message and response. If a child node is found to not be available, the parent stores that in a list. Any child nodes not responding for forty eight hours are considered temporarily lost, and those not responding for thirty days permanently lost. For available nodes, the same synchronization used by peers is performed, but the request is made to every child (instead of just one child) and it is made every hour to reduce propagation time as described below.

For nodes that are temporarily lost, the parent node is responsible for routing requests to any grandchildren directly under that node. As such, the parent node begins to poll those nodes until the old child comes back into operation. This is done in a recursive fashion up to a pre-defined limit based on the capacity of the parent. Any permanently lost nodes are removed by the parent and the geographic index reorganized as appropriate.

Patient record index updates are made with all children, similar to geographic updates. These indices are updated in the same fashion as peer updates, except the frequency is hourly instead of daily.

The keyword index updates are different than the other indices, given the potentially larger update volume. In addition to the index words themselves, each child maintains a first added (as opposed to a last updated) date with each word. Every hour, the parent node sends a request for keyword updates to each of its children. The children each respond with an XML document containing all of the keywords that have been added or deleted in the past day, with an MD5 checksum of the entire document. If the MD5 sums match, the keyword lists for all of the peers are merged into a single list, with duplicates removed. Each keyword is then merged into the parent's list of external keyword entries.

3) Child->Parent Updates

The reverse of the parent child update, the child parent update ensures child nodes are synchronized with parent nodes for geographic index purposes (patient and keyword indices and patient records are not downward-propagated).

The geographic updates are performed on an hourly basis and take the same form as the previous updates in reverse. If a child node cannot contact its parent for greater than forty eight hours, it contacts a peer of its parent node for updates. If the node is unavailable for thirty days, the update with the parent's peer will automatically propagate the new geographic index (with a new parent) to the child.

4) Propagation Delay

There is a potential delay inherent in the DLHC for querying based on the propagation times of updates and the position being queried. The further the geographic distance between the client and the provider holding the record, the longer the potential delay.

There are two specific delay times to worry about – geographic propagation delays and patient/keyword propagation delays. The geographic propagation delays are potentially twice as long as the other delays, as they must propagate up and down the tree structure in the worst case scenario.

With a maximum child size of eight and a minimum child size of two (the worst case), the propagation delays can be easily calculated from the DLHC size. Per above, with a DLHC containing 360,000 providers would have a minimum depth of seven and a maximum depth of nineteen.

For geographic indices, the maximum propagation in the worst case would be eighteen hours for upward propagation and an additional eighteen hours for downward propagation given the hourly inter-level updates, for a maximum of thirty six hours. This case occurs when a new provider is added to a leaf node. Since a new provider's information wasn't previously available, this is an acceptable propagation time. The inter-level update frequencies could be shortened as needed to reduce this time.

Keyword and patient index requests are only propagated upward. Assuming the same levels as above, the maximum propagation delay for these would be eighteen hours. If more timely access to query information is required, individual provider queries and/or lower regional queries can be performed.

B. Queries

There are two types of queries supported by the DLHC – simple and complex. The simple query interface is designed to be easy to use and well defined, while the complex query interface is meant to be

flexible and allow for many different types of query operation.

At the heart of the simple query operation are three basic assumptions:

1. The most common query will be a request for a particular patient record. The system should be optimized for this.
2. Most records have a particular geographic scope associated with them. An individual living in Saratoga Springs, New York is more like to have a record in Albany than Azerbaijan.
3. Keyword searches can form the basis for more powerful searching. Complex queries making use of semantic information can be performed as an offshoot of keyword queries.

Based on the above assumptions, the two different query interfaces are defined below.

1) Simple Queries

There are three query types supported by the simple query interface – geographic scope queries, keyword queries, and patient ID queries. The simple query interface can be used with all three query types at once, and "simple" is meant to denote the interface as opposed to the power of these queries.

The simplest query is the geographic query, which returns all of the providers within a particular geographic region. Because this particular query is intended as a building block for use with the other queries, it is handled locally by each provider and/or client.

