Making Medical Records More Resilient

by

Robert Rudin

B.S. Electrical Engineering (2001)
University of Rochester

Submitted to the Engineering Systems Division
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Technology and Policy

at the

Massachusetts Institute of Technology
September 2007

Signature of Author...
Technology and Policy Program, Engineering Systems Division
August 13, 2007

Certified by...
Peter Szolovits
Professor of Electrical Engineering and Computer Science
Professor of Health Sciences and Technology
Thesis Supervisor

Accepted by..
Dava J. Newman
Professor of Aeronautics and Astronautics and Engineering Systems
Director, Technology and Policy Program

Making Medical Records More Resilient

by

Robert Rudin

Submitted to the Engineering Systems Division on August 13, 2007 in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Technology and Policy

ABSTRACT

Hurricane Katrina showed that the current methods for handling medical records are minimally resilient to large scale disasters. This research presents a preliminary model for measuring the resilience of medical records systems against public policy goals and uses the model to illuminate the current state of medical record resilience. From this analysis, three recommendations for how to make medical records more resilient are presented.
The recommendations are:

1) Federal and state governments should use the preliminary resilience model introduced here as the basis for compliance requirements for electronic medical record technical architectures.

2) Regional Health Information Organizations (RHIOs) should consider offering services in disaster management to healthcare organizations. This will help RHIOs create sustainable business models.

3) Storage companies should consider developing distributed storage solutions based on Distributed Hash Table (DHT) technology for medical record storage. Distributed storage would alleviate public concerns over privacy with centralized storage of medical records. Empirical evidence is presented demonstrating the performance of DHT technology using a prototype medical record system.

Thesis Supervisor: Peter Szolovits
Professor of Electrical Engineering and Computer Science
Professor of Health Sciences and Technology

Acknowledgements

Beginning – let alone completing – the work on this thesis would not have been possible without the assistance and support of many individuals who helped me through the phases of research, from brainstorming to proof-reading.

A special acknowledgement must go to my thesis and research advisor Professor Peter Szolovits, who introduced me to this subject matter and showed me both the interesting nature of health IT research and the profound impact it could have on people's lives. Professor Szolovits' extensive experience, encyclopedic knowledge and kind advice and encouragement have been invaluable to not only this work but also to the shaping of my career interests. Thank you Professor Szolovits.

Dr. Aneel Advani has taken the role of an unofficial reader for this thesis. He has given me numerous fruitful ideas in our many brainstorming sessions during which he generously spent hours educating me on the many disciplines with which he is familiar. I have very much enjoyed collaborating with him. Thank you Dr. Advani.

The Technology and Policy Program in the Engineering Systems Division is a remarkable program that has fostered a community of people interested in addressing hard socio-technical problems from multiple perspectives. I am as gong-ho about the goals and methods of TPP and ESD as I was when I began the program two years ago. The difference between now and then is that now I have many more reasons to justify my excitement. Conversing with my fellow students about research has taught me much that cannot be learned solely through formal instruction. Thank you TPP and ESD and the leadership and administrators who make these innovative programs possible and successful.

On the technical side, I received hours of support on Distributed Hash Table technology from Emil Sit at MIT's CSAIL. Emil's time and patience with me has not gone unappreciated. Similarly, William Simons and Clark Freifeld gave me significant assistance in the intricacies of the Indivo project. Thank you Emil, William and Clark.

The five people I interviewed provided valuable points of reference and validation of the arguments in this thesis. They each voluntarily gave up their time to fill out a survey or answer questions over the phone. Thank you to all.

The policy portion of this work was informed by my experience this past summer working at the Office of Management and Budget, overseeing the federal government's health IT policy initiatives. My supervisor at OMB, Kristy LaLonde, was an excellent mentor and provided direction for this thesis. Thank you Kristy.

My research was supported by a Research Assistantship from the nMesh project ("National Multiprotocol Ensemble for Self-scaling systems for Health") under a research contract from the National Library of Medicine, NIH U54LM008748.

Finally, I must thank my family and friends for their support during this whole process. Many people were patient and helpful as I fleshed out my ideas with them in long conversations. I am fortunate to know so many people who support and value education in general as well as my specific educational pursuits. Thank you to all.

Inevitably, I have missed acknowledging people who haven given me support or assistance and for that I humbly apologize.

R. S. R.
August 2007

Table of Contents

Table of Charts and Figures

Acronyms

AHIC	American Health Information Community
ASCO	American Society of Clinical Oncology
ASP	Application Service Provider
CCHIT	Certification Commission on Health IT
CHIN	Community Health Information Network
EMR	Electronic Medical Records
HIE	Health Information Exchange
HIPAA	Health Insurance Portability and Accountability Act
HISPC	Health Information Security and Privacy Collaboration
HITSP	Health IT Standards Panel
IT	Information Technology
KB	Kilobytes
MW	Merriam and Webster
NCVHS	National Committee on Vital Health Statistics
NHIN	National Health Information Network
MB	.Megabytes
PHR	Personal Health Record
RPO	Recovery Point Objective
RTO	Recovery Time Objective
SDO	Standards Development Organization

Chapter 1: Introduction

In late August of 2005, Hurricane Katrina made landfall on the United States causing large-scale devastation, a multitude of deaths, and scattered over one million people from their primary homes to temporary residences across the country. [1] Among the many people affected by the Category 5 hurricane were patients of the United States healthcare system who, after being displaced from their familiar healthcare routines, were forced to find new doctors and resume needed care in new and unfamiliar settings.

Adding to the worries of many of these patients who would have included people with cancer, HIV, diabetes and other chronic care conditions, medical information that was important for their course of treatment was either destroyed by the hurricane or inaccessible at the point of care. While comprehensive data describing the state of these medical records have not been collected, two anecdotal accounts provide an indication.

At the MD Anderson Cancer Center at the University of Texas, evacuees – some having just undergone surgery, others in need of it – filtered into the center's departments. [2] Many of these patients lacked information about their treatment plan. The job of piecing together patients' medical histories fell to the nurses who attempted to contact doctors and even left notes on Internet message boards hosted by the American Society of Clinical Oncology (ASCO) in the hope that doctors would see the messages. Without statistical data, we are left to wonder about the health effects of poorly-informed medical decisions in cases such as these.

A second example comes in the form of a "success story" from the aftermath of the hurricane: KatrinaHealth. This project was an online service to help individuals affected by Hurricane Katrina gain access to their medication information through authorized providers. [3] The need for this service was obvious as an estimated 40 percent of evacuees used one or more prescription medications before the storm hit. After a pilot starting September 12, 2005, the nationwide rollout occurred on September 22, allowing authorized doctors and pharmacists to access individual patients' previous 90 days of filled prescriptions.

KatrinaHealth certainly provided a valuable and perhaps life-saving service for many evacuees by giving them access to medication information, an important part of their medical records. However, the most striking revelation to come out of the KatrinaHealth project is not the speed at which such services can be created but rather the low expectations this country has toward medical record availability. Hurricane Katrina ended on August 29 yet the KatrinaHealth nationwide rollout did not occur until September 22, more than three weeks later. It took more than three weeks to send medical information – that was already in digital form – over largely preexisting hardware, software and network infrastructure to the places where the data were needed. This observation does not diminish the efforts of those who worked tirelessly to make KatrinaHealth a reality. (Indeed, their efforts were much more than technical in nature: the larger hurdles were policy, business and organizational.) It does, however, demonstrate that the US healthcare system was not prepared to handle the informational demands of a large-scale disaster.

Recent disasters, such as Hurricane Katrina and the events of 9/11, have kept disaster preparedness in the public spotlight and under political scrutiny to some extent. After a period of calm, attention may shift to other matters. It is therefore important to remember that these recent disruptions are not anomalies. Historical statistics show that 17 hurricanes on average directly strike the mainland United States each decade. Of those 17 hurricanes, six are category 3, 4 or 5, some of which cause devastating results: in 1992 Hurricane Andrew left 180,000 people homeless. [4, 5] Destructive natural disasters are not the sole province of hurricanes. The 50,000 residents of Grand Forks, ND were flooded in 1997 when the Red River flowed over its levees. Earthquakes and tornados also occur regularly in the US.

In addition to natural disasters, there are many other kinds of disruption that might impair the usefulness of medical records, such as power outages, fires, and terrorist activity including physical and information attacks. In a survey of 500 small businesses, 90% experienced at least one power outage in 1998.[6] Kaiser Permanente suffered from a 55 hour outage of their $4 billion EMR system.[7] One company claims that one percent of healthcare practices experience some kind of office disaster each year.[8]

While disasters will likely continue to occur with regular frequency, access to medical records will become increasingly important as medical treatments become more complicated and the population continues to age. Already today, remembering all the medical data relevant for care is impossible for many chronic care patients: "the average 75-year-old in the United States swallowed eight different prescription medications each day." [9]

Considering that 1) disasters are likely to continue to happen as they have historically, 2) during the recent Hurricane Katrina, the healthcare system performed poorly with regard to the delivery of medical data to the point of care, and 3) medical treatment plans are likely to become more complicated and therefore even more dependent on information as knowledge of diseases grows, one may ask: how can patient medical data be made more resilient to disasters?

This thesis is an attempt to answer that question. It explores the ways in which the US healthcare system can advance beyond using message boards for contacting doctors, beyond waiting three weeks for electronic prescription lists. This thesis attempts to understand medical record resilience and to outline how the US can better prepare its medical information for the catastrophic events that will inevitably occur.

The needed preparation has a long way to go: the vast majority of patient medical information is still recorded on paper, a medium which, as I discuss more formally, has poor resilience properties. According to a Center for Disease Control study, only 9.3% of office based-physicians in 2005 had full EMR functionality implemented.[10] Hospital EMR use is better, but still far from widespread: the American Hospital Association reports that 37% have moderate to high IT implementations.[11] The lagging adoption rate of EMRs is well-studied and discussed often in health IT literature and in the health IT activities of the federal government. [12, 13] President Bush himself has called for a personal EMR for every American by 2014, created a federal Office of the National Coordinator for Health IT (ONC) and mentioned health IT-related topics in the four most recent State of the Union Addresses.[14] The academic and political attention is largely due to benefits of EMRs that go far beyond increased capacity for disaster resilience. [15] Resilience is then just one of many reasons for encouraging EMR adoption. However, putting medical data into electronic form will not alone cure this nation of its resilience ailments as demonstrated by KatrinaHealth and the ASCO message board examples in which much of the medical information was already in digital format but not readily accessible during disaster.

As healthcare transitions slowly and chaotically into the computer age, resilience may lag far behind EMR adoption as suggested by anecdotal estimates that only 20% of hospitals have advanced data storage architectures and by the sophisticated IT knowledge and maintenance demands of advanced storage architectures, large hurdles especially for small providers.[16] EMR systems implementations – including disaster resilience aspects – show huge variety as evidenced by the over 200 vendors of EMR systems.[17] Some resilience plans will be better than others. How can these differences be better understood and perhaps even captured empirically in a way that can be measured against policy goals of resilience? What policies, strategies and technologies will most effectively achieve these goals? This thesis addresses both of these questions.

The overarching structure of this thesis is simple: problem-solution. Part 1 “A Framework of Resilience and its Application to Today’s Healthcare” defines a resilience model and uses it to measure resilience in today’s healthcare system, formally showing the problem. Part 2, “Improving Resilience” suggests solutions.

Chapter 2 defines a preliminary resilience model in terms of policy goals as opposed to other definitions which have the more narrow aim of institutional recovery. For the model to be useful, however, it must be able to measure real implementations of EMR systems. Chapters 3 and 4 look at the two important aspects of an EMR implementation: technical architectures and communication channels. Chapter 3 considers a number of different technical architectures, their patterns of adoption, and assigns them resilience scores according to the model from chapter 2. Similarly, chapter 4 investigates the communication channels available during disasters for patient medical data, their patterns of adoption and how they map to resilience goals. Communication channels are found to be the most notable bottleneck for resilience. Chapter 5 finishes part 1 by summarizing the current state of resilience using data from a small survey of healthcare executives and offers explanations for the current state of resilience.

Chapter 6 begins part 2 and the search for solutions to the minimal EMR resilience by looking at existing efforts to open the communication channels for patient medical information and the key challenges faced by these efforts. The chapter then offers a possible strategy for overcoming these challenges. Chapter 7 continues to address the challenges of existing efforts at opening communication channels by introducing a technology that is new to EMR storage called Distributed Hash Tables (DHT), and analyzes how this new technology could help improve and accelerate EMR resilience. Performance results of storing patient records on an experimental DHT are presented. Then, chapter 8 explores public policy solutions by analyzing the role of government – an overwhelming force in the healthcare sector – and the policy options that are available to it. Chapter 9 reviews the recommendations made in this thesis and summarizes.

A survey that was administered to several healthcare executives is appended along with source code from the DHT storage experiment.

The holy grail of medical record resilience analysis would be a cost-benefit analysis, where cost is the total price of the medical record implementation including supporting infrastructure and maintenance costs, and the benefit is health outcomes, risk-adjusted for the various kinds of disasters that could impair functionality of the medical record system. This thesis does not explore the cost of medical record systems other than through general comparison. Rather, it aims to contribute to the benefit side of the analysis by assigning measurable policy goals as proxies for health outcomes and evaluating medical record implementations in term of these goals. A detailed analysis of the risks of various disasters was not performed. Rather, this research focuses on large-scale disasters similar to Hurricane Katrina where populations are displaced from their home settings. A thorough risk analysis that relates probability distributions of disasters to policy priorities would be an appropriate extension of this work. Even without a risk analysis, the limitations of current resilience practices – particularly in the realm of communication channels – are made clear, illuminating worthwhile social and policy priorities for overcoming the limitations.

An important result of this work is that medical record resilience should not be viewed as an independent policy objective but rather as part of the larger movement to modernize and integrate healthcare. This thesis aspires to inform that movement by advocating resilience be used as a policy goal and by arguing for the strategic role that resilience could play in the evolution of modern healthcare.

PART 1: A Framework of Resilience and its Application to Today’s Healthcare

Chapter 2: Steps toward Defining Resilience as a Policy Goal

Most would readily agree that medical records should be resilient to natural disasters and other disruptions and that resilience is clearly a good thing for medical records. But what exactly is the meaning of the word "resilience?" It is used in many different contexts such as when a person has ability to "bounce back" from adversity. It is also used in material science, economics, and other fields of study. For each context, resilience has a specific connotation.

Merriam and Webster offer a general definition: "an ability to recover from or adjust easily to misfortune or change".[18] This definition can help to frame a more precise definition of the term for the case of medical records.

Implied in Merriam and Webster's definition are behaviors or qualities that are extant in the phenomenon under study to an expected degree, a baseline at which recovery is directed and which misfortune impairs. These are the qualities that are valued. Therefore, a complete definition of resilience for a particular thing must specify these valued qualities and the expected baseline measurements. Once these are clearly defined, "misfortune or change" is simply anything that threatens the baseline measurement of these qualities.

This thesis argues that value of medical data is defined by two general qualities. The first is preservability: the property that a record and all its internal data persist over time. As a baseline, we expect all of our electronic medical data to persist on whatever technical storage architecture in which the data resides until we decide to remove it. We expect paper notes to remain legible and electronic notes to remain retrievable and non-corrupted. Because electronic storage media are prone to "change or misfortune" such as disk failures, the guardians of these storage implementations are expected to build technical architectures for backing up the data. The effectiveness of these technical architectures in persisting the stored data after a disruption i.e. preservability, is therefore one component of the data's resilience.

The second quality that defines medical data resilience is availability: the property that the data remain accessible. Today, the expected baseline for availability is that medical data are available to the healthcare provider or institution that generated the medical data in the first place and that the data is specifically unavailable to unauthorized parties to protect patient privacy. The availability of medical data to a healthcare institution other than the one that generated the data is only minimally expected in today's healthcare system. (This issue of exchanging health information between institutions is a crucial factor of EMR resilience as will be explained in chapter 4.) Surprisingly, our baseline expectations for patient access to electronic or even paper medical records is quite low despite the requirements of Health Information Portability and Accountability Act (HIPAA) that patients are generally entitled to a copy of their medical record.[19] This is illustrated by a class assignment conducted by Peter Szolovits in which he identifies numerous difficulties in obtaining patient records and notes that "no student was offered any electronic copy of a record." [20] A more formal study finds a wide variation in fees and delivery times for requests of medical records.[21]

These two qualities of resilience – preservability and availability – have been captured to some extent by the federal law in the HIPAA Security Rule, published to the Federal Register on February 20, 2003 in the section entitled “Contingency plan.”[22] This regulation applies to “covered entities,” which include not only healthcare providers but also healthcare clearinghouses and health plans. Before proposing a revised model for resilience, it is worthwhile to investigate the resilience goals expressed in the wording of this federal rule and how they compare with our baseline expectations.

> (A) Data backup plan (Required).
> Establish and implement procedures to create and maintain retrievable exact copies of electronic protected health information.
> (B) Disaster recovery plan (Required).
> Establish (and implement as needed) procedures to restore any loss of data.
> (C) Emergency mode operation plan (Required).
> Establish (and implement as needed) procedures to enable continuation of critical business processes for protection of the security of electronic protected health information while operating in emergency mode.
> (D) Testing and revision procedures (Addressable).
> Implement procedures for periodic testing and revision of contingency plans.
> (E) Applications and data criticality analysis (Addressable).
> Assess the relative criticality of specific applications and data in support of other contingency plan components. [22]

Definitions of the terms used here (data backup, disaster recover, etc.) are not supplied; however the Notice of Proposal Rulemaking (NPRM) which preceded the official rule includes some cursory hints at the intention:

> i) An applications and data criticality analysis (an entity’s formal assessment of the sensitivity, vulnerabilities, and security of its programs and information it receives, manipulates, stores, and/or transmits).
> (ii) Data backup plan (a documented and routinely updated plan to create and maintain, for a specific period of time, retrievable exact copies of information).
> (iii) A disaster recovery plan (the part of an overall contingency plan that contains a process enabling an enterprise to restore any loss of data in the event of fire, vandalism, natural disaster, or system failure).

> (iv) Emergency mode operation plan (the part of an overall contingency plan that contains a process enabling an enterprise to continue to operate in the event of fire, vandalism, natural disaster, or system failure). [23]

The language in this rule is vague. It does not give specific guidance as to how to implement a data backup plan, data recovery plan or emergency mode operation plan, but rather leaves the organization to self-certify or find an external certification body. [22] The NPRM comments elucidate the intentions and implications of such a flexible rule: “For example, in a small physician practice, a contingency plan for system emergencies might be only a few pages long, and cover issues such as where backup diskettes must be stored, and the location of a backup personal computer (PC). At a large health plan, the contingency plan might consist of multiple volumes, and cover issues such as remote hot site operations and secure off-site storage of electronic media.” [23] The current federal rule, therefore, is that contingency plans are, for the most part, to be decided by the healthcare organization itself.

The rule’s flexibility has advantages and disadvantages for medical record resilience. The advantage is that the regulations may not overburden the guardians of patient data. Under this rule, the guardians can determine the contingency plan based on their own criteria rather than meticulously following the results of a bureaucratic process uninformed by the idiosyncrasies of each healthcare organization’s uniqueness.

The disadvantages are, first, that each healthcare organization is left to make all these determinations themselves or, perhaps, with the help of consultants. "Traditionally, disaster recovery/business continuity (DR/DB) services have been based on consulting services provided by vendors such as SunGuard or IBM."[24]. Some provider organizations may over-prepare while others may neglect important aspects of contingency thereby inefficiently allocating resources. More clarification of the specific goals that these rules intend to promote might help providers decide which technical architectures to implement and create consistency among healthcare institutions who have comparable criticality and risk environments.

A second disadvantage is that absent more specific rules, healthcare institutions will implement these plans based on their own objectives, which may differ from the objectives of public policy. For example, a good data recovery plan for an institution may be unhelpful for an individual who was forced by a natural disaster to flee to another part of the country that lacks the capability to communicate with the original institution. The institution’s risk profile will likely be different from that of the patients, but the risk profile that forms the basis of the contingency plan will likely be that of the institution. Furthermore, because the healthcare institution is given so much leeway to determine its own contingency plan, the institution may tend toward under-preparing for unlikely but catastrophic disasters to cut costs, tantamount to gambling with patient records. This rule therefore primarily encourages business continuity of healthcare organizations rather than continuity and effectiveness of patient care. More specificity in contingency plan requirements may help direct the efforts of the healthcare institutions toward the goal of providing patient care during and after disasters rather than providing patient care *by the same institution* during and after disasters. An effort to increase specificity, if done effectively, should also be able to avoid over-burdening healthcare organizations with excessive or inflexible requirements, thereby retaining the advantages of the current vague requirements.

