Arogyashree: A Distributed File System for Large Scale Internet-based Telemedicine

Kovendhan Ponnavaikko and D. Janakiram

Distributed and Object Systems Lab, Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai, India

Abstract. Today, a typical telemedicine system involves a small set of hospitals providing remote healthcare services to a small section of the society using nodal centers, mobile health units, etc. However, the benefits of a telemedicine system increase with scale. One of the key requirements of such a large scale system is to allow large numbers of patient medical records, in the form of electronic files, to be efficiently stored and accessed from widely distributed locations. In this paper, we address the need for a distributed file system to manage patient data in large scale telemedicine systems. We use the resources of unreliable Internet edge nodes distributed among hospitals, labs, etc., to provide reliable file system services to patients and doctors. Besides building an Internet-based system that scales with the number of nodes and files, we also attempt to optimize record access times for doctors to provide timely responses.

1 Introduction

Telemedicine is a fast evolving application that uses modern tele-communication networks to allow patients to be served by remotely located medical practitioners. Huge disparities exist in the distribution of quality health care among urban and rural populations in developing countries [1]. Telemedicine has the potential to lessen this disparity.

The last decade has seen the emergence of countless telemedicine setups all around the world [2], [3]. A typical system involves a small set of hospitals providing remote healthcare services to a small section of the population using satellite technology, nodal centers, mobile health units, etc. However, we envisage the need for much larger Internet-based telemedicine systems at national or provincial levels.

Large scale systems enable a large pool of doctors and hospitals to collectively provide healthcare services to entire populations. Patient records can be made accessible to doctors from any location seamlessly. This significantly increases a patient's chances of receiving high quality care, since specialist doctors may sometimes not be available nearby. Increasing mobility of doctors and patients is another reason that warrants the need for efficient access to patient records from any location. Also, in a large setup, patients have wider options to choose from, in terms of doctors and hospitals.

A large scale telemedicine system is capable of supporting increasing numbers of patients and is robust against failures and attacks. Moreover, such a system can also be

Ponnavaikko K. and Janakiram D. (2009).
Arogyashree: A Distributed File System for Large Scale Internet-based Telemedicine.
In *Proceedings of the 1st International Workshop on Mobilizing Health Information to Support Healthcare-related Knowledge Work*, pages 105-114
DOI: 10.5220/0001819101050114

productively integrated with third party health related systems such as national health insurance schemes.

Management of patients' Electronic Medical Records, or EMRs, is a key issue associated with telemedicine [4]. A patient's EMR may include a large variety of information such as demographics, immunization records, history of illnesses and treatments, allergies, pathological test results, ECGs, X-rays, etc. Though several different formats and semantics are used by various organizations, a patient's medical record is almost always an ever increasing set of electronic files of various sizes.

In a large scale telemedicine system, the problem of efficiently storing and retrieving EMR files of millions of patients assumes huge proportions. Storage and retrieval of EMR files must also be done as efficiently as possible so that patients can receive accurate and timely treatment. *Arogyashree* is an Internet-based Distributed File System (DFS) specialized to efficiently manage patient data in large scale telemedicine projects. Figure 1 highlights some of our system features.

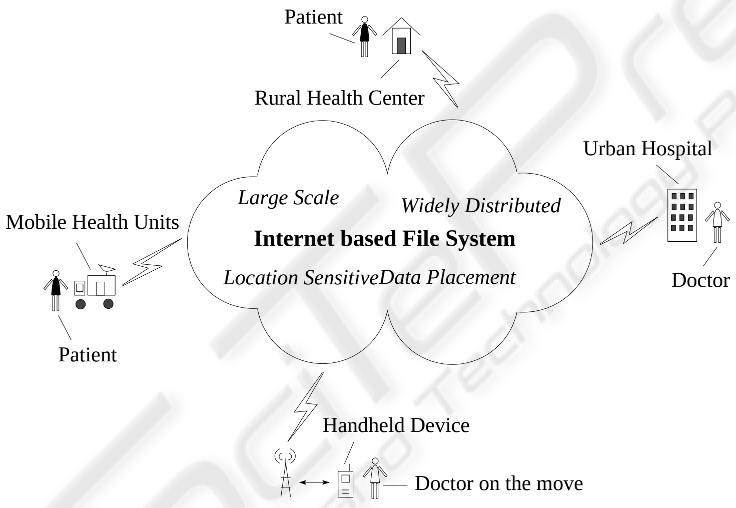

Fig. 1. System Overview.

The requirements of our system and some related work are discussed in section 2. In section 3, we present a few scenarios that are indicative of the purpose of *Arogyashree*. Detailed description of the system follows in section 4.

2 Background and Related Work

2.1 Heterogeneity of Patient Data

These days, patient records are usually stored in multiple healthcare organizations in different formats using different semantics. Several efforts, such as the Clinical Document Architecture (CDA) [5], are in progress to integrate the distributed patient information by making all the data conform to a reference architecture, such as HL7 [6]. In

Arogyashree, we assume that some such architecture is conformed to by all the participants.

Irrespective of the format used, patient data is almost always stored as electronic files in the storage sites of the various healthcare organizations. For example, in the HL7 architecture [6], CDA documents are encoded into XML [7] files with references and links to the other medical images and multimedia files. MedGrid [8] is framework for integrating patient information from various healthcare organizations. The framework, however, uses a central registry to maintain CDA metadata, which restricts it from scaling to billions of files.