Since the geographic index is maintained locally, the geographic query consists of a simple navigation of the R-tree. Because every provider and client maintains a copy of the geographic index, there is no functionality provided for remote geographic searching.

The user interface design is not part of the specifications, but either text-based (type in a zip code and radius) or graphical interfaces are readily supported by the underlying structure.

The patient record query is a simple query for a patient record based on a patient ID. Because the patient query is percolated upward in the geographical index structure, the query can be performed at any layer from the root entries down. Because the root entries are likely to restrict direct querying, it makes sense to couple patient queries with a geographic query by selecting a starting scope where the expected record or records may reside.

The patient query starts at a particular provider that completely contains the region to be searched. The client contacts that provider directly, and that provider searches its own patient index. There are five possible search results:

- Patient record not found. There is no patient record that matches the identifier present in the hierarchy.
- Authentication failure. The authentication failed on accessing the patient record. The response sent back is a "patient record not found" response to prevent guessing attacks.[5]
- Record found locally. The record is available from peers at this level.
- Record present on child providers. The record is present on a child node.
- Record present locally and on child providers. Multiple records exist at a local and sublocal level.

If the record or records are present locally, the response includes the record identifier, the record size, and an MD5 hash of the record contents. The requestor then contacts the other peers associated with that record to determine if it is present on their systems. If it is, the requestor breaks up the record into segments and requests individual segments from each of the peers. The peer responses to the segment requests are encrypted with the key established in the original authentication request to prevent eavesdropping or insertion attacks.

If a record is identified as being below a particular geographic scope, that fact is returned to the requestor. The requestor then recursively queries each of the identified lower-level providers until the record is found.

The difficulty with the patient query is centered on accidental disclosure of a patient record to an unauthorized individual. While sanitized records (those stripped of PII) are permissible for open sharing, full records must be protected, hence the structure and procedure identified for authentication in Figure 5.

Because of the confidentiality restrictions, a mapping of patient ID's to patient names is not done centrally. This may seem like needed functionality, but given the ambiguity of names (e.g. John Smith) and the ability to associate patient ID's with a particular provider if you are a regional provider or have access to a regional provider's data, this is not feasible from a privacy perspective.

The final and most complex query operation is the keyword query. As with the patient query, the keyword

[5] A timer delay with a random amount of wait time can be built in to defeat timing attacks.

query is generally coupled with a geographic query. Global queries may be supported for certain providers, however, to allow for the generation of overall global statistical data.

The keyword query begins by identifying a provider with a specific scope. The provider is then sent up to three messages – once to determine if there are matching keywords at or below that provider's scope, and a second time to request summaries of documents that match those keywords. Finally, individual messages requesting specific patient records are sent.

The initial message takes the form of an XML message with individual keywords. The response back is one of four possible messages, there are no hits, there are local hits, there are child hits, or there are both local and child hits. An example return message snippet might be as follows:

```
<keyword value='Asbestosis'>
        <children>True</children>
        <local>
                <record>123SOMEID…</record>
                <record>124SOMEID…</record>
                <record>125SOMEID …</record>
        </local>
</keyword>
<keyword value='Chronic'>
        <children>True</children>
    <local>
    </local>
</keyword>
```

Once the results are returned, the requestor is responsible for applying any Boolean operations on the document list to provide basic or extended Boolean functionality. The requestor then makes a second request, based on the results of the Boolean operation, to the provider for document summary information. Additionally, the requestor may, if it is relevant to a particular query, send recursive initial messages to the children until all relevant nodes have been exhausted.

The second message response depends on the type of record sent. The response may be simple text snippets, or it may contain binary information (coded as ASCII) which has thumbnails or video snippets. Additional types of rich responses may be defined through the use of custom tags as an extension to the initial specification.

Based on the secondary response, the requestor is then able to provide a list of ranked results to the user using whatever ranking and display mechanism is appropriate. This may include simple result rankings, image galleries, or the use of a mapping API to show the keywords geographically.