Specificity can be achieved through the two qualities of resilience, preservability and recoverability, both of which are reflected to some extent in the HIPAA Security rule. Preservability of medical records is accounted for in the data backup plan requirement; availability is accounted for in the emergency mode operation plan and the disaster recovery plan. However the rule does not show how to measure the effectiveness of any of these plans. If healthcare organizations could measure their technical implementations against the policy goals implicit in this rule, those policy goals would stand a much greater chance of being achieved.

Industry standards offer a way to specify and measure resilience: recovery point objective (RPO) and recovery time objective (RTO). RPO is "the maximum amount of data loss an organization can sustain during an event." [25] One way to think about RPO is the time between periodic backups. For example, if the RPO were one day in length, meeting the objective would require backups to occur daily. RTO is "the period of time within which systems, applications, or functions must be recovered after an outage."[25] Many disaster recovery strategies establish RPO and RTO benchmarks.

This research proposes a preliminary model of resilience using metrics that expand upon industry RPO and RTO by answering policy goals instead of organizational needs, metrics that can be evaluated against technical architectures. Policymakers could then more precisely specify their goals and healthcare institutions could have more instructions on how to implement their storage architectures. The metrics are presented in figure 1.

Preservability	Availability
Recoverability	Delivery time to point of care
Recovery point objective (RPO)	Deliver time of advanced features
Incremental data capture	

Figure 1: The Qualities of Resilience

Preservability, according to the model, has three factors: recoverability, recovery point objective (RPO), and incremental data capture. Recoverability refers to the ability to recover the data after a disruption. This factor is implied by the data backup plan requirement in the HIPAA Security Rule, but the degree of recoverability under the rule is to be determined by the healthcare institution. RPO is borrowed from the industry standard and expresses the maximum amount of time allowed between backups. The Security Rule does not provide a level of detail that covers RPO even though all designers of backup systems must consider this factor and the trade-offs between RPO and cost. Hence, enumerating it explicitly as a policy goal should lead to more clarified requirements. Finally, incremental data capture refers to the ability of a medical record system to continue recording a patient's information into the original system during and after a disruption. This factor may not be as important as the other factors of preservability because the new medical data might be captured by another system, one that is not affected by the disruption. But it is still a desired policy goal for the original medical record system to continue to accept new patient medical data to ease the patient's transition back to that system after the disruption is over. The Security Rule says nothing directly about incremental data capture. The "emergency mode operation plan" and "data recovery plan" may be interpreted to include incremental data capture, but, again, the goals would be left up to the healthcare organization. Combined, these three factors define medical record preservability.

Availability, in this model, has two factors: delivery time of data and delivery time of advanced features (if possible). The delivery time of data refers to the time it takes to get medical data to the place where it is needed for patient care. Therefore, patient health, and not business continuity, is the explicit goal. This factor is similar to the RTO which defines the amount of time needed to recover from the disaster. The object of RTO, however, usually refers to business recovery rather than the availability of data for patient care and is therefore not used in the model. The second factor of availability is the delivery time of advanced features such as role-based data access and clinical decision support features. Because of the strong privacy concerns for medical data, it is a desired feature for EMR systems to retain advanced access control functionality even during a disaster. KatrinaHealth did not use sophisticated role-based access but rather used a blanket authorization scheme that did not permit access to patient data by nurses, nurse practitioners or physician assistants.[3]

A system that makes medical data available but does not retain access control settings would receive a lower score in this category. Similarly, a system that makes data available but does not support emergency clinical decision support, or allow for easy integration to other systems' clinical decision support functionality, would suffer in this metric. To summarize, availability of medical data is availability of both the data and the expected functionality associated with the data at the point of care.

Resilience can be visually represented by plotting the preservability and availability metrics on a chart. Figures 4 and 7 in the following chapters show preservability on the X-axis and availability plotted on the Y-axis, the ideal implementation receiving a plot point at the upper right.

The value of these policy goals might vary with the type of medical data. A recent emergency responder use case that was approved by ONC specifies several data fields that might be desired for higher availability than others: allergies, medications, diagnosis, lab results, and immunizations. [26]

This model is the first iteration of what could become a guideline for HIPAA compliance of medical record technical architectures. By incorporating the resilience scores of medical record systems, cost estimates and the risks of various regional disasters, a framework can be established for evaluating the appropriate technical design against formalized certification requirements.

There are policy goals for EMRs other than resilience. One in particular has profound influence on the design of large-scale EMR storage architectures: privacy or, perhaps, the appearance of privacy. The idea of a centralized database of patient information, notwithstanding any security protection in the system such as encryption, raises strong public fears. According to a knowledgeable source on the matter, "for at least the last several years Massachusetts has been following an explicitly decentralized approach to all of its collaborative data projects. There are other states that take a more centralized approach."[27] This sensitivity to centralized medical information was also present in the minds of the designers of KatrinaHealth: "In addition to avoiding centralization where possible, KatrinaHealth was structured in such a way as to prohibit access to aggregate data." [3] The most resilient EMR system, if it contains a large centralized repository of data, may not pass muster in the harsh court of public opinion.

For the policy goals described in this chapter to be useful as metrics of resilience, they must be demonstrably applicable to medical record systems. That is the aim of the next two chapters. Medical record systems are divided into two components: technical architectures and communication channels. Technical architectures are the design of where the data are physically located and the mechanisms by which it is accessed. Technical architectures primarily impact preservability but still have some influence on availability. Communication channels are the connections between the sources and consumers of the data. Communication channels include not only physical wires and infrastructure but also logic connections i.e. the software infrastructure that supports communication, a requirement for communication to be effective. Communication channels primarily affect the availability metrics. Chapter 3 investigates the resilience of various medical record technical architectures assuming communication channels typical of today’s healthcare system. Chapter 4 investigates changes in the communication channels and how those changes affect resilience metrics. The metrics are scored as rough approximations for illustrative purposes. More rigorous study would be needed to formalize the resilience scores. In both chapters 3 and 4, cost implications are perfunctorily considered. Factors beyond technical architectures and communication channels may be also be important for resilience such as employee training for contingency IT operations. These factors are not discussed at length but could be incorporated into the model in future research.

Chapter 3: Technical Architectures and Resilience

Driven by the tendency of electronic media to fail and by the business requirements for high reliability, storage and backup solutions now compose a large industry which offers a huge and growing variety of implementation alternatives. Measuring the efficacy of the large diversity of technical architectures and storage alternatives is challenging. This chapter shows how technical architectures can be measured in terms of the policy objectives presented in chapter 2 by exploring some of the properties that characterize many popular storage architectures and relating those properties to policy goals qualitatively. Properties of technical architectures that bear upon resilience are described followed by some example technical architectures with various combinations of the properties. Each example architecture is given resilience scores based on qualitatively reasoned estimates of their resilience for a Hurricane Katrina-like disaster. This is not an exhaustive analysis of all medical record technical architectures but rather demonstration that some common categories of technical architectures can be roughly scored using the resilience metrics. To isolate the effect of changing the technical architecture on resilience, the resilience scores assume the communication channels of today. That is, the scores assume that medical data cannot be exchanged electronically between healthcare organizations and that patients cannot easily obtain an electronic copy of their medical record. Current practices of medical record technical architectures are briefly discussed at several points throughout the chapter.

A. Properties of Technical Architectures

The basic component layers of an EMR technical architecture are the storage medium, the storage architecture layer, and the higher-level application layer. The storage medium is the physical component that contains the data such as tape, CD, or hard disk. The storage architecture is the design that links the various storage media together including the hardware and software. A major part of the storage architecture is the database software, which may take a crucial role in the data backup and recovery. The higher-level application layer includes the software and hardware that support EMR functionality such as clinical decision support, clinician order entry and access to knowledge resources.[28] The storage medium can constrain or enable various configurations in the storage application layer. Likewise, the storage application layer can constrain or enable various configurations in the higher-level application layer. The three component layers are shown in Figure 3.

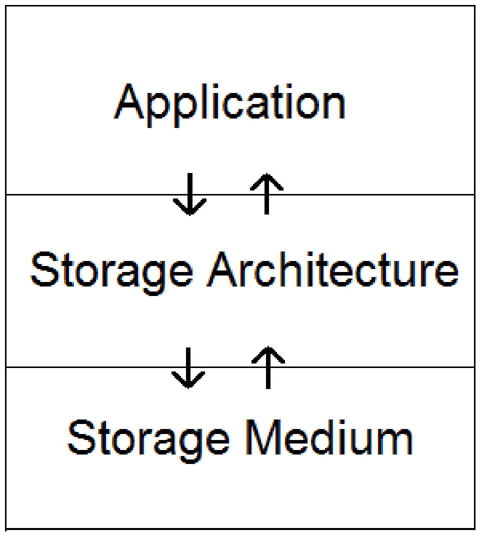

Figure 2: EMR Component Layers

Discussed next are five properties of technical architectures, most of which are best categorized at the storage architecture layer, the layer in which most of the backup planning is implemented. The relationships and constraints between the layers are also discussed in terms of the properties. The five properties are: master model, local versus remote storage, hot versus cold backup, synchronicity, and hot versus cold backup site.

Master Model

Backup storage implementations in use today can be master-slave or master-master models. This property usually describes the storage architecture layer. A master-slave implementation is one in which a primary (master) application process accepts the requests of the storage layer and one or more secondary storage processes (slave) replicate the backup and are used in case the master fails. There are variations on the master-slave model. The Google File System, for example, uses a master-slave for the writes but allows reads to be performed on many different machines.[29] A master-master model allows data reads and writes to be performed on multiple storage locations that are kept synchronized. The requirement of synchronization generally makes a master-master implementation much more expensive than a master-slave or results in slower performance. (Synchronization is discussed in more detail below.)

The master model affects several resilience metrics. It largely determines the recoverability score. Recoverability increases with an increase in the number of slaves and varies with the geographic location of the slaves. In practice, there will be one or two slaves for an EMR storage implementation. Some implementations may have crippled qualities after failover such as becoming "read-only", which would yield a lower score for "incremental data capture" and also a lower score possibly on the "delivery time of advanced features" under availability. This would depend on how much of the application layer was designed into the slaves. Master-master models might have performance constraints limiting the geographical dispersion and, therefore, also limiting the recoverability. Master-master models will probably have the same or better availability than most master-slave implementations because all sites of a master-master implementation would probably have higher-level application support. If the multiple masters were used for load balancing however, the failure of one master may impair the others during a disaster.

Local versus remote

Backup data copies can be made locally or remotely. To satisfy HIPAA compliance, even locally-made backups would have to be transported to a remote location, although it is unclear how far the distance would have to be. Some suggest that 25 miles away from the original source is an acceptable distance, but Hurricane Katrina has proven that general guideline to be too lenient. Regardless, backup sites are still only a few tens of miles away from the source.[30] The requirement of physically transporting the storage significantly limits the frequency of updating and the location of the remote storage. Several companies such as Iron Mountain offer these transportation services and the secure storage of data. Online storage offers the possibility of backup copies to be made remotely in any geographic location even thousands of miles from the original source. However, managing a remote location is impractical for many healthcare organizations, especially small providers. Some companies offer remote storage services at a premium price.[31]

The type of storage media used constrains the options for remote backup. Tape and CD are generally cheaper than disk; however they are difficult to implement on a remote site because they require managing physical components. Disk-based storage lends itself more toward automation and it is therefore the medium of choice for remote storage. Disk is also commonly used locally to protect against storage failures. One such example is RAID technology, which mirrors a hard disk so that in the event of a failure, a backup copy is readily at hand. RAID, however, requires the backup disk to be in close proximity to the primary disk and will therefore not help protect against a large-scale disruption.

The decision of local versus remote backups will affect both preservability and availability. Locally backing up and then transporting to a remote site restricts the options for how distant the backup site could be because of the effort involved in transportation for both storing and recovery of the data, therefore giving a lower recoverability score compared to remote backups. These logistical burdens will also probably cause local backups to inevitably have higher RPO scores relative to an on-line remote backup, widening the window of potential data loss. Online remote backup, however, is dependent on a working network and may have restrictions on location because of performance. Online remote backup would more likely have the capability to record incremental patient data compared to locally-made tape or disk backups, but that would depend on the backup system functionality in the application layer. Similarly, availability of a locally backed up site will likely be more difficult than from online sources because of the effort required to physically access the data and then extract it into a live system, thereby increasing the "delivery time to point of care."

Regardless of the advantages of online backup and the difficulties of managing local backups, many providers continue to use tape or disk backup because of cost considerations.[32, 33]

Hot Backup versus Cold Backup

A third property of storage architectures is whether a backup is "hot" or "cold." Hot backups are defined as backups performed while the system is in active use; cold backups are performed when the system is not in active use.[34] Hot backups generally require that the backup mechanism be integrated with the application or storage software (often the database) because an attempt to copy application data while the application is running may result in an inconsistent snapshot of the data. The might happen because the application may not have saved all its information or because in the course of generating a backup copy, the application may change some of the data. If the backup copy is integrated in the application or storage software, the software can generate a hot backup copy that is internally consistent. Database management system software packages often provide this functionality. Software with integrated hot backup functionality may be more expensive.[35] However, one low-end database product provides this functionality by recommending simply copying the database files in sequence during active operation to obtain a consistent hot backup.[36]

Hot backups will have better resilience scores in RPO and perhaps in availability than cold backups. Because hot backs can be performed more often, the window of potential data loss would be less. Hot backups may or may not be able to deliver advanced features and accept incremental data to a backup site, but cold backup implementations will likely not have these capabilities built-in because cold backups do not need to have application-level knowledge.

Synchronicity

Hot backups may have another property: synchronicity. A backup is synchronous if it is continuously updated at the exact same time that the primary store in updated. That is, when the application writes some data to storage, a synchronous system will not consider the write operation to be complete until the data are added to both the primary storage and the backup storage. (Cold backups are always asynchronous because the backup copy is created when the application is inactive.) If the backup site is a long distance from the primary or is accessed through a slow network, synchronous storage may be too slow to be practical. One extreme form of synchronous storage is a new storage feature known as Continuous Data Protection (CDP), which involves tracking every storage modification so that data recovery can start from any point in the past. Currently, CDP is targeted more for disk and database crashes rather than disaster recovery, possibly because of the difficulty of achieving high performance of CDP across long distances.[35] Synchronous storage can be managed by hardware, the file system, or as part of the application such as within the database. RAID technology is an example of synchronous storage managed by the hardware or the software depending on the implementation. Some use the term “disk mirroring” to describe synchronous storage to two hard disks. Remote mirroring is a form of synchronous backup. [37]

Asynchronous backup is when the backup copy is made or updated at some point in time after the primary storage mechanism receives new data or an update. The amount of time between when the primary store is updated and when the backup receives the update might be as short as milliseconds or as long as days if not longer. The length of this window is the RPO for the storage architecture. Because the data stored on the master is not synchronized with what is stored on the slave, the data would be inconsistent unless the application software managed the backup process to ensure consistency. That is, asynchronous backup by simply copying files periodically independent of the application will have the same consistency problems that any hot backup method would encounter. (Not all database implementations will have these consistency problems though.) Synchronous backup does not have this problem because the data would always be consistent between the master and the slave.

The primary advantage of synchronous backup is that it shortens the RPO to near-zero data loss.

Live Standby Site

Finally, technical architectures might have live standby sites that are fully operational the instant a disruption occurs or they may have cold standby sites that require some setup before they are operational. Some implementations may plan to have no standby site and buy off-the-shelf PCs in the event of a disruption. Live standby sites will have faster recovery times and therefore higher availability scores but will be costly. Cold standby sites may be cheaper, but will require configuration, which entails delay after a disruption.

B. Example Storage Architectures

Armed with some of the important properties of medical record technical architectures, the next step will be to give some concrete examples that have various combinations of these properties. Some of the properties may not make sense in combination such as cold backup and synchronicity. These examples are combinations that do make sense and may be currently in use. All of the examples assume some form of master-slave model.

Each example architectural design is scored on the resilience metrics for a Hurricane Katrina-like disaster. The scores will be different for different kinds of disasters. These numbers are an initial estimate of resilience on a 10 point scale and are therefore only ball park approximation. The estimates are meant to allow clear comparison of the technical architectures.

Architecture 1: Paper

The most basic and most commonly used storage architecture is paper, a medium that does not lend itself well to resilience because paper is easily perishable and copies are labor-intensive and expensive compared to electronic copies. However, paper does not tend to spontaneously erase data and therefore require active management like electronic storage.[38] During Hurricane Katrina, the records in the paper facilities that were flooded were largely destroyed. Paper without backup – which is the common practice as there is no legal requirement to backup paper medical records – would have a low metric on recoverability and RPO. While it is certainly possible to capture new information on paper, that is tantamount to creating a new record rather than using the storage system under study, so that incremental data capture is also scored quite low.

Availability metrics for paper are also rather poor. The delivery time to the point of care would be infinite if the paper was destroyed and would be difficult to transport because it can be voluminous. Paper also does not allow for advanced features such as clinical decision support.

Preservability:	Score (out of 10)
Recoverability of data after disruption	1
Recovery point objective (RPO)	1
Incremental data capture	1
Availability:	
Delivery time to point of care	1
Deliver time of advanced features	1

Chart 1: Architecture #1 Resilience Scores

Architecture 2: Tape backup (master-slave, local, cold, asynchronous, no live site)

An EMR system might backup the data to tape during off-hours daily or weekly and then transport the tape to a remote location. Tape backup systems can store large amounts of data cheaply, even with the price of disk space continuously dropping.

This storage architecture gives moderately good scores for recoverability especially if the backup site was not affected by the disaster. However, the requirement of physically transporting the data limits the destinations where the data can be stored. If the data is stored far away, it then becomes less available, so there is a tradeoff between recoverability and availability. The RPO will depend on the frequency of backups; cold backups cannot be made as often as hot backups, so the RPO will suffer, but even if the backups are made weekly, the RPO would still be good enough to restore a considerable amount of important medical data. Tape backups may not easily lend themselves to incremental data capture. The incremental data would have to be captured and then imported into the doctor’s electronic records after the disruption was over so the incremental score for this architecture would be low.

This architecture will have low availability scores because even if the data is recovered by the healthcare organization, how will it be transmitted to the point of care? Patients will still need to find their physicians or their healthcare organizations in order to get their EMR data. In addition, having to physically recover the data and then restore it, which may be troublesome if it requires a customized tape reader, will add to the delivery times. Even if the data is transmitted to the point of care, it will probably be done so verbally or in unstructured text and therefore impossible to be used with advanced features.

Tape backup if it requires a manual process is prone to error, lowing the recoverability score and increasing the delivery time.

Preservability:	Score (out of 10)
Recoverability of data after disruption	5
Recovery point objective (RPO)	7
Incremental data capture	2
Availability:	
Delivery time to point of care	3
Deliver time of advanced features	2

Chart 2: Architecture #2 Resilience Scores

Architecture 3: File backup to disk (master-slave, local, cold, asynchronous, no live site)

This storage architecture is similar to the previous one, but uses disk instead of tape backup. Disk will cost more than tape in general, but it can be restored faster than tapes hence giving higher availability scores. As with tape, incremental data recorded after the event would have to be imported back into the original provider storage system.

Delivery time of data such as clinical decision support would be easier than tape because the disk could potentially be used directly without a specialized reader. However, this architecture still suffers from the same problems as previous architecture for availability, such as patients having to locate their physicians during a disaster. Advanced features would still be impaired in the destroyed EMR system.

Preservability:	Score (out of 10)
Recoverability of data after disruption	5
Recovery point objective (RPO)	7
Incremental data capture	2
Availability:	
Delivery time to point of care	4
Deliver time of advanced features	2

Chart 3: Architecture #3 Resilience Scores

Architecture 4: Online file backup (master-slave, remote, cold, asynchronous, no live site)

The Internet offers the possibility of online backup directly to one or more remote locations. This will help greatly with recoverability if the backup location does not have its own disruptions or if multiple backup copies are made. However each additional backup copy will involve additional cost. RPO is the same as the previous architecture because a cold backup will still require the EMR system to be turned off. Incremental data capture will still be mostly infeasible.

Availability to the physician is better than the previous architecture because the data is accessible from anywhere online; however the patient still has no easy way to get the data and is forced to try to locate the displaced physician or organization. Also, the data will still have to be restored into a functioning EMR system, delaying the availability, but it will be faster than having to physically retrieve the data.

Preservability	Score (out of 10)
Recoverability of data after disruption	9
Recovery point objective (RPO)	7
Incremental data capture	2
Availability	
Delivery time to point of care	5
Deliver time of advanced features	2

Chart 4: Architecture #4 Resilience Scores

Architecture 5: Online database backup (master-slave, remote, hot, asynchronous, no live site)

As with the previous one, this setup will yield high recoverability. It will also give very high RPO because the hot backup can be performed while the database is running and therefore more frequently. Incrementally captured data will still have the same problems as the previous architecture.

Availability is unchanged from the previous architecture.

The database software is integral to this architecture. The cost of the database software for this additional functionality might be expensive.

Preservability	Score (out of 10)
Recoverability of data after disruption	9
Recovery point objective (RPO)	9
Incremental data capture	2
Availability	
Delivery time to point of care	5
Deliver time of advanced features	2

Chart 5: Architecture #5 Resilience Scores

Architecture 6: Online integrated database backup (master-slave, remote, hot, synchronous, hot backup site)

This is the most expensive and best storage architecture under investigation here: a fully enabled, synchronous backup site in a remote location. Large and small provider organizations will likely not have the resources for a completely redundant system. The fully-enabled backup site will be able to recover all the data after a disruption, but only one backup site scores shy of a perfect score because simultaneous failures do happen. KatrinaHealth advises “Plan ‘backups to backups,’ for when technology inevitably breaks down.” [3] The synchronicity shortens the RPO to zero so it gets the highest score for that category. Patients who use the backup site will be able to have their data added to directly to the new backup site, but many patients will not be able to use the backup site and will still have to re-import their new patient data after the disruption settles. The incremental data capture score is therefore mixed.

The serendipitous patients who either move physically near the backup site or another site that is integrated for data sharing with the backup site would have their medical data highly available. For the patients who were not relocated to the backup hospital location, getting their information to them with advanced features will still be difficult with current communication channels, but a backup site will be better capable of answering requests for displaced patients for their records.

Preservability	Score (out of 10)
Recoverability of data after disruption	9
Recovery point objective (RPO)	10
Incremental data capture	4
Availability	
Delivery time to point of care	6
Deliver time of advanced features	3

Chart 6: Architecture #6 Resilience Scores

A summary of the different architectures and the resiliency scores for a Hurricane Katrina-type disaster are shown in figure 4. (Different types of disaster will yield different resilience scores.) The metrics are weighted to reflect relative importance. Incremental data capture is given 20% of the 10 point scale. While it is important to be able to import the post-disaster medical data back into the original system, patient health outcomes will not be affected by that factor as much as the other preservability factors. Delivery time of advanced features is weighted at 40% of the 10 point scale. While access controls to protect privacy and clinical decision support functionality are quite important, getting the data to the point of care is the first priority. While the weightings and scores are contestable, the sample architectures were described to two Chief Information Officers of large New England Healthcare organizations who both agreed with the relative rankings.

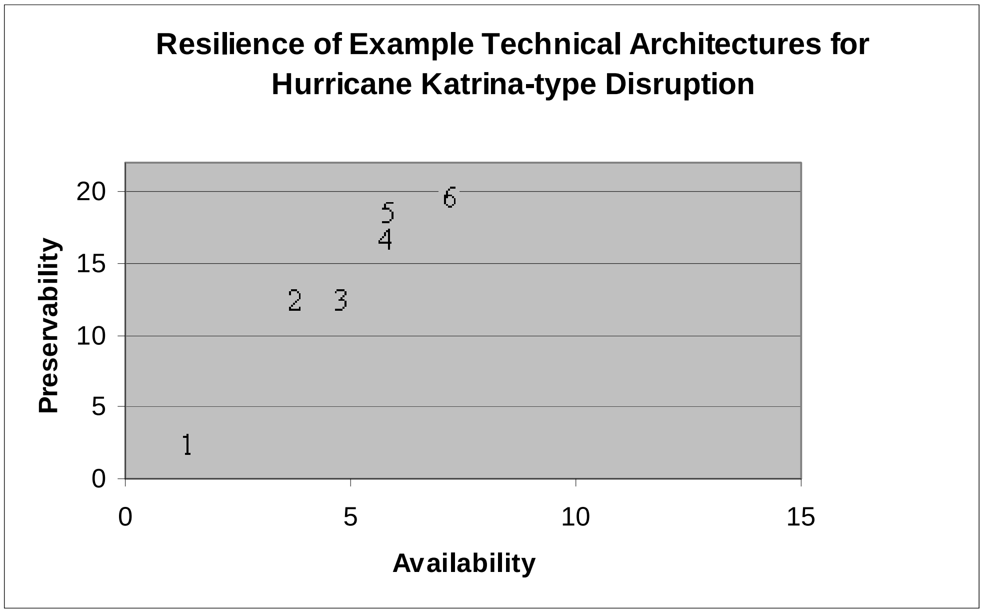

Figure 3: Resilience of Example Technical Architectures

Even without a more detailed evaluation of technical architectures, this model in its current form is revealing and informative of the capabilities and shortcoming of existing technology insofar as it addresses the policy goals of resilience for severe disasters such as Hurricane Katrina. As shown in figure 4, existing technology can be used to produce fairly good preservability for EMRs (except for the ability to incrementally capture data.) Cost may be a barrier for the high preservability systems, but the capability exists. Even the low cost computerized EMR architectures have reasonably good preservability if the secondary site is out of range of the disaster. Availability, however, is significantly lacking in all example storage architectures, even the most sophisticated and expensive. Notice that the right region of figure 4 – the high availability region – is completely barren of technical architectures. This graphically shows the limitations of improving only the technical architecture. To improve the crucially important factor of availability, communication channels must be considered, which is done in the next chapter.

Chapter 4: Communication Channels and Resilience

Chapter 3 showed that technical architectures for EMR systems are an important component of resilience for medical data. It also showed the limitations on resilience of technical architecture in today's healthcare, most importantly the limitations on availability. This chapter explores communication channels of medical information flows and demonstrates how improving these communication channels can increase resilience beyond the limits of technical architectures. Communication channels are divided into two categories: institution-institution and institution-patient. The institution-institution channel is primarily the exchange of information from one healthcare provider to another healthcare provider but it could include sources of medical information from other types of organizations such as laboratories and pharmacies. The destination however must be a point of care, usually a provider, for it to be relevant to the resilience model defined in chapter 2. The institution-patient channel refers to patient access to their own healthcare information. Patients are always at the point of care since patients are the ones receiving the care; therefore institution-patient communication is relevant to the resilience model without qualification. The transfer of medical information between health information systems is sometimes called Health Information Exchange (HIE). This chapter describes the currently deployed alternatives for each type of communication channel and, for each alternative, the extent of usage and effects on resilience.

Institution-institution channel

Of the currently deployed alternatives for inter-institutional exchange of medical records, almost none involve electronic HIE. Institutions have deployed very sophisticated methods of exchanging data inside their organizations, but very few examples can be found of electronic transfer of medical information between organizations.[1] In non-disaster situations today, when medical records must be exchanged between institutions, the records are typically photocopied, printed or, for image files, saved to CD and then carried or mailed to the destination healthcare provider. In disaster situations, when there is no time to prepare records, ad hoc projects such as KatrinaHealth are created or treatment uninformed by medical records is given.

Because disasters – such as Hurricane Katrina – often force patients to visit new providers who do not have previous copies of the patients' medical records,

[1] Even exchanging electronic medical records within institutions is rarely a trivial objective especially for healthcare providers who use best-of-breed systems made by different vendors.

disasters tend to stress the already weak institution-institution communication channel. (It is interesting that under the pressure of Hurricane Katrina, a system of health information exchange was created that provided better institution-institution communication than what is available today, almost two years after the hurricane.) Fortifying inter-institutional communication for normal, non-disaster situations by creating electronic HIE would indubitably strengthen inter-institutional communication during disasters as well.

HIE would likely have a profound effect on resilience. It would potentially decrease both the delivery time to point of care and delivery time of advanced features. It might also increase the capacity for incremental data capture if the post-disaster records could be easily integrated back into the original providers EMR system. Because of the lack of working examples and variation in possible designs of an HIE, this chapter does not evaluate any specific HIE designs against the resilience model. (Part 2 explores new ways to increase resilience through HIE which does involve evaluating HIE designs against the resilience model.)

Institution-Patient Channel

Similar to the institution-institution communication channel, the institution-patient communication channel has few glowing examples. As described previously, patients often have difficulty obtaining their medical record in paper, let alone in electronic form.[20, 21] One website advises patients by offering a "Scheme for getting patient medical records from recalcitrant HMOs."[39] According for Forrester Research Inc., 28% of US consumers track their medical information in some form of medical record.[40] However the figure for electronic personal health records is much lower, estimated by the Markle Foundation at only “250000 to 500000 people.” [41]

When patients do manage to obtain a copy of their medical data, they might store the information in a personal health record (PHR). PHRs are a dynamic and actively developing industry receiving growing attention in government and industry. The Secretary of the DHHS has identified PHRs as a top priority.[42] What exactly is a PHR? The National Committee on Vital Health Statistics (NCVHS) “found that there is no uniform definition of ‘personal health record’ in industry or government, and the concept continues to evolve.”[43] However most PHRs can be loosely divided into three categories: standalone, tethered and interconnected.[42] These PHR categories can be evaluated against the resilience metrics.

Standalone

The standalone PHR is the simplest approach. It involves the patient entering data herself into her record which might be a website or a USB drive. For a standalone PHR, the patient is responsible for assembling the record from various medical sources. Several companies offer these kinds of PHRs such as Medem Inc. and MedicAlert. The main problem with standalone PHRs is the considerable inconvenience of creating and maintaining them with updated information. Especially for patients with complex conditions, inputting all of a patient's medical information can be tiresome and error-prone. However, without better institution-patient communication, this is the only option for many patients who would want their medical information to be available during a disaster.

Chart 7 shows the resilience scores for architecture 7 which is composed of the most advanced technical architecture – number 6 described above – with the addition of a standalone PHR. The resilience scores are also determined in part by the technical architecture of the PHR – such as whether the PHR is carried by the patient or provided through a website. This example uses the architectures that yield the best resilience scores. Recoverability receives the highest score if the PHR is carried by the patient. If the PHR is an application service provider (ASP), the PHR would receive the recoverability score of ASP's technical architecture. The RPO would depend in part on how often the patient updates her PHR. Updating a standalone PHR continuously would be logistically near impossible.[44] Delivery time to point of care receives the highest score for patient-carried PHRs; however other variations of stand alone PHRs – such as access through a website - would lower the delivery time score for some scenarios such as when the patient was unconscious and could not log into a websites. A standalone PHR could potentially deliver some advanced access control features depending on how the software was constructed, because the patient would control the data. Clinical decision support and incremental data capture would not be improved because the EMR system of the patient's post-disaster provider would not be integrated with the PHR.

Preservability	Score (out of 10)
Recoverability of data after disruption	10
Recovery point objective (RPO)	10
Incremental data capture	4
Availability	
Delivery time to point of care	10
Deliver time of advanced features	5

Chart 7: Architecture #7 Resilience Scores with Standalone Patient-Carried, Fully Updated PHR (master-slave, remote, hot, synchronous, hot backup site, stand alone PHR)

It is believed that most PHRs in used today are not standalone but rather integrated with a provider's EMRs in some way.[42] The next two approaches for PHRs both involve PHRs that are integrated with EMRs.

Tethered

Tethered PHRs are more convenient than standalone PHRs because they are populated by the healthcare provider's EMR system automatically, ensuring a degree of reliability and saving time compared with the patient-entered approach. These advantages of convenience make tethered PHRs much more usable for patients compared with standalone PHRs. This can explain their greater – but still low – adoption rate. Tethered PHRs are tied (tethered) to one healthcare provider or institution that hosts the PHR service as illustrated by figure 5. Examples of tethered PHRs include the Veterans Health Administration's My HealtheVet, Beth Israel Deaconess Medical Center's PatientSite and United Healthcare's myUHC.com.

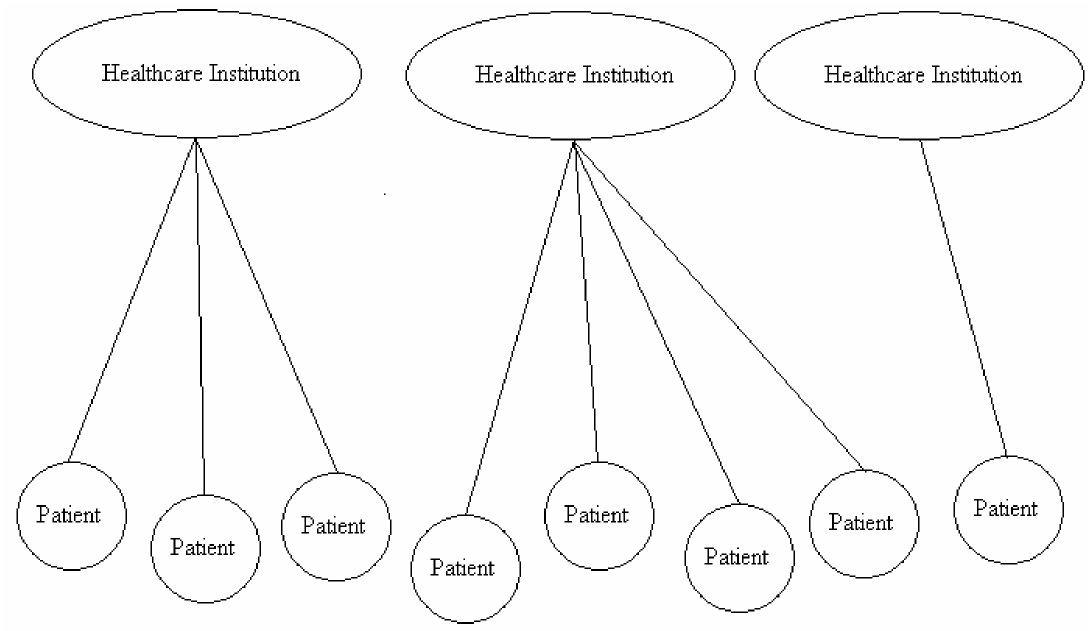

Figure 4: Tethered PHR

Interestingly, healthcare providers are not the only purveyors of tethered PHRs. Health plans (payers) such as United Healthcare and employers are also entering the PHR business. Wellpoint, for example, recognized the resilience advantages of PHRs and automatically added 18,000 affected members from the region hit by Hurricane Katrina to its PHR pilot.[45] The records stored by health plans, however, are limited in that they only store claims information rather than detailed patient records. A group of large employers including Walmart have recently announced an initiative to give their 2.5 million employees online

PHRs.[46] But employees may be hesitant to give health information to their employer for fear of discrimination. Without the participation of providers, tethered PHRs have limited value.

Tethered PHRs have significant drawbacks. Designing and operating a tethered PHR requires significant resources and they are therefore only provided by large institutions. Even if all healthcare institutions offered a tethered PHR, patients would have to manage a separate account for each institution, a logistical burden for patients with multiple providers.

The resilience scores of tethered PHRs are the same as those of a standalone PHR offered through an ASP. However, the enormous convenience and reliability advantages of tethered PHRs make them much more likely to achieve widespread adoption.

Interconnected

The holy grail of institution-patient communication is the interconnected PHR, which gathers patient data from all healthcare institutions relevant to a patient's health and integrates the data into a longitudinal record of the patient's medical history. Such a PHR would be paramount to a form of HIE. This is shown in Figure 6.

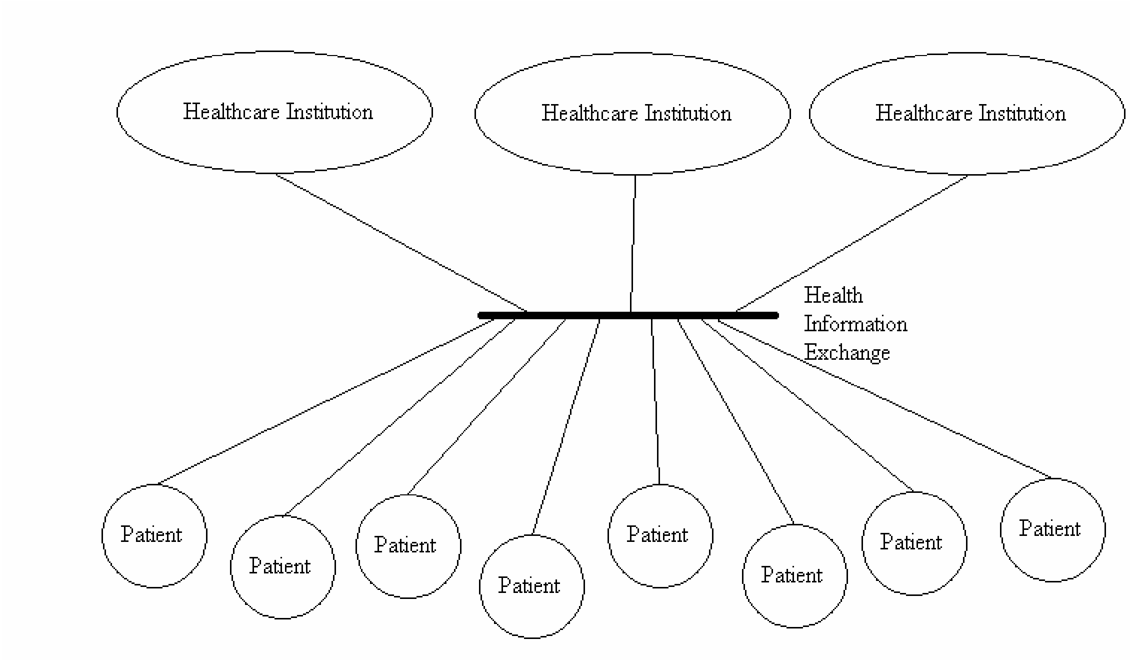

Figure 5: Interconnected PHR

The resilience scores of an interconnected PHR, architecture 8, would be the highest for every category, shown in chart 8. (This assumes a highly recoverable

PHR technical architecture). Incremental data could be captured into the interconnected PHR and advanced features would be delivered through the interface between the PHR and the provider's EMR. Today, no examples of this kind of PHR are in substantial use in the United States. One research implementation of an interconnected PHR, called Indivo, is discussed later in this thesis.

Preservability	Score (out of 10)
Recoverability of data after disruption	10
Recovery point objective (RPO)	10
Incremental data capture	10
Availability	
Delivery time to point of care	10
Deliver time of advanced features	10

Chart 8: Architecture #8 Resilience Scores with Interconnected PHR (master-slave, remote, hot, synchronous, hot backup site, interconnected PHR)

Figure 4 from above is reproduced here as figure 7 with the addition of the architectures 7 and 8. As the chart visibly demonstrates, opening the communication channels in healthcare has the potential to improve resilience.

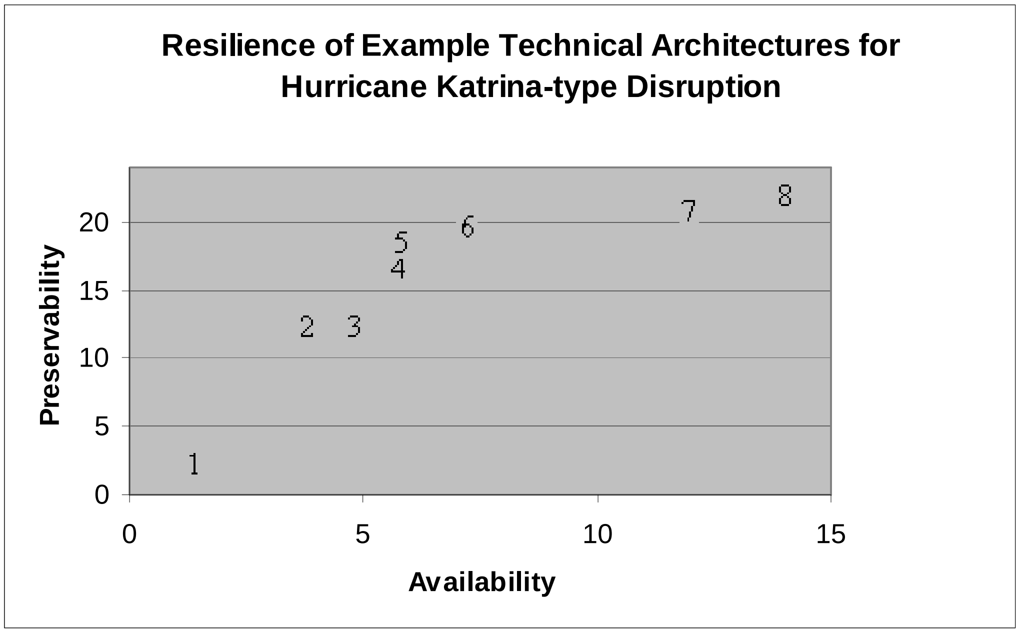

Figure 6: Resilience of Example Architectures with Opened Communication Channels

Significant improvements in availability could be obtained by simply allowing patients to gain easy access to an electronic copy of their medical record in a readable format. The more sophisticated PHRs – which would mean some form of HIE – would raise the scores of all the resilience metrics to near the maximum that could reasonably be expected. Limitations in communication channels are therefore arguably a greater bottleneck to medical record resilience than limitations in storage architectures.