The final message request is the request for a particular record based on the user's choice above. The record request takes the same form as the patient request, though the personally identifiable information is stripped and not returned as part of the response to sidestep privacy concerns. An example process flow is shown in Figure 6.

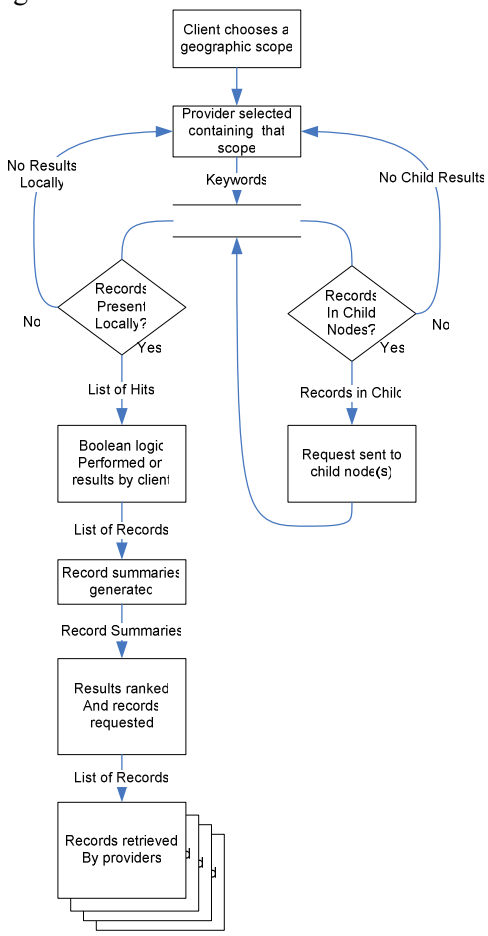

Fig. 6. Keyword query process flow

For automated requests, those where the results will be processed directly by a computer, the second request message may be skipped and the relevant records requested directly.

2) Advanced Queries

In addition to the simple query structure noted above, DLHC supports a framework which allows for advanced queries. Advanced queries may be of any type defined by a particular provider. A radiology provider may allow for similarity matching of tumor features, while a public health provider may provide structured querying of HL7-based patient records.

DLHC does not seek to strongly define every possible query type (or future possibilities for video and other searching). Instead, DLHC supports locally published WSDL to define the services available at a particular provider. An individual requestor may send a "list

query services available" message to the provider, and will receive the WSDL for any enhanced query abilities present.

VI. Patient Deletion

Deleting a particular patient record in a distributed system is a non-trivial task. There are two types of deletions – local deletions (delete a record from a specific provider) and mass deletions (delete every record for a given patient).

There are multiple reasons for record deletion. First, an organization may have a policy in force that records are only maintained for a specific period of time before being wiped out. This may be done for sensitivity reasons (a methadone clinic, for example) or for simple storage reasons (a high volume emergency room). Second, a patient may wish to expunge their own record from a specific provider or from the system as a whole. Third, specific records may be removed for a batch of patients based on class action suits, acts of law, etc.

When a deletion occurs for a specific provider, the record must be deleted from all of the peers of that provider as well (though only that distinct record – other records for the same patient may be retained).

For an individual provider deletion, the requestor first sends an authenticated query on patient ID. Based on the results of that query, the requestor sends a second message containing the same authentication information as was used in the query along with the record ID of the record to be deleted. The provider receiving that request is then responsible for marking that record as deleted. The deletion will be propagated to all of the peers of that provider (and to upward indices) based on the propagation delays noted above. In the event a more rapid deletion is required, the requestor may individually contact all of the peers that hold the record with separate deletion requests.

For a mass deletion, it is the responsibility of the requestor to identify locations the patient records exist then individually contact providers. A cascading deletion is technically possible, but the potential for abuse make the operation too risky to implement, given an alternative exists.