Chapter 5: Current State of Resilience

The previous two chapters include some discussion of the usage intensities of various medical record technical architectures and healthcare communication channels. This chapter summarizes the current state of medical record resilience supplemented with the results of a survey of four healthcare executives from New England healthcare provider organizations and then offers some explanations for today's resilience (or lack thereof). The explanations will help inform the proposed solutions described in part 2.

A. Current Practices of Medical Record Resilience

As shown above, numerous technical architectures exist for storing and backing up medical information. Indeed, the commercial alternatives for backup storage are abundant. Of the various architectural alternatives, which are actually being used for EMRs today? Publicly available data is sparse. As stated above, anecdotal estimates say that 20% of hospitals have advanced storage architectures, but it is not clear what is meant by "advanced."[16] Legacy systems generally use tape backup – architecture 2 – perhaps involving transporting the tape to the clinicians home in the trunk of a car for storage.

Four executives of large New England healthcare provider organizations (each employing over 100 physicians) agreed to answer a survey on their institution's contingency plans. The survey questions are displayed in Appendix A. All four organizations do formal contingency planning, three of the four having used external consultants for auditing or to help design the contingency plans. Preservability and availability in these plans showed some variation.

The institution with the most advanced resilience plan makes two backup copies of all medical data, two miles and 35 miles away from the primary location. Another institution currently has only one backup center two miles away and has plans to establish another at a more remote location about 35 mile away. The remote location currently stores tape backups of EMR data. A third institution makes two local backup copies of all data and one remote copy 20 miles away. Remote backups of 20 to 35 miles may be an informal industry standard goal. These contingency plans, while probably exhibiting better recoverability than most healthcare organizations in the country, may not suffice for Hurricane Katrina-like disasters that encompass a large area. All institutions backup all parts of patient medical records.

The frequency of backup varied somewhat across the four institutions. One institution claims continuous backup to two backup sites. Another uses a combination of nightly backups and intervals of 15 minutes. The third uses a mirror site and makes daily tape backups.

The interviewees also diverged on availability for their medical record systems. One executive specified the institution's goals in terms an RTO of 15 minutes. Another claimed 12 hours, hastily adding that this was unacceptable and they have plans to improve it. A third claimed that the RTO varied from five minutes to three days. The failure-over process was a combination of automated and manual for two institutions and fully automated for a third.

For all four institutions, the EMR system came with the contingency plan integrated. (One institution uses a home-grown EMR system.)

The institutions in the survey are not typical of healthcare institutions in the country, but do give an indication of some of relatively advanced resilience practices. The dominant form of backup for medical records is the complete lack of a backup system because the vast majority of medical records are still paper-based, the architecture with lowest resilience scores. Electronic data for all but the large provider organizations will likely use primitive electronic backup architectures such as tape and, as revealed by the survey, even some large institutions still rely on tape backup. Even a completely redundant backup storage site – such as architectural example 6 above – will not alone yield high scores in availability or in incremental data loss. HIE or PHRs, which can connect displaced patients to their providers, are necessary to increase these metrics. The next section offers explanations for the current state of both storage architectures and communication channels.

B. Explanations

Absent empirical studies of resilience, the current state of resilience can be explained through qualitative reasoning. The low resilience of many providers' EMR technical architectures, complete lack of HIE, and the low numbers of PHRs integrated with EMRs have many overlapping causes. The most significant factor is the slow adoption of EMR systems by healthcare providers. EMR adoption alone, however, is not a panacea for resilience as illustrated in the previous chapters. This chapter presents technical, political, economic, industrial structure and cultural explanations for the current state of resilience.

Technical

A provider who intends to upgrade an EMR system with high-resilience storage technology will undoubtedly meet technical integration difficulties. Integration problems are not unique to healthcare; however because healthcare involves many different kinds of complex technologies with no one single company capable of making everything necessary, integration issues are especially severe in healthcare settings. This will cause providers to postpone the decision to upgrade to more resilient storage architectures and remain using tape backup or other low-resilience systems.

HIE and PHRs face an enormous technical obstacle: lack of standards for interoperability. Until recently, healthcare used an unmanageably large number of overlapping and conflicting technical formats and communication standards. (Recent developments to establish technical standards are discussed in part 2.) Without interoperability, EMRs and PHRs can only be integrated on point-to-point basis, an absurd proposition considering the number of integration projects that that would need to be performed.

Political

As argued in chapter 2, provider organizations are given huge flexibility in contingency planning under federal law and will therefore direct their efforts at business continuity rather than the policy goals of resilience. By letting the healthcare organizations determine their degree of resilience themselves without specific guidance or compliance certification, other factors will determine the degree of resilience that actually gets implemented. The lack of more strict laws governing a healthcare system that is largely shaped by government policies argues that low EMR resilience could be viewed as political or governmental failure.

Economic

Probably the most important reason for low resilience of medical records is a lack of economic incentives. If providers had strong incentives to make medical records more resilient, all other obstacles would likely be surmounted.

The economic reasons for low EMR resilience have significant commonality with the economic reasons for slow EMR adoption: inability to measure quality and externalities. For EMR adoption, because healthcare quality cannot be measured accurately, providers are not reimbursed based on the quality of healthcare delivery; rather, they are reimbursed based on fee-for-service.[2] Providers, therefore, are not compensated for the improved quality that comes with EMR usage. The fee-for-service payment mechanism and inability to measure quality therefore results in an externality: the benefits of EMRs are given to the patients (fewer unnecessary tests and procedures, possibility of alerts, ability to do

[2] Not all providers are reimbursed using the fee-for-service model. Other models such as “capitation” may reward providers more for increased quality if it results in fewer physician visits.

clinical decision support) and health plans (fewer unnecessary services for reimbursement) while the providers would have to foot the bill, not only paying the cost of the EMR itself, but also the down-time costs for learning a new system and drastically changing workflow patterns.[47] (Providers may benefit from EMRs in the long run through more effective workflow and competitive advantage in attracting patients and in hiring other providers into a practice but the initial hurdle is high.) Also, EMRs have been empirically shown to exhibit network externalities, indicating that their value increases with more providers who adopt them.[48] Providers are less motivated to adopt EMRs until enough providers already have them, reaching a “tipping point.”

Slow adoption of advanced EMR storage architectures can be explained as an externality as well: patients benefit from disaster resilient storage but providers must pay the purchase and operation costs. Also similar to EMR adoption, EMR resilience cannot currently be measured; so an optimal reimbursement mechanism would be impossible. (The resilience model in this thesis begins to address this problem.) Until measurements are established, providers will not likely be able to use resilience as a competitive weapon to attract patients. Even with resilience measurements, patients may not optimally consider resilience as a factor for choosing a provider, but they may then regret their inattention to resilience after a disaster occurs. Without the ability to use resilience to attract new patients, provider organizations will tend to invest in backup functionality that will help their organization in a practical way, for continuity of business operations during a disruption so as to retain customers. Disasters that happen frequently will be planned for. However, it is likely that most providers will not prepare for the less common – but potentially severe – disasters such as hurricanes or floods. Healthcare organizations, like all businesses, are going to try to control costs and spend only the amount of money needed for disaster recovery. "Regardless of how it is used, redundancy entails additional cost to any enterprise. It reduces efficiency and therefore is not congruent with management's goals and objectives." [49] Formally disaster planning can be expensive and may require the services of expensive consulting organizations, which is an option for large hospitals and providers organizations, but smaller providers will probably do the minimal necessary to satisfy HIPAA.

The economic reason for the lack of HIE can also be explained as an externality and as a network externality. Payers and patients would benefit from HIE because patients would receive fewer redundant tests and more informed healthcare, resulting in healthier patients and fewer doctor visits, which saves the payer reimbursement costs but eliminates provider fees. Providers will also likely resist sharing patient data in an attempt to prevent patients from easily switching to a competitor. During a disaster, HIE would increase resilience, but the original provider would not benefit because the patient would likely be using a different provider (as was the case with Hurricane Katrina). HIE exhibits a network externality because the value of exchanging data increase with the size of the sharing network. Until the network grows to a significant size, providers cannot use HIE as a competitive weapon because the value would not be high enough.

PHRs also exhibit an externality where patients benefit but providers have to pay as well as a network externality similar to HIE. Patients would undoubtedly have better care from PHR by being able to more carefully manage their health information and having access to that information during a disaster. Providing high availability PHR services would require technical expertise and would be expensive, as described previously, which would make PHRs affordable only to the larger providers. It is conceivable that providers would charge patients for using PHRs but the business case has not been made that shows that patients are willing to pay for PHRs. Especially if the PHR was tethered to a provider, patients would be very skeptical of paying a PHR fee for every provider they visit. An interconnected PHR system would have a network externality, resulting in slowing the adoption until it reaches a critical mass.

Industrial Structure

Another reason for low resilience is the fractured nature of the US healthcare system. "The health care 'system' in America is not a system. It's a disconnected collection of large and small medical businesses, health care professionals, treatment centers, hospitals, and all who provide support for them."[50] With so many different healthcare institutions operating independently, establishing sensible and consistent policies for the entire healthcare system is challenging if not impossible. It might be easier to improve the resilience of a more tightly coordinated or centralized healthcare system. As it is, getting providers to adopt EMR systems with resilient storage architectures, integrating their systems with other organizations for HIE, and supplying highly-resilience PHRs for all patients requires a formidable amount of coordination to have it done with any degree of efficiency. The US government, even though it plays a huge role in healthcare, paying for approximately 40% of healthcare bills, has not effectively used its power as a payer to increase EMR resilience. [51]

The fractured and decentralized nature of the US healthcare "system" inhibits collective action that might overcome the externalities. For example, one might suggest that the payer could buy PHRs for the providers as a Coasian bribe since the payers might benefit from lower cost of healthcare during a disaster.[52] Similarly, the payers could potentially build a network for HIE and supply highly resilient storage architectures. Why has this not already been done? It can be explained, in part, as the result of yet another network externality, among payers.[53] If a payer funded the construction of an HIE network or purchased advanced storage architectures for a provider, competing payers would be able to free ride off the donor payer’s gift. But it is in the interests of all of the payers to establish the HIE network. This is a classic Olsonian collective action problem: payers are not concentrated enough to coordinate so tightly.[54] Some insurance companies have shown an ability to coordinate and to pay for EMRs and HIE in New England.[55] But while we are waiting for the payers to organize, resilience lags. Patients could also potentially coordinate and solve this problem themselves, but the chances of patients – a tragically diffuse stakeholder – overcoming a mammoth collective action problem are very small.

Cultural

Cultural attributes might also contribute to the low levels of EMR resilience. Specifically, public perception of centralized storage, public expectations of medical record resilience, and the US healthcare provider culture of autonomy may impede progress toward resilience.

As stated previously, centralized storage evokes public fears of privacy vulnerabilities. This concern may prevent companies from taking advantage of economies of scale in building large storage facilities and collaborating. Large companies that specialize in backing up medical records might provoke these fears, but at least one is trying to offer this service.[56]

Because extracting medical data from a healthcare institution is so onerous if not impossible, public expectations for resilience – especially availability – may be comparably low, resulting in little pressure on providers to increase resilience. These expectations may rise as the public becomes more accustomed to the technological capabilities of electronic systems and with the occurrence of large disasters such as Hurricane Katrina that provide stark reminders of the importance of resilience. More empirical evidence of the negative health effects caused by low resilience would help raise public expectation and perhaps lead to political attention and policy changes.

US healthcare providers view themselves as autonomous practices and therefore may be hesitant to cooperate on HIE projects. If an integration project involves surrendering some control over their organizational operations, they may view it as infringing upon their autonomy.

These are some of the possible explanations for the current state of resilience. The forces described may change, motivating an increase in medical record resilience. In the meantime, mediocre resilience – mediocre availability scores, in particular – will remain.

PART 2: Improving Resilience

Having clear policy goals and a rough measure of the current state of resilience relative to those goals is an important part of understanding how to make medical records more resilient. The next step, discussed in part 2, makes specific recommendations through strategy, technology and policy. Chapter 6 describes existing efforts to bypass the largest blockage to medical record resilience today: lack of open communication channels in healthcare. It then offers the current initiatives a strategic solution to help accelerate the opening of these communication channels. Entailed by some of the current HIE organizational designs is the problem of centralized medical record storage, which is investigated in Chapter 7. Distributed storage technology is explored as a solution to this problem. Chapter 8 looks beyond current initiatives to new public policy options for regulating an increase in resilience above current requirements and the factors that should influence such changes in policy. Finally, chapter 9 summarizes and concludes.

Chapter 6: Opening the Communications Channels

As shown in part 1, the path toward higher medical record resilience – for Hurricane Katrina-like disasters at least – must involve bettering communication channels among healthcare institutions and between the institutions and patients. The ideal for both institution-institution communication and institution-patient communication involves interconnecting and integrating with provider-generated EMR data, which implies some form of HIE. Since its inception, ONC has emphasized institution-institution communication as the highest priority, but recently the politically-appointed coordinator, Robert Kolodner, has placed PHRs as a top priority, claiming that "most Americans will have EHRs by 2014, and personal health records will drive that effort." [57] This chapter describes current efforts to open the communication channels of healthcare and explores one of its formidable challenges: establishing a sustainable business model for HIE organizations. A strategic recommendation for establishing sustainable business models for these organizations is then presented.

A. Stirrings of HIE

Current efforts at realizing HIE hope to avoid the fate of previous efforts to accomplish the same goal. The 1980s and 1990s saw a movement to establish community health information networks (CHIN) with the purpose of exchanging health information between institutions. The failure of the CHIN movement is still heavy on the minds of many leading figures in present-day HIE initiatives.

The establishment of the ONC in 2004 elevated HIE and other health IT objectives to national priority. ONC has created four organizations as part of the initiative: Health IT Standards Panel (HITSP), Certification Commission on Health IT (CCHIT), Health Information Security and Privacy Collaboration (HISPC), and the National Health Information Network (NHIN). Briefly, the work of each is as follows. HITSP is a standards development organization (SDO) that convenes numerous stakeholders who have an interest in EMR interoperability and creates standards out of the numerous overlapping standards that already exist. CCHIT certifies EMR products based on various criterion including standards approved by HITSP. HISPC is the newest of the four and works on privacy and security issues. Finally, the NHIN aims to create a national system of HIE. The NHIN has four ongoing pilot projects and hopes to glean from these projects valuable insights as to how medical records might be exchanged at a national scale. All of these organizations answer to a federal advisory board called the American Health Information Community (AHIC) which has monthly meetings at DHHS, often attended and led by the Secretary himself, Mike Leavitt.

Often discussed when talking about the NHIN are Regional Health Information Organizations (RHIOs). RHIOs, stated simply, are organizations that implement HIE. Currently, numerous RHIOs across the country exist in various stages of development.[58] Skeptics will point out that the RHIO movement in many ways resembles the failed CHINs movement of the 1980s. Indeed, RHIOs face many obstacles and difficult choices on their road to HIE. Recently, some luminaries have suggested that many RHIOs will go out of business because of the lack of a sustainable business model. [59, 60] Some have speculated that a national network may rise to do HIE, but nothing resembling such as network currently exists nor are there even preliminary plans for one. Today, RHIOs are the best hope for HIE.

HIE could be implemented by creating an interconnected PHR system, as discussed in part 1. One research project at Children Hospital in Boston called Indivo (formerly PING) has an open source PHR prototype that is designed to integrate with any EMR system via subscription agents.[61] Indivo gives the patient a single point of access to her medical record and allows the patient fine-grain control of access permissions. It is based on open standards and software, which supports long-term survivability of the data. It uses cryptographic methods to assure that only authenticated and authorized users can access or modify records and cryptographic signatures to assure that data are never decoupled from their provenance. It also incorporates logging and auditing, and thus provides direct accountability to the patient. Indivo's working prototype is in beta version and is expected to be piloted in the near future. It is possible that a RHIO or RHIO-type organization might use a model similar to Indivo to host a PHR. Indivo also allows data to be exchanged among healthcare institutions by letting the patient directly move it: the cryptographic technologies that Indivo uses would protect the integrity of the data while enabling patients to hide parts of their medical history when transferring data between providers. This would require providers' EMR systems to use a standardized authentication method but would not require RHIOs to transfer or store the data thereby possibly easing the sustainability issues that RHIOs face. Widespread adoption of an authentication standard, however, is no small feat.

The next section of this chapter investigates some proposed RHIO business models to overcome the sustainability hurdles.

B. RHIO Business Models

Numerous proposals for RHIO business models hope to achieve sustainability. It may be that unique stakeholder environments in many localities will dictate disparate requirements for a local RHIO. Regional laws, concentrations of Health Plans and healthcare provider markets and local cultural preferences may all influence the way a region can achieve HIE. I discuss three proposed RHIO business models: health data banks, infomediaries and charity-based.

Health Data Banks

Pioneered and promoted by William Yasnoff, this business model analogizes RHIOs to the financial banking system so that doctors would receive payments for making “deposits” of information into the RHIO and patient would pay for the “withdrawal” of information or pay a monthly subscription fee.[62] The elegance of this model is that it uses Coasian principles of the free market to internalize the externalities. With health data banks, patients can look at the data directly, share it with other providers, or make it part of a personal health record. The fee might be paid by the patients directly or by the patients’ health plans or employers. Yasnoff hopes the ability of providers to get paid for releasing the information electronically will also encourage EMR adoption. Details of the payment mechanism would have to be designed and could prove to be insurmountable obstacles. Washington State has recently adopted this approach.[63] If it proves successful, it may serve as a model for other regions.

A key component of this business model is that patients would have access to their data which they could potentially store in a PHR or with an organization that provides such services. Patients who use RHIOs to extract their medical data into PHR provider organizations would have a veritable backup of their data, even if the RHIO does not store a copy. (The question of RHIO storage is addressed in the next chapter.) The PHR organizations will likely have higher resilience for their customers’ data because they are directly accountable to the patient for giving access to the patient’s medical data, even during an emergency. It is not clear how many patients would use PHRs, but RHIO business models such as this one that are oriented toward getting the data into direct contact with patients generally yield better resilience. (There are certainly many more reasons besides resilience for giving patients access to their own data.)

Infomediary

A recent article lays out a vision of RHIOs to serve as infomediaries between the numerous stakeholders.[64] An infomediary is an organization that manages information that is used by different stakeholders as an intermediary. The term is taken from ecommerce applications where it is used in examples such as entities that gather information about products for consumers (aggregators) or that gather information about consumers for marketing purposes.[65]

In an infomediary business model, RHIOs would sell data to drug companies and to medical devices companies who would use the data to monitor the effectiveness of their products. The RHIO might also facilitate providers in submitting reporting requirements to government agencies and health plans. It is unclear if providers should pay, get paid, or have the service for free. Under this model, the patients may have to pay directly to access their data or indirectly through providers of PHRs that would connect to the RHIO. Payers might be willing to fund much of the operation if better health outcomes can be demonstrated, saving them money in claims fees. Many variations of this business model are possible. In a way, the infomediary business model is just a generalized data health banks model discussed above in that it also fundamentally relies on Coasian exchanges where each party would pay or get paid according to the direction of value flow. The infomediary model is reproduced visually in figure 8. [64]

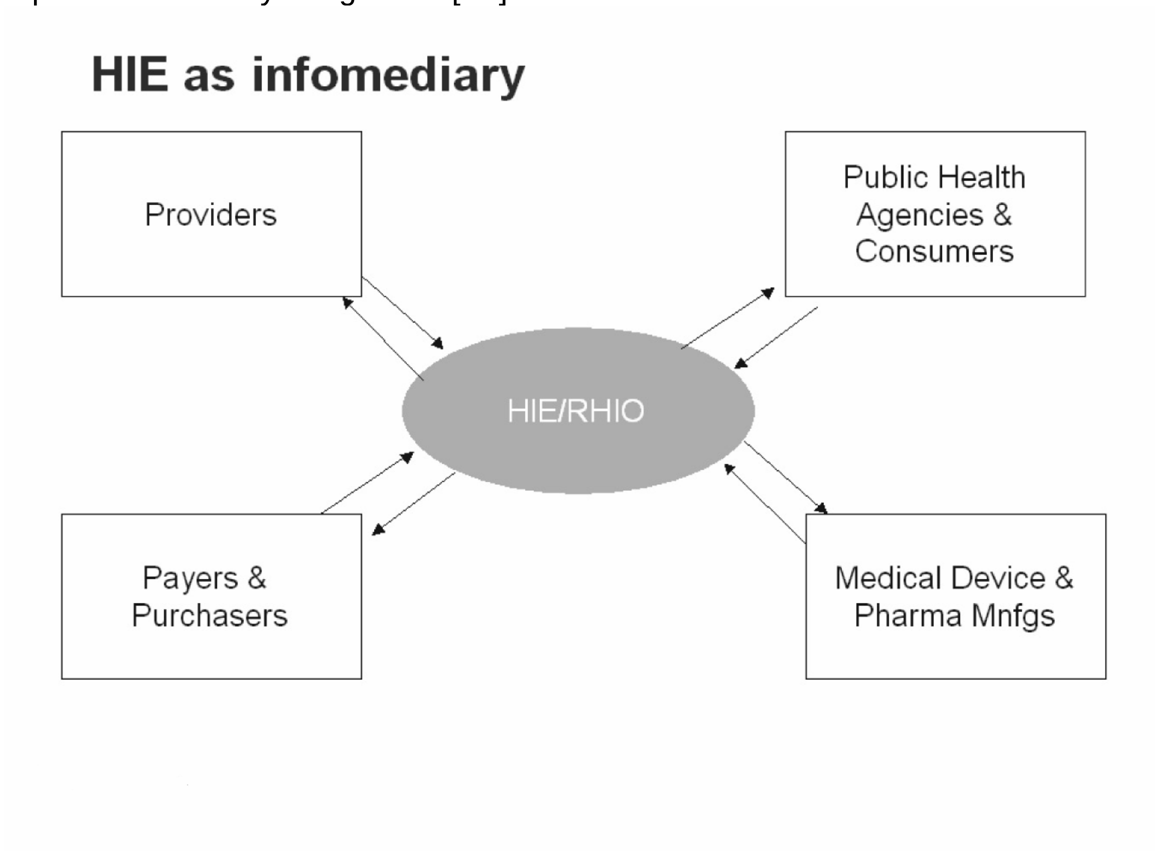
HIE as infomediary
Providers
Public Health Agencies & Consumers
HIE/RHIO
Payers & Purchasers
Medical Device & Pharma Mnfgs

Figure 7: Infomediary RHIO business model

One enormous hurdle for this business model is coordinating all the important stakeholders. Stakeholders that benefit from this RHIO model but refuse to participate will be free-riding and might derail the entire initiative. This is yet another example of an Olsonian collective action problem and might be surmountable only in special cases where enough of the stakeholders are concentrated and their interests are aligned. Perhaps the most advanced initiative along these lines is the Massachusetts eHealth Collaborative in which a large local payer has agreed to fund HIE between three communities to evaluate whether better health outcomes (and therefore lower claims) will result. The project involves donating EMRs to every provider in three Massachusetts communities.[66] Overcoming the collective action problems might be more difficult in other States that have more diffuse interests, especially if the payers and provider organizations are accountable to central organizations outside of the local regions.

Charity-based

Another option for RHIOs is to not even try to form sustainable business models but rather rely on grants. A white paper from the HIT Transition group makes this case forcefully: “RHIOs are charitable organizations which serve the public good, and RHIO leadership can accelerate their development by embracing the identity and operational behaviors consistent with established and proven charitably-supported NPO (non-profit organization) models.”[58] The white paper further points out that non-profit organizations in healthcare operate much like commercial businesses and do not rely to a large extent on contributed income, making them different from most non-profits in other industries such as the American Red Cross, the United Way and the YMCA, which do rely on grants and donations. RHIOs should devote more energy toward fund raising, the paper says, which they can do strategically by using the donations for capital-intensive infrastructure projects to show donors something tangible; operating expenses can rely on subscription fees.

The white paper argues that times have changed since the CHINs’ debacle and that there is growing public awareness of the importance of HIE and therefore the public would be willing to sustain HIE indefinitely as a public good funded through grants.

C. RHIOs and Disaster management

All proposed business models (and almost all regional initiative) have significant challenges. Many RHIOs may dissolve as their grant money runs out, just like the Santa Barbara County Care Data Exchange, an 8 year and over $10 million project. If RHIOs could offer services other than HIE that would leverage HIE-related investments, perhaps a sustainable business model might be easier to establish. Michael Porter, the famous Harvard Business School professor, argues that offering new products and services is one of the five forces that contribute to profitability.[67] Even a charity-based business model would benefit from an added revenue stream. An extra service that RHIOs might offer local healthcare providers could be disaster management such as providing best practice advice, consulting, or storing the provider’s medical data in a disaster resilient manner.

There are many reasons that disaster management might be a suitable service for RHIOs to offer. First, both HIE and disaster management involve managing medical data according the laws of a specific region. RHIOs, because they will inevitably have to implement systems that obey the regional security and privacy laws, would be in a position to give legal planning advice for providers who are creating backup solutions and have to obey the same laws.

Second, since the data that are exchanged by RHIOs and the data that are required during a disaster are the same data (i.e. patient medical information), additional services for that data can come relatively cheaply. Once a provider's EMR system is integrated with a RHIO, allowing the RHIO to perform an additional backup service would have low marginal cost compared to making a separate full-fledged backup system to store the same data.

Third, once a relationship between a RHIO and provider is established, the "relationship cost" of adding a new service such as disaster management is small, whereas "locating and maintaining a relationship with a second, new customer is much more expensive." [58] RHIO should take advantage of this efficiency and offer a "two for the price of one" deal.

Fourth, RHIOs could ostensibly solve an information asymmetry problem among providers who are at a disadvantage: they do not know which disaster management consulting company to believe. RHIOs could therefore help providers discern which advice is closer to industry best practices for their specific region's degree of risk and avoid spending more than is necessary, lending credibility to the practice of disaster management and more efficiently allocating investments.

Fifth, the RHIO might store all the data centrally, relieving the provider of all storage costs just as some online word processing providers store all the data on their company servers. Alternatively, RHIOs could store only a backup copy and serve as hot sites for all local providers. In the survey administered to four healthcare executives, each was asked "Would you consider participating in a collaborative effort with other local healthcare delivery organizations to coordinate practices and resources for contingency planning and operations?" All four of the executives surveyed answered yes to this question. One of them, a CIO of a large New England healthcare organization, wrote "Willing to discuss creating shared hot sites." Another executive was already engaged in conversation with a different hospital for sharing backup sites. (The issue of RHIO storage is fundamental and discussed more extensively in the next section.)

Sixth, if RHIOs catch on, patients will naturally look to the RHIOs during a disaster to find their medical information because RHIOs provide access to their data from outside their usual healthcare institutions. RHIOs will therefore be expected by the public to have planned for disasters. They could leverage their expertise in this area by offering similar services to providers or doing the disaster management for the providers.

Seventh, both resilience and HIE are public goals that are not valued by the market as much as by the public. The additional service of disaster management might make patients or employers more likely to pay for the data and donors more willing to contribute depending on which business model is used.

Eighth, and finally, disaster planning might serve as a good excuse for providers to come together for discussion on forming a RHIO. One HIE luminary suggested that a big hurdle for RHIO formation is simply that provider organizations are not accustomed to working together; rather they view each other as competitors. Resilience planning might be a first step for organizations to work collectively and, as the luminary put it, "do laundry together."[68]

Resilience planning and HIE can therefore be synergistic; RHIOs should exploit this synergy to their strategic advantage. One healthcare executive supported this strategic argument, saying that having RHIOs do disaster management was “right on the mark.” RHIOs might start helping providers by recommending disaster management consulting companies or publishing some information regarding best practices for participating members. As trust between the RHIOs and providers builds, the RHIOs may consider providing more advanced resilience services such as storage services for the providers. Government regulations could also potentially facilitate the development of RHIOs as providers of disaster management services in addition to HIE, which is discussed in chapter 8.

Offering disaster management services is a possible solution to the lack of medical record resilience because offering these services would help RHIOs establish more sustainable business models and therefore help open the communication channels in healthcare. Sustainable business models are not, however, the only impediment that RHIOs face. Public concerns with centralized storage of health data also put significant restrictions on RHIOs. The next chapter investigates RHIO storage alternatives and presents a possible technology solution to address the public concerns over privacy of centralized electronic data storage.

Chapter 7: RHIO Storage and Distributed Hash Tables

A fundamental design choice for RHIOs is the location of the data. The possibilities fall into two general categories: centralized and decentralized. This chapter evaluates both possibilities, arguing that centralized is currently preferable but that it is hobbled by privacy concerns. The chapter then presents experimental results of a potential technology solution, Distributed Hash Tables (DHT). This is an experimental software mechanism that would allow data to be easily administered by a centralized organization such as a RHIO but stored in a decentralized manner. The chapter includes a background of DHT technology; a background of the medical record project Indivo (formerly PING) that is used in the experiment and how Indivo might be used by a RHIO; the experimental setup and results of using a DHT implementation as the storage mechanism for medical records; and an evaluation of the experimental results as applied to the use of DHT technology as a mechanism for medical record storage.

A. RHIO Storage

RHIOs must decide whether data will be centralized or decentralized. Centralized storage means that participating providers store their data at the RHIO's location in one specific place (and presumably at least one backup location). The famous RHIO Indiana Health Information Exchange uses centralized storage but the data is logically isolated in "vaults" that keep each provider's data separate.[69] A decentralized storage approach generally means than each provider will store data in a server that they maintain themselves but is accessible by other providers either directly or via centralized RHIO servers that coordinate the information exchanges but does not store medical data. This approach is described by the Markle foundation's "Connecting for Health" project.[70] The decision of centralized versus decentralized storage involves important trade-offs.

Centralized storage in a RHIO would allow for high resilience because all the data would be stored in a consistent, disaster resilient manner. Providers would potentially be able to use the RHIO's data storage as their contingency plan. Using centralized storage would therefore facilitate RHIOs in offering disaster management services and therefore – as argued in the previous chapter – achieving sustainable business models. Centralized storage has the additional advantages of being generally easy to manage.

For decentralized storage, the RHIO would not be able to provide backup storage services because it would not be storing the data in one place. Providers would then still need to make their own contingency plans, which they would probably be slow to implement as argued in part 1. Therefore, a decentralized approach is generally worse for resilience compared with a centralized approach. Also, decentralized storage would probably be more expensive in total and harder to manage than a centralized approach because of its increased complexity and the requirement of every provider to host a server.

A centralized approach is generally preferable from the perspectives of resilience and practicality; however, it has the disadvantage of raising privacy concerns.[3] Massachusetts, for example, follows an explicitly decentralized approach. Public fears may not allow the centralization to happen on a large scale for many regions. Public fears regarding privacy may be assuaged over time as some pilot projects are demonstrated to be functional and secure; however, this change might take many years. One way to overcome the public fear of centralized storage is to centralize only the medical data from patients who give their explicit permission to use centralized storage. But that way, providers would still need to manage backup services for those patients who do not agree to centralized storage. Also, if the public is truly concerned with privacy, many patients will hesitate to give permission to store their medical data in a large centralized repository.

It is important to separate perceived privacy vulnerabilities from real privacy vulnerabilities. A centralized repository may have better privacy safeguards than a decentralized repository because of factors beyond storage location, such as software design, encryption, authentication architecture and physical protection of the storage media. The location of the data is thus one of many factors that determine its vulnerability, but it is perhaps the factor that is most visible to the public. However, for a large-scale project to receive public support, it must be attractive to public perceptions in addition to demonstrating resistance to failure.

Because centralized storage is a more viable option – perhaps the *only* viable option – for RHIOs compared with decentralized storage, privacy concerns with centralized storage will likely continue to impede the resilience possibilities offered by RHIOs. A solution to this problem may be found in Distributed Hash Table technology.

[3] In addition to privacy concerns, some providers may fear giving up their data to a centralized location that was not under their control; however relieving the providers of the burden of operating a server and managing a backup site might persuade them to participate.

B. DHT Background

DHTs provide the underlying technology for a distributed file system. The core advance of DHTs is the ability to find specific data items quickly within a network given an identifier.[71] This advance allows data to be strewn across multiple computers that consist of nodes of the DHT and then recovered in a time-efficient manner. A DHT-based file system consists of a collection of these nodes which are configured to operate as one unified file system. The nodes store copies of the data and interact with one another to keep track of the data's location. Each node serves as an access point for an application program to read or write data into the file system. DHTs emphasize the following properties: decentralization, scalability and fault tolerance.[72] With these properties, DHTs were originally designed to be the basis of an unerasable publishing medium. The requirements of medical data – which is rarely deleted – are quite similar.

Decentralization in this context refers to the network of nodes that maintain the data. These nodes can be in any geographical location that is connected to the Internet, although network latencies will significantly affect data access times. The physical location of these nodes will affect the resilience scores of the storage architecture, with widely dispersed nodes generally achieving higher resilience. The specific relationship between node topology and resilience would be an interesting area for future research.

Scalability refers to the ability of the network to expand and still function effectively. DHTs – born out the peer-to-peer research – can be expanded with commodity hardware: simply adding a new computer to the network effectively increases the network's storage capacity, avoiding complex configuration changes.

Fault tolerance means than in the event of one or more node failures, the network can recover. By maintaining replicas of the data at various nodes, DHTs have redundancy built-in. For some DHT implementations, the degree of redundancy can be adjusted through configurations settings. The technical advantage of DHTs is that when a node fails, the software will automatically detect the failure and reestablish the desired degree of redundancy among the remaining nodes. Therefore, data loss will only occur if a sufficient number of nodes fail nearly simultaneously, and one can achieve lower likelihoods of that occurring by investing in greater redundancy.

DHT-base files systems can be implemented with a technique called erasure codes where a block of data is encoded into k fragments, of which only some subset m, are needed to reassemble the original data.[73] When k equals m, there is no redundancy because all fragments must be retrieved to reassemble the original data. The level of redundancy increases as the ratio of k to m increases. However, with increased redundancy, the size of the fragments also increases, requiring more disk space. For example, if k and m are set to 10 and 5, respectively, and the size of the original data is 100 bytes, there would be 10 fragments, each 20 bytes in length and any 5 of the 10 fragments would be required to reassemble the original data. Notice that the k/m ratio is 2 (10 divided by 5) and that the total number of bytes is 200 (that is, 20 times 10), which is twice the original size of the data. Erasures code greatly increase fault tolerance for a given level of redundancy above a simple copy. In this example, even if up to 5 fragments were unavailable due to node failures, the original data could still be reassembled from the remaining fragments. Without DHTs, if the 100 bytes were merely copied to a backup location, failures in only two nodes (the primary and the backup) would render the data irretrievable. Erasure codes therefore provide the potential for higher fault tolerance per disk space used.[74, 75]

Qualitatively, DHT technology appears to fit the requirements of RHIOs for storing large amounts of medical records if the RHIO maintains a distributed collection of computers hosting the DHT. [4] Decentralization, scalability and fault tolerance are all desired qualities for EMR storage. However, the performance requirements of medical record storage and retrieval have not been tested empirically against DHT implementations. Without reasonable performance, DHTs would be useless in a healthcare environment. The next section presents background information about a medical record project that was used to experimentally test EMR storage performance on a DHT-based file system.

C. Indivo Background

The open source software project Indivo, described previously, was selected to test DHT performance. Indivo is one architecture that would be one possible for RHIOs to use as the basis of HIE. By replacing Indivo's underlying data store with a DHT-based file system, we investigate DHT technology's potential advantage of providing increased resilience.

[4] While DHTs were originally designed for peer-to-peer applications, a cooperative file system where each node was maintained by a different provider institution would probably be rejected because it could not easily be held accountable. RHIOs, on the other hand, would be a natural fit for hosting a DHT-based file system even though it would not make use of all the peer-to-peer optimizations in DHTs.

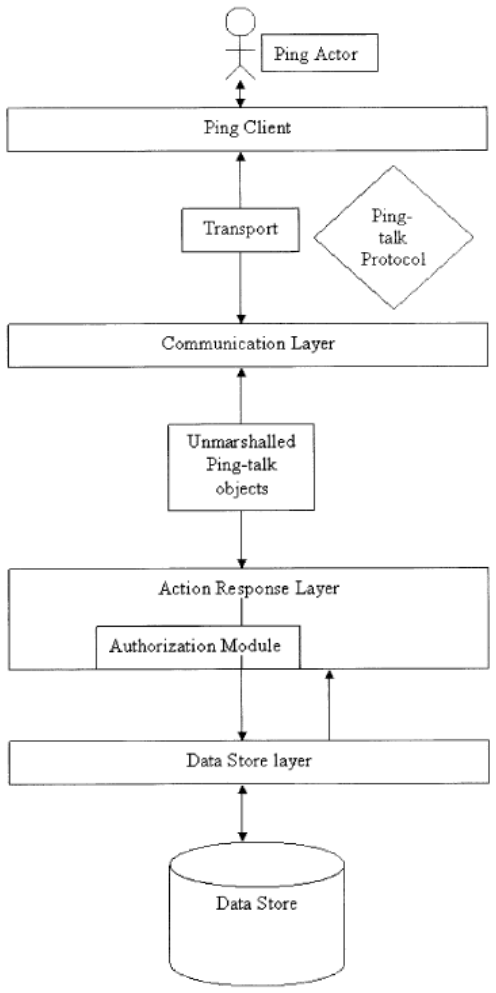

Figure 8: Indivo (formerly PING) Architecture Layers

The high-level architecture is visually presented in Figure 9.[61] The Ping Actor as shown could be a patient, physician or other member of the healthcare system. The Ping Actor is given access to medical data based on access permissions. A patient could then, for example, give her physician access to some or all of her medical data or transfer data between medical laboratories, pharmacies and other healthcare institutions. Assuming the physician and other sources of healthcare information participate in this system by importing new patient data, this would be tantamount to HIE, integrating all the actors in the healthcare system under a unified system.

Particularly relevant to resilience is Indivo's data storage layer. The current Indivo system is designed for centralized storage. This means that all medical data relevant to a particular Ping Actor is stored in one location that would, perhaps, be administered by a RHIO. (Other organizations could host an Indivo server for use within the organization.) Indivo's prototype implementation makes use of a transactional open source software package called Berkeley DB. Because Indivo is a prototype project, a professional contingency plan has not been designed for it, but software that makes one or more backup copies of the data has been partially implemented. This backup software is asynchronous i.e. backing up occurs at regular intervals rather than as data is written. Indivo currently does not support synchronous backup functionality.

Indivo stores all medical records including binary records in XML format. Large files such as medical images are stored as one large XML file, a design that does not scale well for increasing file sizes and number of users because the XML parser puts the entire file into memory. However, Indivo's modular design allows portions of a patient record to be accessed without fetching the entire record, mitigating some of the performance issues.

D. Medical Record Storage Performance on DHT Experiment

The storing of medical data has unique application characteristics particularly with regard to sophisticated access control features and audit trail requirements that will result in unique performance requirements. The goal of this experiment is to evaluate performance and resilience qualities of DHT technology for storing medical data. A research project called DHash was selected to provide the DHT implementation.[73] For the medical record application, the project mentioned earlier, Indivo was used.[61] The experimental setup and results are described below.

Experimental Setup

Indivo and DHash both provide interfaces that would allow them to be integrated with each other; however, the interfaces require customized code to make them compatible. New software, called hearingaid, was created to translate between the interfaces of the two projects. Indivo was written in Java while DHash was written in C++. The hearingaid code makes use of the Java Native Interface (JNI) to translate between Java and C++. The hearingaid code is shown in appendix B.

Indivo is modularized so that all accesses to the storage layer are performed in one Java class. A custom storage class was written to direct all storage accesses to the hearingaid layer. Indivo expects the storage layer to accept any arbitrary key-value pairs, as shown in the following read and write methods:

```
write (key, data_value);
data_value read (key);
```

The length of the keys and values are determined by Indivo.

DHash has more restrictions than Indivo. DHash is optimized for smaller chunks of data on the order of 16 kilobytes (KB). Also, DHash uses keys that are fixed at 20 bytes in length. More significantly, DHash uses asynchronous communication for its read and write methods. That is, when a read or write method call finishes, DHash will return the result of the method call in a separate thread. It was implemented in this manner to allow large numbers of reads and writes to occur in parallel. DHash surfaces the following methods:

```
insert (key, data_value);
data_value retrieve (key);
```

The hearingaid code accepts read and write requests from Invido, converts all keys to 20 bytes in length, and translates the calls into DHash method calls. To translate the write requests, hearingaid breaks the data to be written into 16 kilobytes or smaller chunks, stores each chunk into DHash while saving the keys of these chunks in an intermediary block. It then stores the intermediary block under the desired key specified by Indivo. For reads, hearingaid does everything in reverse: it reads the intermediary block of keys, fetches the 16 KB chunks for each key, and assembles the original data block for Indivo. Through experimentation, it was found that Indivo would often write data to DHash and then read it right away expecting to get the same value, but DHash sometimes returned a previous value that was written to that particular key because of its asynchronous operation. To correct for this problem, code was added to repeatedly read data from the key to which data was just previous written until the expected result was returned. Unfortunately, this has obvious deleterious effects on performance.