VII. Future Work

The DLHC has been defined and examined from a framework standpoint, but there is significant room for enhancement and extension to the existing model. Specific areas for extension include security, querying, and geographic replication.

A. Security

The design of DLHC takes into account security at a base level, but several scenarios bear further evaluation.

First, while DLHC makes it difficult for a malicious provider to come online without collusion, it does not prevent subversion through the injection of multiple bad records by attacking the platform or implementation of a given provider. While a voting mechanism is inappropriate for DLHC, the patient and the record creator are the ultimate arbiters of information accuracy should a question arise – it is in the detection of issues that future work can be done. A mechanism for patients and record creators to be notified of changes to their records would be an easy addition.

Second, DLHC does not put in place an intrusion detection capability. By integrating intrusion detection and adding human review of anomalous events, such as the requested update of a large volume of data, a further layer of security could be achieved. A simple threshold detector based on the number of queries/updates/etc. compared to the average would detect denial of service attacks.

Finally, revocation of a malicious node must be done on a system-by-system basis. A more robust mechanism for revocation for providers and patients, as well as a client authentication mechanism, would increase the overall strength of DLHC.

B. Querying

While built to be extensible for advanced queries, there is the potential to support some advanced queries out-of-the-box in the future. The types of queries implemented would depend heavily on the usage models seen in practice.

The first area for query enhancement would be better aggregate queries. These would be variable/value pairs used by statisticians to support natural language queries where only specific values of certain attributes are sought, and a temporal factor is included, such as "How many patients where sex=male and age>50 who had diabetes were seen in the Midwest last year?"

The second area for query enhancement would be rich media queries. The search of x-rays for similar fractures or the search of a colonoscopy video for key regions of interest require more local research before being implemented on a distributed scale, but would be of great benefit to doctors searching for similar cases.

C. Redundancy

The current DLHC design replicates information

locally for redundancy purposes. This follows the principle of spatial locality – most searches for records will be performed geographically close to where they physically reside. There are two downsides to this implementation – redundancy and remote performance.

For redundancy, using spatially close replication puts data at risk for regional catastrophes. Hurricanes, monsoons, tidal waves, earthquakes and other natural disasters are regional in nature. Likewise, human-caused outages like explosions, digging, and fires can have regional impact. Because of this, the highest level of availability that can be achieved on an Internet-connected system without geographic redundancy is asymptotic to 99.95%[64]. One solution might be to implement virtual providers that are geographically distant but appear on the geographic index as being regionally co-located.

The second concern with locality is performance. The latency associated with a request made thousands of miles away will always be greater, on the average, than that of a local request. This is becoming less of an issue as long-haul bandwidths increase, but for quick responses to queries and updates, having a close copy would be useful. Though this would not affect the individual query much based on the spatial locality principle, aggregate queries would benefit from closer copies. As with the availability concern above, this could be addressed quickly with virtual providers.

VIII. Conclusion

The Digital Library for Healthcare addresses the need for access to electronic patient records using a hybrid peer-to-peer and service oriented architecture approach. By defining a series of services that sit on top of existing systems and are platform independent, and by using successful peer-to-peer techniques to perform rapid searching and information retrieval, the DLHC's framework is scalable, secure, and has few barriers to successful implementation and integration.

References

[1] J. Prow, "32 & 16 Years Ago," in *IEEE Computer*. vol. 40, 2007, p. 15.

[2] W. Pratt, K. Unruh, A. Civan, and M. M. Skeels, "Personal health information management," *Communications of the ACM,* vol. 49, pp. 51-55, 2006.

[3] J. J. Cimino, V. L. Patel, and A. W. Kushniruk, "What do patients do with access to their medical records?," *Medinfo,* vol. 10, pp. 1440-4, 2001.

[4] B. Fisher, R. Fitton, C. Poirier, and D. Stables, "Patient record access--the time has come," *Stud Health Technol Inform,* vol. 121, pp. 162-7, 2006.

[5] C. C. Tsai and J. Starren, "msJAMA. Patient participation in electronic medical records," *Jama,* vol. 285, p. 1765, 2001.