Indivo supports both encryption and transactions however both were disabled for this experiment. Transactions are not supported by DHash, a significant limitation discussed more below.

This experiment tested, first, performance and, second, redundancy. For the performance test, the control case was to use Indivo with the storage software that it supports: Berkeley DB. The experimental case swapped out Berkeley DB for DHash nodes. The control and experimental cases are illustrated in figure 10 and 11 respectively.

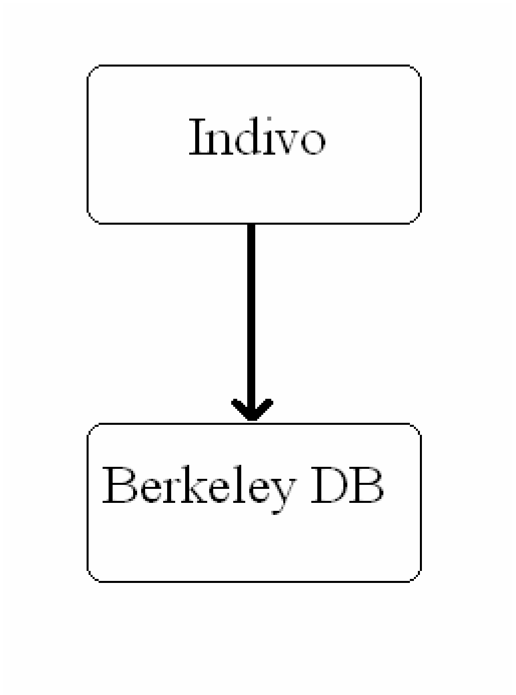

Figure 9: DHT Control Case: store to Berkeley DB

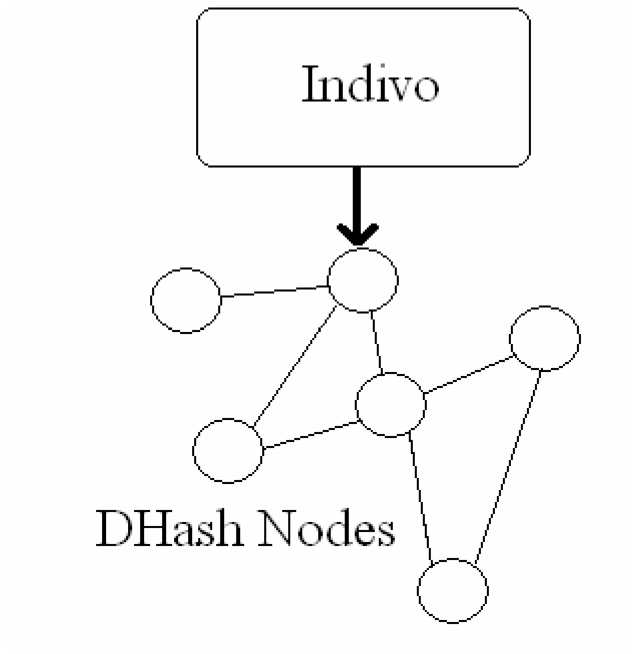

Figure 10: DHT Experimental case

For each scenario, sample medical records consisting of binary data converted to ASCII text in an XML file were written and then read back, recording the amount of time for the complete operation to occur which included authentication of a sample patient. The sample sizes were: 1.2 KB, 539 KB, 6.0 MB, 10.1 MB, and 16.1 MB. When some larger file sizes were attempted, Indivo's server ran out of memory. For the DHash operations, the experiment was performed with 5 nodes and with 18 nodes. Replication values (i.e. erasure code settings) of 6 and 3 were used for k and m respectively. While testing, it was found that the implementation of DHash occasionally exhibited unstable behavior. Only small files could be added for the 18 node experiment before the software crashed. It was found that DHash was more stable when multiple "virtual nodes" were used at each computer in the DHash network. Therefore, for the 5 node case, virtual nodes were used at the five computers in the following degree: 15, 5, 5, 5, and 3. The 18 node case involved adding 13 more nodes with 3 virtual nodes each. Also, a stable testing environment could not be found for testing the nodes in a geographically distributed setup, so all nodes were within a local area network. In a realistic case where the nodes are distributed around a region or country, the performance results will be significantly worse than the results obtained here depending on the network latencies. However, the control case for geographically distributed nodes would arguably require a synchronous copy made to a remote machine and would therefore also have long latencies. An exploratory test was performed with a geographically distributed network of computers using the Planetlab platform.[76]

For the redundancy test, the computer with 3 virtual nodes was removed from the 5 node experimental case and the file of size 6.0 MB was read from the remaining nodes.

Results

The results of the performance tests are shown in figure 12. Because only small files were testable on the 18 node DHT test, the subset consisting of just the smaller files are put in figure 13 to allow better visibility. It is apparent that the time for writing and reading both scale roughly linearly with the file size of the sample file. The control case exhibits reads and writes taking approximately the same amount of time. For the DHash test with 5 nodes, the writes take roughly twice as long as reads. This might be partly because of the time it takes to make the extra redundancy, which does not occur in the control case as Indivo does not currently support synchronous backup to remote locations. It also might be due to the extra read operation is performed after every write operation to ensure that the data is written correctly. For larger files, writes to DHash roughly approached 0.25 MB/second and reads approached 0.5 MB/seconds compared with 2 MB/second for both reads and writes to the control case. Therefore DHash is slower than the control case by factors of roughly 4 and 8 for reads and writes respectively. The experimental case with 18 nodes showed somewhat slower read and write times for the few data points that were recorded. This makes sense because more nodes will take longer to find the location of each data storage point. It is a bit puzzling that for the 18 node case some of the reads take longer than the writes. A more thorough and rigorous test with a more stable version of DHash would have to be performed to better understand why this happens.

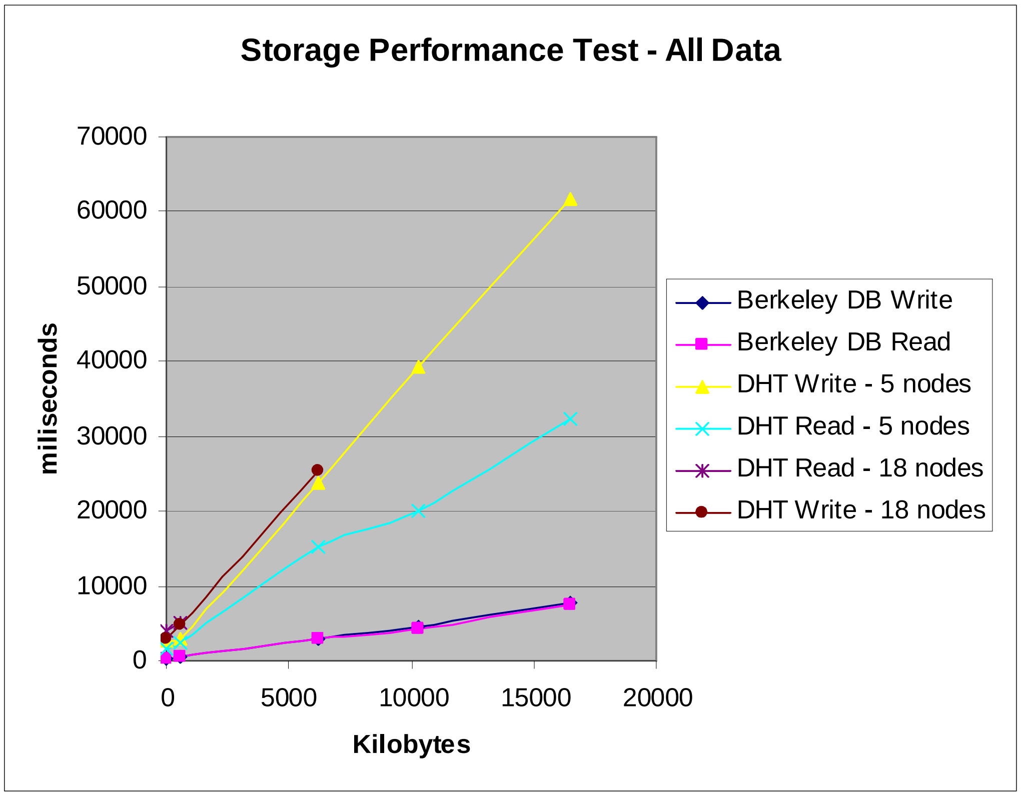

Figure 11: Storage Performance test results

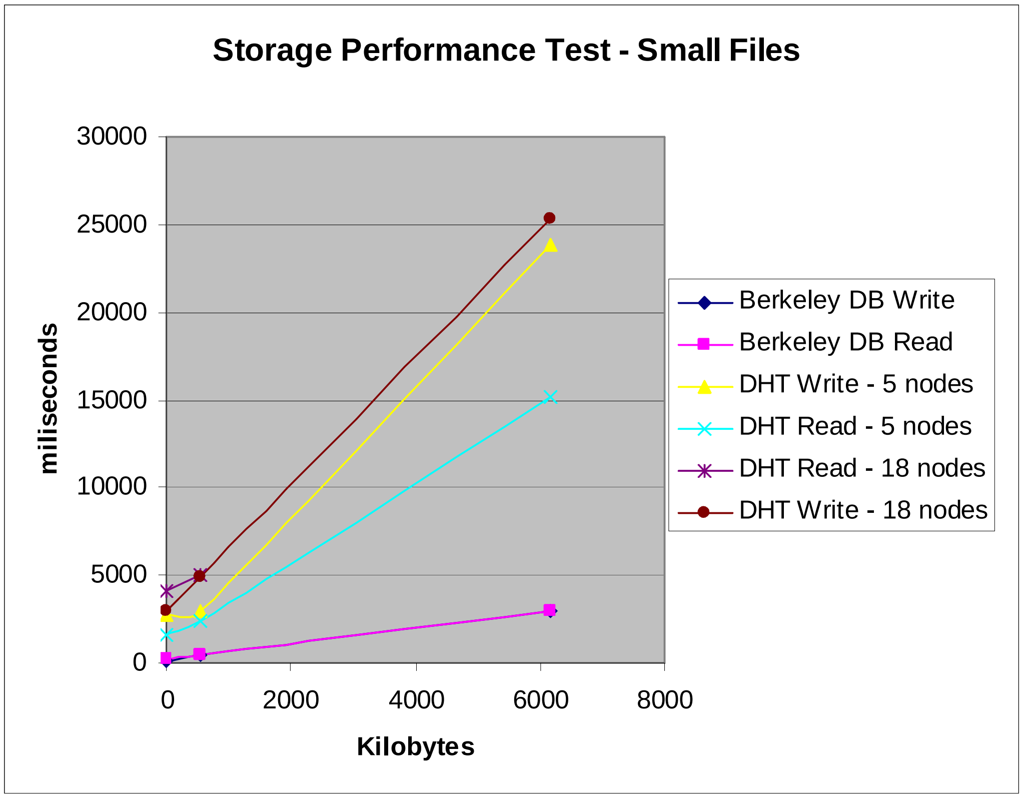

Figure 12: Storage Performance test results for smaller files.

A significant factor of performance which was not explicitly taken into consideration is the network bandwidth. Each node's network bandwidth could affect the performance of a read or write if the DHT chooses to access that node. Therefore a slow network connection might disproportionately slow down access times. However, DHTs enable increased parallelism for reads because data can be read from multiple nodes at the same time, possibly making the network connection less of a bottleneck.

For the exploratory test on Planetlab, the results were that occasionally the file operations would freeze for a period of time, presumably waiting for a slow node of the network. Because Planetlab gives no guarantees of network latencies or load on the various computers, it is difficult to diagnose the cause of these sporadic delays. These delays might be due to slow nodes or fast nodes that suddenly become slow, as suggested in the DHT literature.[77]

The result of the redundancy test, which was performed three times on different occasions, was that the file was read successfully from the remaining nodes after one (with three virtual nodes) had been detached from the DHT network.

E. Evaluating DHTs

While the relatively slower performance of DHTs may prove impractical for some use cases, optimizations may make the performance problems less severe. DHT's are at a point in research progress where their advantages may outweigh their disadvantages for storing medical records, thereby justifying closer scrutiny by storage companies and open source developers.

Several design changes both within DHash and in the EMR application can be made that would make the performance problems of DHash less of an obstacle. For DHash, optimizations that consider bandwidth and computer usage could make it work on a system like Planetlab. However, a medical storage environment would likely involve dedicated machines with known bandwidth and therefore avoid the problems exhibited in the Planetlab experiment. Still, these kinds of optimizations would make DHash more robust in a dynamic computing environment.

EMR applications might use DHash as a permanent repository, but use a local file system as a temporary cache for more common or more recent data accesses. This would amount to a form of asynchronous communication which may leave a small window of potential data loss for write operations but would also greatly increase performance. Also, Indivo's current method of storing all files as individual XML files does not lend itself to storing large files. A better approach would be to buffer the data so that memory is used more efficiently and in a scalable fashion. This would involve breaking the XML file into smaller pieces to be stored incrementally.

The slower access times of DHTs may be irrelevant in most situations. Smaller files, which contain the relevant data during emergencies, can still be read in a matter of seconds. The larger files such as image files will likely only be looked at when there is sufficient time to fetch the file from the DHT in advance. Preparing a subset of emergency data could obviate time-consuming searches. Indivo's modular approach allows parts of a record to be retrieved individually so that smaller files containing text information could be still be retrieved without the entire patient record. Also, the write operations to DHTs, while significantly more time consuming than storing directly to disk, is a less urgent operation compared to reading the data and therefore can tolerate more delay.

With the addition of mitigating optimizations, DHTs' performance is within reasonable bounds for use in storing medical data.

The advantages of DHTs are, first, they are decentralized: no one physical location contains all the data. This directly addresses the problem with public perception of centralized storage of medical data. Second, there is a high level of redundancy per disk space used because of erase codes and the level of redundancy is easily configurable. Third, performance and cost will scale with increased size because DHTs use commodity hardware for nodes. Therefore large amounts of data can be stored among an increasingly large set of nodes with only moderate decreases in performance. Delay increases on the order of log N, where N is the number of nodes.[71] Fourth, expensive database software for managing and synchronizing the replicas is not needed. Backup is done as an intrinsic part of the file system and is easily manageable by simply inserting and removing nodes. Fifth, DHTs replace the expenses involved with leasing high speed lines for mirroring with many cheaper network connections that might achieve high performance through parallelism.

Disadvantages over traditional centralized backup systems are, first, DHTs require more physical locations to achieve their disaster resilience. Managing so many locations might be expensive but will be required for increased preservability. Second, performance of current DHT technology can be severely hurt by one slow node, as shown in the experiment with Planetlab. This problem is less severe if it is accessed asynchronously or with dedicated computers and bandwidth. Third, distributed transactions have not yet been implemented. While this constraint may be relaxed for medical data since medical data are rarely deleted or changed[5] (in computer parlance, the term is *immutable*), potential lack of consistency might still prevent the large scale deployment of DHTs or it may allow them to be used only as a backup rather than as a replacement for all medical storage systems. If the application server is centralized, consistency may be enforced within the server application; however, managing backup servers may then prove somewhat burdensome.

To be used for medical record storage, DHT technology must demonstrate several additional capabilities. First, performance for queries must be shown to be reasonable. Second, DHT technology must be demonstrated to maintain its functionality under heavy load. Third, because DHTs are not available as professional products, the cost of DHT software is unknown. Similarly, no estimates exist of the cost of supporting the required hardware and bandwidth infrastructure.

DHTs may be close to a stage in research where they are ready to cross the chasm from academic research into real world deployment. As medical records become increasingly digitized, the market for decentralized storage may similarly grow. Storage companies or open source communities should consider investigating DHT-based file systems as replacements to modern databases, which are complex, hard to administer and expensive.[78]

[5] In fact, in the Indivo model, although stored data may be made inaccessible by the patient, they are never deleted. Even when modified, the old data are retained as part of the auditing facility, and merely marked as no longer current.

Chapter 8: The Role of Government

The previous two chapters argue for potential ways to achieve increased resilience by focusing mostly on what RHIOs could do differently, strategically and technologically. Absent from the discussion so far is the role of government. Does the government have an interest in increasing medical record resilience? If so, what laws or regulations would best support the government's objectives? This chapter addresses both of these questions in turn.

The US federal government has already taken an interest in regulating medical record resilience with the HIPAA Security Rule's section on contingency plans as described in part 1, implying that resilience of medical records is an appropriate place for government involvement. Plans issued from the Department of Homeland Security further support this view.[79] However, it is worthwhile to consider why the government might have such an interest.

Foremost among the reasons is that the current poor state of medical record resilience is the result of a market failure, particularly for communication channels but also for the adoption of advanced backup storage architectures. The policy goals described in part 1 are clearly distant targets today. Some possible reasons for this state are described in chapter 5, such as the powerful externalities in US healthcare. A market failure is, arguably, an opportunity for government intervention to correct the problem. "The argument for speeding up adoption for any type of innovation must satisfy two conditions: (1) adoption is better than no adoption, and (2) adoption today is better than waiting and adopting tomorrow. In this framework, the basic reason for intervention is correcting a market failure."[12] Increased resilience would satisfy both of these reasons and would therefore warrant government involvement provided the government could ameliorate this condition without causing more harm.

It is possible that the industry will self-regulate to some degree. For example, medical associations may establish certification bodies to verify disaster resilience plans, EMR systems and even communication channels. So far, the industry has not initiated such a plan, probably primarily due to the low adoption rate of EMRs in the first place. Even with higher EMR adoption, all of the reasons for low resilience described in chapter 2 will likely prevent any kind of momentum from building organically within the healthcare sector.

If the government were to get involved, how should it proceed? The current federal effort led by ONC will, if successful, significantly increase resilience, especially though opening the communication channels via HIE and PHRs. Because EMRs are crucial to resilience, any acceleration of EMR adoption through legal mandates, subsidies or other incentives would, if implemented effectively, undoubtedly facilitate an increase in medical record resilience. Another avenue would be to directly regulate resilience by tightening HIPAA's contingency plan requirements such as requiring certain preservability or availability scores for every provider.

The decision of how to tighten resilience requirements above the minimal level set by HIPAA will involve a sophisticated cost-benefit calculation. Public policies create uniform rules and therefore will inevitably have some negative consequences. The goal should be, of course, for the positive to outweigh the negative.

The following is a preliminary list of some of the important factors that should be considered as part of this calculation. Because regional differences will affect resilience planning, State or regional governments would likely be the ones performing these calculations and enacting the regulations.

1. Risk of disaster in region. Certification requirements should be tighter for states that are more prone to high risks. This would be a form of legally mandated insurance mediated by regional risk, similar to the way car insurance rates are adjusted according – in part – to the risk of accidents in a particular region.

2. Adoption rate of EMR. When tightening contingency plan regulations, probably the most important thing is to make sure the regulations do not slow the adoption of EMRs. If the backup requirements are too high without commensurate subsidies, the cost to providers of adopting EMRs will add to the already high barriers for EMR adoption. Therefore, it may be preferable to wait until EMR adoption is already over the "tipping point" before instituting new resilience requirements.

3. Local RHIO storage architecture. RHIOs that offer centralized storage could do much of the resilience planning and would therefore perhaps greatly benefit from tighter resilience laws because providers would be motivated to join a RHIO to fulfill its legal resilience obligation. RHIOs that offered strong disaster management services would see their membership climb as a result of tighter contingency laws and HIE might then happen as a side-effect in some cases.

4. Stakeholder willingness to pay for resilience. Payers may in some cases be willing to support greater resilience because they have an interest in patients receiving better healthcare during disasters. (Healthier patients file fewer claims.) If this happens without government involvement, certification may be unnecessary. Time will tell if this will happen in significant magnitudes.

5. Portion of providers who have access to the Internet. The Internet is still not universal. For regions that do not have extensive Internet infrastructure, higher requirements for resilience might prove impossible for some providers.

States should consider incrementally tightening medical record resilience requirements and initiating certification programs for contingency plans informed by these factors.

Chapter 9: Summary and Conclusions

In a system as complex and decentralized as the US healthcare system, making large-scale improvements such as increasing the resilience of medical records will be a challenging undertaking. Yet, it is an undertaking that is worthwhile if healthcare is to be adequately prepared for large-scale disasters.

An essential first step toward improving resilience is specifically defining the socially desirable goals that could serve as the basis for public policy. Part 1 of this thesis attempts to do that by creating a preliminary policy model of resilience in terms of preservability and availability, emphasizing that one overlooked public policy goal of resilience is that medical data should be available to the point of care. Once the public policy goals and their priorities are established and agreed upon, the next step is to explore the mechanisms for achieving those goals.

Thus far, the US has relied on the free market as the mechanism to implement resilience in medical record systems as well as flexible regulations such as the HIPAA Security Rule. Despite advances in information technology that make high levels of resilience feasible, the current level of resilience is notably minimal, suggesting that the market is failing and an increased government role is justified.

Government efforts to increase resilience as well as technological advances that might help the government efforts are explored in part 2. Several recommendations that are interrelated and aim to nudge the healthcare system into increased resilience are presented. These recommendations apply only to the healthcare system in the United States as it exists today. More centralized healthcare systems in countries that have less public concern for privacy may be best served by taking a different approach to improving medical record resilience. The recommendations are summarized here, categorized by the three general stakeholder groups at which they are targeted: RHIO designers, storage companies, and policymakers.

RHIO designers

RHIOs should take on a role in disaster management through centralized storage, if possible, and through advising providers on their contingency plans. Centralized storage would be a sensible approach because the patient medical data stored by the RHIO is the same data that would be needed during and after an emergency. This way RHIOs would relieve providers of the burden of creating an expensive contingency plan, effectively supplying providers with a shared hot-site. Even for RHIOs with decentralized storage, RHIOs would be expected to play a significant role in resilience planning because of the importance of HIE during a disruptive event. It would therefore be sensible for RHIOs to also provide services such as spreading best practices, which would replace consulting services for providers and help forge standardized contingency plans.

In addition to providing disaster management services, RHIO business models that give patients access to their own data would further promote resilience by making for more direct accountability to the patient for medical record storage.

Storage Companies

Storage companies or open source communities should consider developing DHT-based file systems. This would help solve the privacy concerns that some RHIO, face, allowing for data to be centrally controlled as well as stored among many nodes in a distributed fashion. DHTs also offer the potential for lower cost compared to current database management software, high scalability, and high resilience because of their fault tolerance characteristics.

Policy makers

Policymakers at the federal and state levels should consider establishing a HIPAA compliance certification program for contingency plans that supports the policy goals described in part 1. The implementation of such a certification should be through a state government certification mechanism if the industry does not show efforts at self-regulation as EMRs are adopted. Increasing the legal requirements for resilience should be made carefully and should consider many factors such as the level of EMR adoption, risk of large-scale disaster and RHIO business models in the particular region.

In conclusion, medical record resilience can be defined as a goal of public policy and can be realized through a mix of strategic, technology, and policy mechanisms. The policy model expresses the social goals of medical record resilience; the recommendations are the means for implementing those goals. As the recommendations are further shown to be feasible, the policy goals are justified as worth pursuing.

Bibliography

1. Mann, D., *Katrina Shows Need for Electronic Health Records*. September, 23 2005: Fox News.com; http://www.foxnews.com/story/0,2933,170146,00.html, accessed April 10, 2007.
2. *Hurricanes Push Electronic Medical Records*: Yahoo Health; http://health.yahoo.com/topic/cancer/resources/article/mdanderson/797E5C5F-0B3F-4E70-9D0FDCB1F4B330BD, accessed April 9, 2007.
3. Weisfeld, V.D., *Lessons From KatrinaHealth*. June 13, 2006: KatrinaHealth.org; http://katrinahealth.org/katrinahealth.final.pdf accessed April 10, 2007.
4. Blake, E.S., *THE DEADLIEST, COSTLIEST, AND MOST INTENSE UNITED STATES TROPICAL CYCLONES FROM 1851 TO 2004*. 2005: National Hurricane Center; http://www.nhc.noaa.gov/pdf/NWS-TPC-4.pdf accessed April 12, 2007.
5. Popiolkowski, J., *A history of recent U.S. disasters*. January 13, 2006: Stateline.org; http://www.stateline.org/live/details/story?contentId=80383 accessed April 12, 2007.
6. Toigo, J.W., *Disaster Recovery*. 3rd ed. 2003, Upper Saddle River, New Jersey: Prentice Hall PTR.
7. Rosencrance, L., *Problems abound for Kaiser e-health records management system*. November 13, 2006: Computer World http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9005004&pageNumber=1 accessed April 26, 2007.
8. *Amazing Charts*, http://www.amazingcharts.com/OSBU/index.htm accessed May 18, 2007.
9. Howard, M., *How Two Rights Can Make a Wrong*, in *New York Times*. 2007.
10. Catharine W. Burt, E.H., David Woodwell, , *Electronic Medical Record Use by Office-Based Physicians: United States, 2005*. 2005: Center for Disease Control; http://www.cdc.gov/nchs/products/pubs/pubd/hestats/electronic/electronic.htm accessed April 15, 2007.
11. Association, A.H., *Forward Momentum: Hospital Use of IT*. 2005: AHA http://www.aha.org/aha/content/2005/pdf/FINALNonEmbITSurvey105.pdf accessed April 15, 2007.
12. Bower, A., *The Diffusion and Value of Healthcare Information Technology*. 2005: RAND.
13. *Office of the National Coordinator for Health Information Technology*: Department of Health and Human Services http://www.hhs.gov/healthit/ accessed April 25, 2007.

14. *State of the Union Addresses, 2005-2007*: White House http://www.whitehouse.gov/news/releases/2007/01/20070123-2.html; http://www.whitehouse.gov/news/releases/2006/01/20060131-10.html http://www.whitehouse.gov/news/releases/2005/02/20050202-11.html accessed April 25, 2007.
15. *The Computer-Based Patient Record: An Essential Technology for Health Care*. 1997: Institute of Medicine http://books.nap.edu/openbook.php?isbn=0309055326 accessed April 15, 2007.
16. Andrews, J., *Much healthcare data at risk*, in *Healthcare IT News*. January 1, 2006 http://www.healthcareitnews.com/story.cms?id=4266 accessed April 15, 2007.
17. Crounse, B., *The great hunt for an EMR*. March 21, 2006: Microsoft Healthcare Providers; http://www.microsoft.com/industry/healthcare/providers/businessvalue/housecalls/ImplementingEMRs.mspx accessed April 16, 2007.
18. *http://m-w.com/dictionary/resilience accessed April 16, 2007*.
19. *§ 164.524 Access of individuals to protected health information.*: Department of Health and Human Services; http://www.hhs.gov/ocr/regtext.html accessed April 16, 2007.
20. Szolovits, P., *Getting One's Own Medical Records*: Unpublished class study at Massachussetts Institute of Technology, class 6.872.
21. Fioriglio, G. and P. Szolovits, *Copy Fees and Patients' Rights to Obtain a Copy of Their Medical Records: From Law to Reality*. 2005: AMIA 2005 Symposium Proceedings.
22. *HIPAA Security Rule*. February, 20 2003: http://www.cms.hhs.gov/SecurityStandard/Downloads/securityfinalrule.pdf accessed April 16, 2007.
23. *HIPAA Security Rule, Notice of Proposed Rulemaking*. April 12, 1998: http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=1998_register&docid=fr12au98-28.pdf accessed April 16, 2007.
24. Hinegardner, S., *Data Storage for Managing the Health Enterprise And Achieving Business Continuity.* Journal of Health Information Management. 17(2).
25. *Business Continuity Glossary.* Disaster Recovery Journal http://www.drj.com/glossary/drjglossary.html accessed May 18, 2007.
26. *Emergency Responder Electronic Health Record Detailed Use Case*. December 20 2006: Office of the National Coordinator for Health Information Technology http://www.hhs.gov/healthit/documents/AHICEmergencyEHRDetailedUseCase.pdf accessed April 16, 2007.
27. Interview. 2007: Conversations with a knowledgeable source on Massachusetts health information exchange projects.

28. Shortliffe, E.H. and J.J. Cimino, *Biomedical Informatics: Computer Applications in Health Care and Biomedicine 3rd Ed*. 2006, Springer. pp 452.
29. Ghemawat, S., H. Gobioff, and S.-T. Leung, *The Google File System*. 2003: http://216.239.37.132/papers/gfs-sosp2003.pdf accessed April 19, 2007.
30. *University of Chicago Hospitals and Health Systems*. 2005: EMC, customer profile. http://www.emc.com/cp/pdf/H1748_UChicago_CP_ldv.pdf accessed April 19, 2007.
31. *Hospital Washes Hand of Backup*. June 2004: Storage Magazine http://storagemagazine.techtarget.com/magItem/0,291266,sid35_gci969978,00.html accessed April 19, 2007.
32. Biggar, H., *Recovery-focused Data Protection: Research Shows Your Future Depends On It*. January, 2007: EMC http://www.emc.com/analyst/pdf/ESG_RM_WP_final.pdf accessed April 19, 2007.
33. Francis, B., *IBM invigorates LTO tape storage*. November 29, 2004: Info World http://www.infoworld.com/article/04/11/29/48NNtape_1.html accessed April 19, 2007.
34. *PCMag.com Encyclopedia*: PC Magazine, http://www.pcmag.com/encyclopedia_term/0,2542,t=hot+backup&i=44380,00.asp, http://www.pcmag.com/encyclopedia_term/0,2542,t=cold+backup&i=39952,00.asp accessed April 19, 2007.
35. Greiner, C., *Continuous data protection: addressing timely data recovery and much more*. October 2005: Ovum http://www.emc.com/analyst/pdf/Ovum200510.pdf accessed April 19, 2007.
36. *Performing Backups*: Oracle http://www.oracle.com/technology/documentation/berkeley-db/je/GettingStartedGuide/backup.html acccessed April 20, 2007.
37. Staimer, M., *Data determines the right disaster recovery*. January 2005: Storage Magazine, http://storagemagazine.techtarget.com/magItem/0,291266,sid35_gci1042972,00.html accessed April 19, 2007.
38. Fallows, J., *File Not Found: Why a store tablet is still better than a hard drive*. September 2006: Atlantic.
39. *Scheme for getting patient medical records from recalcitrant HMOs*: Health Administration Responsibility Project; http://www.harp.org/recordmethod.htm accessed April 23, 2007.
40. *Today in E-Health Business*. February 6, 2007: AISHealth.com http://www.aishealth.com/EHealthBusiness/020607.html accessed April 23, 2007.
41. Krohn, R., *The Consumer Centric Personal Health Record — It's Time.* Journal of Health Information Management, Winter 2007. 21(1).

42. Tang, P.C., *et al., Personal Health Records: Definitions, Benefits, and Strategies for Overcoming Barriers to Adoption.* Journal of the Amer Med Informatics Assoc, Mar/April 2006. 13(2).
43. *Personal Health Records and Personal Health Systems: A Report Recommendation from the National Committee on Vital and Health Statistics.* February, 2006: U.S. Dept. of Health and Human Services http://www.ncvhs.hhs.gov/0602nhiirpt.pdf accessed April 23, 2007.
44. Yasnoff, W.: http://williamyasnoff.com/ accessed April 23, 2007.
45. *WellPoint Takes the Personal Health Records Leap*. September 1, 2006: Managed Care, http://healthdatamanagement.com/portals/article.cfm?type=managed_care&articleId=13926 accessed April 23, 2007.
46. Baker, M.L. and Z. Davis, *Large Employers to Provide Online Personal Health Records*. December 7, 2006: eWeek http://www.eweek.com/article2/0,1759,2069937,00.asp accessed April 23, 2007.
47. *Financial, Legal, and Organizational Approaches to Achieving Electronic Connectivity in Health-care*. October 2004: Connecting for Health http://healthcare.xml.org/resources/flo_sustain_healtcare_rpt.pdf accessed April 25, 2007.
48. Tucker, A.R.M.a.C.E., *Privacy, Networks Effects and Electronic Medical Record Technology Adoption*. January 8, 2007: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=960233 accessed April 27, 2007.
49. Sheffi, Y., *The Resilient Enterprise*. 2005: MIT press.
50. *US Department of Health & Human Services*: http://www.hhs.gov/transparency/ accessed April 24, 2007.
51. Leavitt, M., *Remarks as Prepared for Delivery at the Detroit Economic Club*. January 29, 2007: Department of Health & Human Services http://www.hhs.gov/news/speech/2007/012907.html accessed April 24, 207.
52. Coase, R., *The Problem of Social Cost.* Journal of Law and Economics, October 1960.
53. Kleinke, J.D., *Dot-Gov: Market Failure And The Creation Of A National Health Information Technology System.* Health Affairs, 2005. 24(5).
54. Olson, M., *The Rise and Decline of Nations*. 1984, New Haven and London: Yale University Press.
55. Halamka, J., *The Perfect Storm for Electronic Health Records.* Journal of Healthcare Information Management, 2006. 20(3).
56. *EMR Experts*: http://www.emrexperts.com/partners/steelgate.php accessed April 24, 2007.
57. *HIMSS Conference*: http://www.himss07.org/onlinedaily/Wednesday.aspx accessed April 30, 2007.

58. *Funding RHIO Startup and Financing for Life*. June 2006: Healthcare IT Transition Group.
59. Atoji, C., *Massachusetts RHIO Makes Progress*. February 22, 2007: Didgital Healthcare and Productivity http://www.digitalhcp.com/newsitems/2007/february/02-22-07-dhp-mass-rhio accessed April 30, 2007.
60. Glaser, J., *The Integration of EHR, PHR and eprescribing: A Collision?* February 2, 2007: Massachusetts Health Data Consortium http://www.mahealthdata.org/forums/events/2007/HIT_0202/slides/HIT07_Glaser.pdf accessed April 20, 2007.
61. Simons, W.W., K.D. Mandl, and I.S. Kohane, *The PING Personally Controlled Electronic Medical Record System: Technical Architecture. .* J Am Med Inform Assoc, 2004.
62. Yasnoff, W. 2006: eHealth Trust http://ehealthtrust.com/docs/YasnoffConsumerCentricHC.ppt accessed April 30, 2007.
63. *Washington State Builds Around Yasnoff Model*: Rural Telecommunications Congress http://www.ruraltelecon.org/index.php?q=node/41 accessed April 30, 2007.
64. deBrantes, F., et al., *The Potential of HIEs as Infomediaries.* Journal of Health Information Management, Winter 2007. 21(1).
65. *Infomediary*: Webopedia http://www.webopedia.com/TERM/I/infomediary.html.
66. *Massachusetts eHealth Collaborative*: http://www.maehc.org/communities.html accessed May 24, 2007.
67. Porter, M. and V. Millar, *How Information Gives You Competitive Advantage.* Harvard Business Review, July-Aug 1985.
68. *Personal communication*.
69. Overhage, J.M., *Indiana Health Information Exchange*: http://www.mc.vanderbilt.edu/vcbh/ds/st_hit_rndtbl/presentations/03/03-02_Indi.ppt accessed April 30, 2007.
70. *Common Framework*: http://www.connectingforhealth.org/ accessed April 30, 2007.
71. Stoica, I., et al., *Chord: A scalable peer-to-peer lookup protocol for Internet applications.* 2002: IEEE/ACM Transactions on Networking.
72. Balakrishnan, H., et al., *Looking up data in P2P systems*. 2003: Communications of the ACM.
73. Sit, E., J. Cates, and R. Cox, *A DHT-based Backup System*. 2003: Project Iris http://project-iris.net/isw-2003/papers/sit.pdf accessed May 13, 2007.
74. Lillibridge, M., et al., *A cooperative backup system scheme*. June 2003: In Proceedings of the 2003 USENIX Technical Conference.
75. Weatherspoon, H. and J. Kubiatowicz, *Erasure coding vs. replication: A quantitative comparison*. 2002: Proceedings of the 1st International Workshop on Peer-to-Peer Systems

76. *Planetlab*: https://www.planet-lab.org/ accessed May 28, 2007.
77. Sean Rhea, B.-G.C., John Kubiatowicz, and Scott Shenker, *Fixing the Embarrassing Slowness of OpenDHT on PlanetLab.* Proceedings of the Second Workshop on Real, Large Distributed Systems (WORLDS '05).
78. Bosworth, A., *Adam Bosworth's Weblog*. 2004: http://www.adambosworth.net/archives/000038.html accessed May 15, 2007.
79. *National Infrastructure Protection Plan*. 2006: Department of Homeland Security.

APPENDICES

Appendix A: Healthcare Contingency Planning Survey

Healthcare CIO Survey on EMR Contingency Planning

Directions: Please answer the following questions as best you can. Completion of this survey is completely voluntary. You are not required to answer any or all of the questions. **The results of this survey will be presented in a way that does not allow you to be identified with your answers.** Thank you very much for your participation.

Please email results and questions to Bob Rudin: bobrudin@mit.edu

Background: I am a graduate student at MIT studying Technology and Policy. This survey is part of my Masters research on the resilience of electronic medical records.

Q1. How big is your healthcare delivery organization?
- A) 100+ physicians
- B) 50-100 physicians
- C) Less than 50 physicians

Q2. Does your organization do formal contingency planning?
- A) Yes
- B) No; how would you describe your planning efforts? ______________________

Comments:

Q3. How has your organization developed and implemented a contingency plan for data backup and recovery of electronic medical records? (Choose all that apply.)
- A) Hired external consultants. Were you satisfied with their work? _________
- B) Developed and implemented internally.

Comments:

Q4. Which kinds of disruptions have you planned for in your contingency plan? (Choose all that apply)
- A) Application or database host failures
- B) Storage device failures
- C) Regional floods
- D) Earthquakes
- E) Tornadoes
- F) Other ______________________

Which have occurred?

Q5. Would you consider participating in a collaborative effort with other local healthcare delivery organizations to coordinate practices and resources for contingency planning and operations?
- A) Yes
- B) No

Comments: (Please explain your answer.)

Q6. In the event of a disruption, which processes are automated and which are manual? (Circle manual OR automated for each.)

1) Application host failover to backup: manual OR automated
2) Database failover to backup: manual OR automated
3) Storage device failover to backup: manual OR automated

Q7. What is your recovery time object (RTO) for data to be restored and accessible in the event of a disruption?

Q8. What kinds of patient data have backup copies made regularly? (Choose all that apply)

A) All
B) Doctor's notes
C) Lab results
D) Images
E) Other________

Comments:

Q9. How many backup copies are made? How far geographically are the backups located from the original source? (Choose all that apply.)

A) 1 copy; distance_______ (Kinds of data: _____________)
B) 2 copies; distances_____, ______ (Kinds of data: _______________)
C) More: ___________

Q10. How frequent are the backup copies made? (Choose all that apply)

A) Weekly (Kinds of data: ______________________________)
B) Daily (Kinds of data: ________________________________)
C) Continuously (Kinds of data: _______________________________)
D) Other______

Comments:

Q11. What technologies does your organization use for backup and recovery? (Choose all that apply.)

A) Database replication (Kinds of data: ______________________________)
B) File replication (Kinds of data: ______________________________)
C) Online disk backup (Kinds of data: ______________________________)
D) Tape backup (Kinds of data: ______________________________)
E) Other: ________________________

Q12. Were backup and recovery plans for patient data integrated into the EMR software when it was purchased?

A) Yes
B) No

Please describe any integration difficulties encountered when setting up the backup and recovery plans:

Q13. I am formulating a model for defining the policy goals of EMR resilience in terms of preservability and availability.

Preservability	**Availability**
Recoverability of data after disruption	Delivery time to point of care
Recovery point objective (RPO)	Delivery time of advanced features
Incremental data capture	

Definitions:

1) Recoverability - the ability to recover the data after a disruption.
2) Recovery point objective (RPO) – the time between periodic updates.
3) Incremental data capture - the ability to continue recording a patient's information during and after a disruption.
4) Delivery time to point of care - time it takes to get data to the point of care.
5) Delivery time of advanced features – time it takes to reestablish features such as roll-based data access and clinical decision support features.

Do you agree with this model as the foundation for a definition of EMR resilience as a policy goal?

A) Yes

B) No; why not? _____________

Do you have any feedback on this model?
Comments:

If possible, please send me a copy of any written documents containing your data backup and recovery plans.

Thank you for your participation!