[6] V. Masero, "Health care information systems," in *Proceedings of the 2005 ACM symposium on Applied computing* Santa Fe, New Mexico: ACM Press, 2005.

[7] J. J. Cimino, V. L. Patel, and A. W. Kushniruk, "The patient clinical information system (PatCIS): technical solutions for and experience with giving patients access to their electronic medical records," *Int J Med Inform,* vol. 68, pp. 113-27, 2002.

[8] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G. B. Laleci, "A survey and analysis of Electronic Healthcare Record standards," *ACM Comput. Surv.,* vol. 37, pp. 277-315, 2005.

[9] M. B. Moore, "Acceptance of information technology by health care professionals," in *Proceedings of the symposium on Computers and the quality of life* Philadelpia, Pennsylvania, United States: ACM Press, 1996.

[10] "Gnutella Homepage," http://www.gnutella.com. Accessed On: January 15, 2007

[11] H. K. Huang, Z. Aifeng, L. Brent, Z. Zheng, D. Jorge, K. Nelson, and L. W. C. Chan, "Data grid for large-scale medical image archive and analysis," in *Proceedings of the 13th annual ACM international conference on Multimedia* Singapore: ACM Press, 2005.

[12] H. Hung-Chang and K. Chung-Ta, "Modeling and evaluating peer-to-peer storage architectures," in *Proceedings of the Parallel and Distributed Processing Symposium*, pp. 24-29, 2002.

[13] H. Vagelis, J. C. Peter, P. Nagarajan, D. Yi, A. W. Jeffrey, and P. B. Redmond, "A flexible approach for electronic medical records exchange," in *Proceedings of the international workshop on Healthcare information and knowledge management* Arlington, Virginia, USA: ACM Press, 2006.

[14] S. Rieche, K. Wehrle, O. Landsiedel, S. Gotz, and L. Petrak, "Reliability of data in structured peer-to-peer systems," pp. 108-113, 2004.

[15] P. Judge and M. Ammar, "CITADEL: a content protection architecture for decentralized peer-to-peer file sharing systems," pp. 1496-1500, 2003.

[16] K. A. Stroetmann, M. Pieper, and V. N. Stroetmann, "Understanding patients: participatory approaches for the user evaluation of vital data presentation," in *Proceedings of the 2003 conference on Universal usability* Vancouver, British Columbia, Canada: ACM Press, 2003.

[17] M. Roantree, J. Murphy, and W. Hasselbring, "The OASIS multidatabase prototype," *SIGMOD Rec.,* vol. 28, pp. 97-103, 1999.

[18] M. Beyer, K. A. Kuhn, C. Meiler, S. Jablonski, and R. Lenz, "Towards a flexible, process-oriented IT architecture for an integrated healthcare network," in *Proceedings of the 2004 ACM symposium on Applied computing* Nicosia, Cyprus: ACM Press, 2004.

[19] W. M. Omar and A. Taleb-Bendiab, "Service Oriented Architecture for E-health Support Services Based on Grid Computing Over," in *Proceedings of the IEEE International Conference on Services Computing (SCC'06) - Volume 00*: IEEE Computer Society, 2006.

[20] M. Turner, F. Zhu, I. Kotsiopoulos, M. Russell, D. Budgen, K. Bennett, P. Brereton, J. Keane, P. Layzell, and M. Rigby, "Using Web Service Technologies to Create an Information Broker: An Experience Report," in *Proceedings of the 26th International Conference on Software Engineering*: IEEE Computer Society, 2004.

[21] W. M. Omar and A. Taleb-Bendiab, "E-health support services based on service-oriented architecture," *IT Professional,* vol. 8, pp. 35-41, 2006.

[22] A. Shail, "Object-oriented technology for health care and medical information systems: workshop report," in *Addendum to the proceedings of the 10th annual conference on Object-oriented programming systems, languages, and applications (Addendum)* Austin, Texas, United States: ACM Press, 1995.

[23] S. R. Amendolia, F. Estrella, R. McClatchey, D. Rogulin, and T. Solomonides, "Managing Pan-European mammography images and data using a service oriented architecture," pp. 99-108, 2004.

[24] E. Vasilescu and S. K. Mun, "Service Oriented Architecture (SOA) Implications for Large Scale Distributed Health Care Enterprises," pp. 91-94, 2006.

[25] D. W. Forslund, R. L. Phillips, D. G. Kilman, and J. L. Cook, "Experiences with a distributed virtual patient record system," *Proc AMIA Annu Fall Symp,* pp. 483-7, 1996.

[26] H. Takeda, Y. Matsumura, S. Kuwata, H. Nakano, N. Sakamoto, and R. Yamamoto, "Architecture for networked electronic patient record systems," *Int J Med Inform,* vol. 60, pp. 161-7, 2000.

[27] S. Abiteboul, B. Alexe, O. Benjelloun, B. Cautis, I. Fundulaki, T. Milo, and A. Sahuguet, "An Electronic Patient Record "on Steroids": Distributed, Peer-to-Peer, Secure and Privacy-conscious " in *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, Toronto, Canada, pp. 1273-1276, 2004.

[28] I. Bilykh, Y. Bychkov, D. Dahlem, J. H. Jahnke, G. McCallum, C. Obry, A. Onabajo, and C. Kuziemsky, "Can GRID services provide answers to the challenges of national health information sharing?," in *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research* Toronto, Ontario, Canada: IBM Press, 2003.

[29] M. R. Lyu, E. Yau, and S. Sze, "A multilingual, multimodal digital video library system," in *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries* Portland, Oregon, USA: ACM Press, 2002.

[30] Q. Liu, R. Safavi-Naini, and N. P. Sheppard, "Digital rights management for content distribution," in *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21* Adelaide, Australia: Australian Computer Society, Inc., 2003.

[31] M. Petros, R. Mema, T. J. Giuli, S. H. R. David, and B. Mary, "The LOCKSS peer-to-peer digital preservation system," *ACM Transactions on Computer Systems,* vol. 23, pp. 2-50, 2005.

[32] F. B. Bastani and I. L. Yen, "A Fault Tolerant Replicated Storage System," in *Proceedings of the Third International Conference on Data Engineering*: IEEE Computer Society, 1987.

[33] B. F. Cooper and H. Garcia-Molina, "Peer-to-peer data trading to preserve information," *ACM Trans. Inf. Syst.,* vol. 20, pp. 133-170, 2002.

[34] J. Lu and J. Callan, "Content-based retrieval in hybrid peer-to-peer networks," in *Proceedings of the twelfth international conference on Information and knowledge management* New Orleans, LA, USA: ACM Press, 2003.

[35] J. Prewett, "Cluster security - the paradigm shift," in *Fifth IEEE/ACM Intemational Symposium on Cluster Computing*, Cardiff, UK, pp. 74-76, 2005.

[36] "Health Insurance Portability and Accountability Act ". vol. 45 CFR 164.308, U.S., Ed., 1996.

[37] M. Markel, "Safe harbor and privacy protection: a looming issue for IT professionals," *Professional Communication, IEEE Transactions on,* vol. 49, pp. 1-11, 2006.

[38] Z. Yinghua, X. Xing, W. Chuang, G. Yuchang, and M. Wei-Ying, "Hybrid index structures for location-based web search," in *Proceedings of the 14th ACM international conference on Information and knowledge management* Bremen, Germany: ACM Press, 2005.

[39] S. H. Jagbir, D. Erdogan, and S. Raj, "Health Level-7 compliant clinical patient records system," in *Proceedings of the 2004 ACM symposium on Applied computing* Nicosia, Cyprus: ACM Press, 2004.

[40] M. G. Christel, D. B. Winkler, and C. R. Taylor, "Multimedia abstractions for a digital video library," in *Proceedings of the second ACM international conference on Digital libraries* Philadelphia, Pennsylvania, United States: ACM Press, 1997.