Appendix B: Source code for Hearingaid

Hearingaid.c:

```
#include <chord.h>

//stl
#include <3.3/vector>

#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <stdio.h>
//#include <pthread.h>

//#include <dhash.h>
#include <dhash_common.h>
#include <dhashclient.h>
//#include <verify.h>
#include <dhblock.h>

#include <jni.h>
#include "CallBackData.h"

//#include "test_TestChord.h"
//#define CLASSPREFIX(a) Java_test_TestChord_##a
//#define VARIABLEPREFIX(a) test_TestChord_##a

//#include "org_chip_ping_phr_datastore_file_DistributedStore.h"
//#define CLASSPREFIX(a) Java_org_chip_ping_phr_datastore_file_DistributedStore_##a
//#define VARIABLEPREFIX(a) org_chip_ping_phr_datastore_file_DistributedStore_##a

#include "org_chip_ping_iostore_file_DistributedStore.h"
#define CLASSPREFIX(a) Java_org_chip_ping_iostore_file_DistributedStore_##a
#define VARIABLEPREFIX(a) org_chip_ping_iostore_file_DistributedStore_##a

//#define BLOCKSIZE 16384
// supports files up to (BLOCKSIZE-260)/20 * BLOCKSIZE/20 * BLOCKSIZE
// currently:  10815307776 bytes


dhashclient *dhash = NULL;
FILE *outfile;

void logme(char * buffer, bool detailed = false)
{
//    if (!detailed) //comment this out to print detailed
//    {
        fprintf(outfile, buffer);
        warn << buffer;
//    }
}
void logmeln(bool detailed = false)
{
    logme("\n", detailed);
}

char hexLUT(unsigned char i)
{
    if (i<10) return i + '0';
    if (i>15) return 0;
    return 'a' + (i - 10);

}

char * getChordIDstr(const char * chordIDbytes)
{
    unsigned char * usstr = (unsigned char *) chordIDbytes;
    char * str1 = new char[2*VARIABLEPREFIX(CHORDLENGTH)+1];
    for(int i=0; i < VARIABLEPREFIX(CHORDLENGTH); i++)
    {
        str1[i*2] = hexLUT(usstr[i] / 16);
        str1[i*2 + 1] = hexLUT(usstr[i] & 15);
//          warnx << "str1[" << i*2 << "] = "; printf("%c\n", str1[i*2]);
//          warnx << "str1[" << i*2 + 1 << "] = "; printf("%c\n", str1[i*2 + 1]);
    }
    str1[2*VARIABLEPREFIX(CHORDLENGTH)] = '\0'; //end the string
    return str1;
}

char * getChordIDstr(chordID chID)
{
    char * pchchordID = new char[VARIABLEPREFIX(CHORDLENGTH)];
    mpz_get_raw (pchchordID, sha1::hashsize, &chID);
    return getChordIDstr(pchchordID);
}

JNIEXPORT void JNICALL CLASSPREFIX(chordInit)
  (JNIEnv *env, jclass obj, jstring socketpath, jstring logfile)
{
    outfile = fopen(env->GetStringUTFChars(logfile, NULL), "w");
    logme ("Hello World\n");

    dhash = New dhashclient(env->GetStringUTFChars(socketpath, NULL));
    return;
}



void store_cb(CallBackData * cbData, dhash_stat st, ptr<insert_info> pii)
{
  char str1[255]; sprintf(str1, "store_cb: cbData->GetCount is this %d\n", cbData->GetCount());
  logme(str1);
  if(st != DHASH_OK)
  {
    logme("C: bad store\n");
    cbData->SetError();
  }
  if (cbData->GetCount() <= 0)
  {
    logme("C: Strange. Some weirdo is calling me back unexpectedly.\n");
    cbData->SetError();
  }
  else cbData->DecrementCount();
}

chordID write_block(char *buf, int len, CallBackData *cbData)
{
  chordID ID = compute_hash(buf, len);
  dhash->insert (ID, buf, len, wrap(&store_cb, cbData));
  return ID;
}

JNIEXPORT jbyteArray JNICALL CLASSPREFIX(store)
  (JNIEnv * env, jclass obj1, jbyteArray databyteAr)
  {
    char str1[250];
    logme("Inside hearingaid store\n", true);

      //extract char's from byte array of data
    jbyte *databytes = env->GetByteArrayElements(databyteAr, 0);
    jsize databyteArlength = env->GetArrayLength(databyteAr);
    char * pchdata = (char *) databytes;

    logme("C: Storing data.\n");
    sprintf(str1, "C: Datasize = %d\n", databyteArlength);
    logme(str1);

    //divide the data into 16K chunks and write each chunk into chord

    char * pchIndex = pchdata;
    int numberOfChunks = (databyteArlength / VARIABLEPREFIX(BLOCKSIZE)) + 1;

    //make array of chordIDs to store in afterward: as large as necessary for now
    char chordIDChunk[numberOfChunks * VARIABLEPREFIX(CHORDLENGTH)];
    chordID ID;
    int i;

    CallBackData *cbData = new CallBackData(numberOfChunks);

    sprintf(str1, "C: numberOfChunks = %d:\n", numberOfChunks);
    logme(str1);

    for (i = 0; i < numberOfChunks - 1 ; i++)
    {
            ID = write_block(pchIndex, VARIABLEPREFIX(BLOCKSIZE), cbData);
       mpz_get_raw (&(chordIDChunk[i*VARIABLEPREFIX(CHORDLENGTH)]), sha1::hashsize, &ID);
            pchIndex += VARIABLEPREFIX(BLOCKSIZE);
    }
    //finish up the last write
    int remainingBytes = databyteArlength % VARIABLEPREFIX(BLOCKSIZE);

    sprintf(str1, "C: remainingBytes = %d:\n", remainingBytes);
    logme(str1);

    ID = write_block(pchIndex, remainingBytes, cbData);
    mpz_get_raw (&(chordIDChunk[i*VARIABLEPREFIX(CHORDLENGTH)]), sha1::hashsize, &ID);
    //poll until all data arrives
    while (0 < cbData->GetCount() && !cbData->GetError()){
            acheck ();
    }

    char str2[255]; sprintf(str2, "store: cbData->GetCount() is this %d\n", cbData->GetCount());
    logme(str2);

    //write chordIDChunks
    cbData->SetCount(1);
    ID = write_block(chordIDChunk, numberOfChunks*VARIABLEPREFIX(CHORDLENGTH), cbData);
    while (0 < cbData->GetCount()  && !cbData->GetError()){
        acheck ();
    }
    if (cbData->GetError()) logme("C: There was an error involving chord");
```

```
    //make byte array from chordID
    char * pchchordID = new char[VARIABLEPREFIX(CHORDLENGTH)];
    mpz_get_raw (pchchordID, sha1::hashsize, &ID);

    sprintf(str1, "C: Returning this chordID: %s\n", getChordIDstr(pchchordID));
    logme(str1);

    jbyte * ResultChordArbytes = (jbyte *) pchchordID;
    jbyteArray resultChordAr = env->NewByteArray(VARIABLEPREFIX(CHORDLENGTH));
    jsize resultArlength = env->GetArrayLength(resultChordAr);
    env->SetByteArrayRegion(resultChordAr, 0, resultArlength, ResultChordArbytes);

    return resultChordAr;
  }



void retrieve_cb(CallBackData * cbData, int offset, dhash_stat st, ptr<dhash_block> bl, vec<chordID> vc)
{
    if (cbData->GetCount() <= 0)
    {
        logme("C: Strange. Some weirdo is calling me back unexpectedly.\n");
        cbData->SetError(); return;
    }
    if(st != DHASH_OK)
    {
        logme("C: bad retrieve");
        cbData->SetError(); return;
    }
    if (NULL == bl->data.cstr())
    {
        logme("C: Received data was NULL.\n");
        cbData->SetError(); return;
    }

    if (NULL == cbData->GetData())
    {// must now make the memory since it wasn't supplied
     //that means it is either the chordID block, or the last chunk; we don't know the sizes before hand
        cbData->NewData(bl->data.len());
        cbData->SetData(bl->data.cstr(), bl->data.len());
    }
    else
    {// use the data and offset that was supplied
     // that means the data is just an ordinary chunk of size BLOCKSIZE
        if (VARIABLEPREFIX(BLOCKSIZE) != bl->data.len() )
        {
            logme("C: Unexpected Block size.");
            cbData->SetError(); return;
        }

              cbData->SetData((char*) bl->data.cstr(), bl->data.len(), offset);

    }

    char str1 [255];
    sprintf(str1, "C: retrieve_cb: bl->data.len() = %d;\n", bl->data.len());
    logme(str1, true);
    cbData->DecrementCount();

}
```

```
JNIEXPORT jbyteArray JNICALL CLASSPREFIX(retrieve)
  (JNIEnv *env, jclass obj1, jbyteArray chordByteAr)
  {
     logme("C: Inside retrieve.\n", true);
     char str2[250];

       //make chordID from byte array
     jbyte *chordBytes = env->GetByteArrayElements(chordByteAr, 0);
     char * chordChars = (char *) chordBytes;
     chordID crdID;
     mpz_set_rawmag_be(&crdID, chordChars, sha1::hashsize);

     //get the chordID block
     CallBackData *cbData= new CallBackData(1);

     dhash->retrieve(crdID, wrap(&retrieve_cb, cbData, 0));
     while (0 < cbData->GetCount() && !cbData->GetError()){
         acheck ();
     }

     if (0 != cbData->GetLength() % VARIABLEPREFIX(CHORDLENGTH))
     {
             char str1 [255];
        sprintf(str1, "C: cbData->GetLength() is %d which is not divisible by VARIABLEPREFIX(CHORDLENGTH)\n",
(int) cbData->GetLength());
        logme(str1);
             return NULL;
     }

     sprintf(str2, "C: retrieve: chordID block: cbData->GetLength() = %d\n", cbData->GetLength()); //for apach
     logme(str2, true);

     //take the chordIDs from cbData, put them in readChordIDs,
     //and read the results of those IDs into cbData
     int numberofChunks = cbData->GetLength() / VARIABLEPREFIX(CHORDLENGTH);
     char * data = cbData->GetData();

     CallBackData * cbChunkData = new CallBackData(numberofChunks-1);
     cbChunkData->NewData(VARIABLEPREFIX(BLOCKSIZE)*numberofChunks);

     //make vector of chordIDs to read
     std::vector<chordID> readChordIDs;
     readChordIDs.clear();
     for (int i = 0; i < numberofChunks; i++)
     {
         char chordIDBytes[VARIABLEPREFIX(CHORDLENGTH)];
                memcpy(chordIDBytes, &(data[i*VARIABLEPREFIX(CHORDLENGTH)]),
VARIABLEPREFIX(CHORDLENGTH));
         mpz_set_rawmag_be(&crdID, chordIDBytes, sha1::hashsize);
                readChordIDs.push_back(crdID);
     }

     for (int i = 0; i < numberofChunks - 1; i++)
     {
                dhash->retrieve(readChordIDs[i], wrap(&retrieve_cb, cbChunkData,
i*VARIABLEPREFIX(BLOCKSIZE)));

     }
     //get the last one
     CallBackData * cbLastChunkData = new CallBackData(1);
     dhash->retrieve(readChordIDs[numberofChunks-1], wrap(&retrieve_cb, cbLastChunkData, 0));
```

```
    while ( ((0 < cbChunkData->GetCount()) || (0 < cbLastChunkData->GetCount() ) ) &&
                        !cbChunkData->GetError() && !cbLastChunkData->GetError())
    {
       acheck ();
    }


    if (cbChunkData->GetError() || cbLastChunkData->GetError()) logme("C: Error in getting chunk", true);

    //copy cbLastChunkData's bytes to cbChunkData
    size_t retrievedDataLength = VARIABLEPREFIX(BLOCKSIZE)*(numberofChunks - 1) + cbLastChunkData-
>GetLength();
    sprintf(str2, "retrievedDataLength = %d\n", retrievedDataLength); //for apach
    logme(str2, true);
    sprintf(str2, "numberofChunks = %d\n", numberofChunks); //for apach
    logme(str2, true);
    sprintf(str2, "cbLastChunkData->GetLength() = %d\n", cbLastChunkData->GetLength()); //for apach
    logme(str2, true);
    //recycling the variable from above
    data = cbChunkData->GetData();
    memcpy(&(data[VARIABLEPREFIX(BLOCKSIZE)*(numberofChunks - 1)]), cbLastChunkData->GetData(),
cbLastChunkData->GetLength());

    jbyte *resultbytes = (jbyte*) data;
    jbyteArray resultbyteAr =  env->NewByteArray(retrievedDataLength);
    jsize resultbyteArlength = env->GetArrayLength(resultbyteAr);
    env->SetByteArrayRegion(resultbyteAr, 0, resultbyteArlength, resultbytes);

   // delete [] G_RESULT_DATA; //should get rid of memory leak
    return resultbyteAr;
  }


JNIEXPORT jbyteArray JNICALL CLASSPREFIX(gethash)
  (JNIEnv * env, jclass obj1, jbyteArray databyteAr)
  {
    char str1[250];
    logme("Inside hearingaid gethash\n", true);

      //extract char's from byte array of data
    jbyte *databytes = env->GetByteArrayElements(databyteAr, 0);
    jsize databyteArlength = env->GetArrayLength(databyteAr);
    char * pchdata = (char *) databytes;
    int keylength = databyteArlength;

    chordID ID = compute_hash(pchdata, keylength);

    logme("C: Getting Hash.\n");
    sprintf(str1, "C: Getting Hash: Datasize = %d\n", keylength);
    logme(str1);

    char * pchchordID = new char[VARIABLEPREFIX(CHORDLENGTH)];
    mpz_get_raw (pchchordID, sha1::hashsize, &ID);

    sprintf(str1, "C: Returning this chordID: %s\n", getChordIDstr(pchchordID));
    logme(str1);

    jbyte * ResultChordArbytes = (jbyte *) pchchordID;
    jbyteArray resultChordAr = env->NewByteArray(VARIABLEPREFIX(CHORDLENGTH));
    jsize resultArlength = env->GetArrayLength(resultChordAr);
    env->SetByteArrayRegion(resultChordAr, 0, resultArlength, ResultChordArbytes);
    return resultChordAr;
  }



/*
 * Class:     org_chip_ping_datastore_file_DistributedStore
 * Method:    storeKey
 * Signature: ([B[B)V
 */
JNIEXPORT void JNICALL CLASSPREFIX(storeKey)
  (JNIEnv *env, jclass obj1, jbyteArray chordByteAr, jbyteArray dataByteAr)
{
    char str1[250];
    logme("C: inside storeKey.\n", true);       //make chordID from byte array

    jbyte *chordBytes = env->GetByteArrayElements(chordByteAr, 0);
    char * chordChars = (char *) chordBytes;
    char * strChordID = getChordIDstr(chordChars);

    //extract char's from byte array of data
    jbyte *databytes = env->GetByteArrayElements(dataByteAr, 0);
    jsize databyteArlength = env->GetArrayLength(dataByteAr);
    if (databyteArlength != VARIABLEPREFIX(CHORDLENGTH))
    {
        sprintf(str1, "C: Trying to store to an ID that is not %d in length.\n", (int)
VARIABLEPREFIX(CHORDLENGTH));
        logme(str1);
        return; //FIXME: throw an exception... return false or something.
    }

    char * pchdata = (char *) databytes;
    char * strData = getChordIDstr(pchdata);
    logme("C: Storing. Key:"); logme(strChordID);
    logme(";\nC: Data: "); logme(strData);
    sprintf(str1, "; Datasize = %d\n\n", databyteArlength); //for apache
    logme(str1);       //make chordID from byte array

    CallBackData *cbData = new CallBackData(1);
    //insert into chord
    dhash->insert (compute_hash(chordChars, VARIABLEPREFIX(CHORDLENGTH)), pchdata,
VARIABLEPREFIX(CHORDLENGTH) ,
          wrap (store_cb, cbData), NULL, DHASH_NOAUTH); // doesn't compile without the getusec() thing

     while (0 < cbData->GetCount() && !cbData->GetError()){
        acheck ();
    }
  }

 void retrievelatest_cb(CallBackData * cbData, dhash_stat st, ptr<dhash_block> bl, vec<chordID> vc)
{
   logme("C: Inside retrievelatest_cb.\n", true);
   if (cbData->GetCount() <= 0) logme("C: Strange. Some weirdo is calling me back unexpectedly.\n");
   else cbData->DecrementCount();

   if (st != DHASH_OK)
   {
       logme("C: st != DHASH_OK\n");
       cbData->SetError();
             return;
   }
   char str1[255];
   sprintf(str1, "C: bl->vData.size() = %d\n", bl->vData.size()); logme(str1, true);

   if (bl->vData.size() > 0)
   {
       for (unsigned int j=0; j < bl->vData.size(); j++)
       {
           sprintf(str1, "C: Stored version %d: ", j); logme(str1, true);
           logme(getChordIDstr(bl->vData[j]), true); logmeln(true);
       }
             if (VARIABLEPREFIX(CHORDLENGTH) != (int) bl->vData[bl->vData.size() - 1].len())
             {
           sprintf(str1, "C: Latest value is not size of chordID. Rather it is: %d\n", bl->vData[bl->vData.size() - 1].len());
           logme(str1);
           cbData->SetError();
           return;
       }
             else
             { //we have found the latest written chordID
           cbData->NewData(VARIABLEPREFIX(CHORDLENGTH));
           bool berror = cbData->SetData((char*) bl->vData[bl->vData.size() - 1].cstr(),
                                           VARIABLEPREFIX(CHORDLENGTH));
           if (berror) logme("Size of callback data doesn't match size of data you are trying to set");
             }
   }
   else
   {//take the one and only chordID in there
       long lversionResultSize = (long) bl->data.len();
       if (lversionResultSize != VARIABLEPREFIX(CHORDLENGTH))
       {
         sprintf(str1, "C: Latest value is not size of chordID. Rather it is: %d\n", (int)lversionResultSize);
         cbData->SetError();
         return;
       }
       cbData->NewData(VARIABLEPREFIX(CHORDLENGTH));
       bool berror = cbData->SetData((char*) bl->data.cstr(),  VARIABLEPREFIX(CHORDLENGTH));
       if (berror) logme("Size of callback data doesn't match size of data you are trying to set");
   }
}


/*
 * Class:     org_chip_ping_datastore_file_DistributedStore
 * Method:    retrieveLatest
 * Signature: ([B)[B
 */
JNIEXPORT jbyteArray JNICALL CLASSPREFIX(retrieveLatest)
  (JNIEnv *env, jclass obj1, jbyteArray chordBytesAr)
  {
      //make chordID from byte array
    jbyte *chordBytes = env->GetByteArrayElements(chordBytesAr, 0);
    char * chordChars = (char *) chordBytes;
    char * strChordID = getChordIDstr(chordChars);
    logme("C: Getting from this key: "); logme(strChordID); logmeln();

    //prepare callback object
    CallBackData *cbData = new CallBackData(1);

    //get version block
    dhash->retrieve(compute_hash(chordChars, VARIABLEPREFIX(CHORDLENGTH)),
                      DHASH_NOAUTH, wrap (retrievelatest_cb, cbData));
    while (0 < cbData->GetCount() && !cbData->GetError()){
       acheck ();
    }

    if (0 == cbData->GetLength())
    {
        logme("C: No data found.\n");
        return NULL;
    }

    logme("C: Got latest chordID: ");logme(getChordIDstr(cbData->GetData())); logme("\n");

    //now that you have the latest chordID, don't for get to go get the data stored under it, dude!

    //copy last chordID into an array and return it
    jbyte *resultbytes = (jbyte*) cbData->GetData();
    jbyteArray resultbyteAr =  env->NewByteArray(VARIABLEPREFIX(CHORDLENGTH));

    jsize resultbyteArlength = env->GetArrayLength(resultbyteAr);
    env->SetByteArrayRegion(resultbyteAr, 0, resultbyteArlength, resultbytes);
    return resultbyteAr;
  }
```

CallBackData.c:

```
class CallBackData
{

        public:
                CallBackData()
                {
                        m_count = 0;
                        m_data = NULL;
                        m_size = 0;
                        m_error = false;
                }
                CallBackData(int count)
                {
                        m_count = count;
                        m_data = NULL;
                        m_size = 0;
                        m_error = false;
                }
                void SetCount(int i)
                {
                        m_count = i;
                }
                void DecrementCount()//make sychronous
                {
                        m_count--;
                }
                int GetCount()//make sychronous
                {
                        return m_count;
                }
                void SetError()//make sychronous
                {
                        m_error = true;
                }
                bool GetError()//make sychronous
                {
                        return m_error;
                }
                bool SetData(const char * data, size_t size, size_t offset = 0)//make sychronous
                {
                        if (size > (m_size - offset))
                        {
                                //logme
                                return true; // avoid buffer overrun
                        }
                        memcpy(&(m_data[offset]), data, size);
                        return false; //no error
                }
                void NewData(size_t size)
                {
                        if (NULL != m_data) delete []m_data;
                        m_data = new char[size];
                        m_size = size;
                }
                int GetLength()
                {
                        return m_size;
                }
```

```
				char * GetData()
				{
					return m_data;
				}

		private:
				int m_count;
				char * m_data;
				size_t m_size;
				bool m_error;

};
```