[41] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data* Boston, Massachusetts: ACM Press, 1984.

[42] L. Arge, M. d. Berg, H. J. Haverkort, and K. Yi, "The Priority R-tree: a practically efficient and worst-case optimal R-tree," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* Paris, France: ACM Press, 2004.

[43] J. Surrago, "Doctor's offices and clinics," in *Rough Notes*. vol. 2, February 2001, pp. 1-2.

[44] D. A. Lelewer and D. S. Hirschberg, "Data compression," *ACM Comput. Surv.,* vol. 19, pp. 261-296, 1987.

[45] USCB, "World Population Information," United States Census Bureau, 2006.

[46] R. Bayer, "Binary B-trees for virtual memory," in *Proceedings of the 1971 ACM SIGFIDET Workshop*, New York, New York, pp. 219-235, 1971.

[47] S. Falk, E. W. Hugh, Y. John, and Z. Justin, "Compression of inverted indexes For fast query evaluation," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* Tampere, Finland: ACM Press, 2002.

[48] A. Ebidia, C. Mulder, B. Tripp, and M. W. Morgan, "Getting data out of the electronic patient record: critical steps in building a data warehouse for decision support," *Proc AMIA Symp,* pp. 745-9, 1999.

[49] G. O'Neill and P. P. Barry, "Training Physicians in Geriatric Care: Responding to Critical Need," *Public Policy and Aging Report,* vol. 13, pp. 17-21, 2003.

[50] R. Baeza-Yates and B. d. A. N. Ribeiro, *Modern information retrieval*. New York: ACM Press 1999.

[51] B. Shekhar, "Getting Gigascale Chips: Challenges and Opportunities in Continuing Moore's Law," *Queue,* vol. 1, pp. 26-33, 2003.

[52] W. B. Frakes and R. Baeza-Yates, "Information retrieval: data structures and algorithms," Prentice-Hall, Inc., 1992.

[53] G. B. Agnew, R. C. Mullin, and S. A. Vanstone, "An implementation of elliptic curve cryptosystems over F₂155," *Selected Areas in Communications, IEEE Journal on,* vol. 11, pp. 804-813, 1993.

[54] Z. Chenghao, T. Yongqiang, F. Jie, Z. Guozhen, and Z. Jianguo, "Building Hospital EPR with IHE Technical Framework," pp. 5684-5686, 2005.

[55] C. Taylor and J. Alves-Foss, "Diversity as a computer defense mechanism," in *Proceedings of the 2005 workshop on New security paradigms* Lake Arrowhead, California: ACM Press, 2005.

[56] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen, "SOAP Version 1.2 Specification," W3C, 2003.

[57] M. A. Rahaman, A. Schaad, and M. Rits, "Towards secure SOAP message exchange in a SOA," in *Proceedings of the 3rd ACM workshop on Secure web services* Alexandria, Virginia, USA: ACM Press, 2006.

[58] "Web Services Description Language," http://www.w3.org/TR/wsdl. Accessed On: January 20, 2007

[59] "Universal Description, Discovery and Integration (UDDI) Specification,"http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3. Accessed On: January 20, 2007

[60] D. Arthur and R. Panigrahy, "Analyzing BitTorrent and related peer-to-peer networks," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm* Miami, Florida: ACM Press, 2006.

[61] T. Suel, P. Noel, and D. Trendafilov, "Improved file synchronization techniques for maintaining large replicated collections over slow networks," pp. 153-164, 2004.

[62] "University of Chicago Hospitals," http://em.uchicago.edu/fr_hosp.html. Accessed On: January 19, 2007

[63] "The Rsync algorithm," http://olstrans.sourceforge.net/release/OLS2000-rsync/OLS2000-rsync.html. Accessed On: January 19, 2007

[64] Qwest Communications, "Geographic Redundancy and the Internet," Philadelphia, Pennsylvania, Technical Note 2001.