2.2 Scalability: Usage of Internet Edge Nodes

One of the main requirements of our system is that it must scale to support a large number of patients (order of 10^6), and hence a large number of files (order of 10^9). Most of the existing solutions rely on manual data transfers. This requires users to learn how to access each file, which is not practical.

Some distributed file systems such as NFS [9] and AFS [10] use a limited number of servers to serve both data and metadata, severely constraining the system's scalability. Others such as GFS [11] and Lustre FS [12] widely distribute data serving responsibilities, but use a limited number of metadata servers. Peer-to-Peer file sharing systems, such as $Kazaa$ [13], support dynamic environments and also scale well. However, most such systems do not support mutable data. Doctors must not only be able to access patient files from any location, but must also be able to update the files with new results, treatment prescriptions, etc.

We handle the issue of scale by exploiting the resources (storage, CPU, network, etc.) of the virtually unlimited number of Internet edge nodes distributed among hospitals, medical colleges, labs, healthcare organizations and even other volunteering institutions and individuals [14] to provide reliable file system services to patients and doctors. The ubiquitous and inexpensive nature of the Internet enables such a system to have a global reach and not be limited by even national boundaries.

Internet edge nodes are made responsible for serving both data and metadata. When a large number of devices are used, it is critical to have efficient load balancing mechanisms in place to ensure that there are no potential bottlenecks in the system. Since edge nodes can arbitrarily fail or get disconnected, the system must also incorporate adequate fault tolerance mechanisms to render reliable file system services.

2.3 Heterogeneity and Mobility of Devices

Edge nodes can range from mobile handsets and PDAs to laptops, desktops and high-end servers. Node availability and capabilities such as storage capacity, processing speed, network connectivity, user interface, mobility, etc. can vary widely. The system must take into consideration these capabilities as well as node locations while scheduling data placements and accesses. Allowing mobile nodes to participate in the system enables doctors to remain connected for longer periods of time.

2.4 Control over Data Placement

Data management systems such as Ceph [15] and OceanStore [16] also scale well with large amounts of data and metadata. However, they provide little control over the kind and location of nodes on which metadata [16] and data [15] are stored. In *Arogyashree*, one of our primary concerns is to dynamically place data and metadata in appropriate locations so that doctors can access their patients' records without much delay. Access time to medical records is a critical parameter that affects the performance of a telemedicine system.

3 Indicative Scenarios

In this section, we present a few scenarios that demonstrate the usefulness of our file system for large scale telemedicine.

Scenario One: A rural patient, P_i, while registering with the system, will normally prefer doctors and hospitals from nearby towns or cities to provide him/her remote medical assistance. All of P_i's records, as and when they are generated, are then replicated in the storage nodes at the preferred hospitals. When medical attention is required, P_i visits a Rural Health Center (RHC) in his/her village. The practitioner at the RHC calls the city hospital (H_c) and makes a request for a doctor. The operator in the hospital connects up P_i at the RHC with the appropriate doctor at H_c. Since all of P_i's EMR is readily available at H_c, the doctor can instantly look them up, interact with P_i, and prescribe the necessary treatment and medication, without encountering data transfer delays.

Scenario Two: Suppose a patient, P_i, is regularly treated for a cardiovascular disease by a cardiologist, D_h. Specialist doctors usually work at different places at different times of the day/week. Our file system ensures that replicas of P_i's EMR are maintained on (or close to) storage nodes used by D_h. Thus, when P_i comes to a RHC with a critical problem, such as a cardiac arrest, D_h, wherever he/she is, will have immediate access to P_i's records with a high probability. D_h will thus be able to quickly recommend appropriate first aid measures and investigations.

Scenario Three: Mobile Health Units (MHU) play an important role in providing telemedicine services to remote populations. Using satellites or other wireless communication protocols, MHUs are generally capable of connecting to the Internet. Since mobile devices are also integrated into *Arogyashree*, MHUs can upload patient information into the file system immediately after recording them. In several existing telemedicine setups, medical data is sent by a MHU to a nodal hospital, where doctors have to be present to analyze the data and recommend appropriate treatment. In *Arogyashree*, patient data is automatically transferred to storage nodes close to relevant doctors, relevance being dictated by history, patient preference, proximity, specialization, etc. Doctors can then access the records and suggest a proper course of action. Doctors on the move can use their mobile devices to retrieve patient information and upload prescriptions. Thus, rural patients can be provided specialist care instantaneously.

4 Arogyashree: A DFS for Large Scale Telemedicine

4.1 Clustering

The primary task of a file system for large scale telemedicine is to manage billions of EMR files in such a way that a patient's data is stored close to the doctors who are most likely to treat him/her. This necessitates the usage of storage nodes close to doctors and hospitals. Therefore, we exploit the resources of the large number of nodes, connected to the Internet, which are distributed among hospitals, medical colleges, labs, healthcare organizations and even other volunteering institutions and individuals to store and serve patient files.

In order to manage the large number of nodes efficiently, system administrators group nodes into large non-overlapping clusters (order of 10^3 nodes per cluster), based on proximity. The cluster structure is known to the entire system (by means of well known sites).

Every location in the geographical area (province, country, etc.) covered by the telemedicine system is associated with one of the clusters. Hospitals, doctors and patients in a particular location, therefore, belong to the corresponding cluster, which will be their *home* cluster. All of a patient's EMR files are maintained by his/her home cluster.

Clustering is done in such a way that each cluster has a sufficient number of hospitals, doctors and nodes in its jurisdiction. This may require nodes in rural areas to be clustered together with nodes in the closest urban areas (Fig. 2), since urban areas, generally, have a much larger number of hospitals and doctors than rural areas.

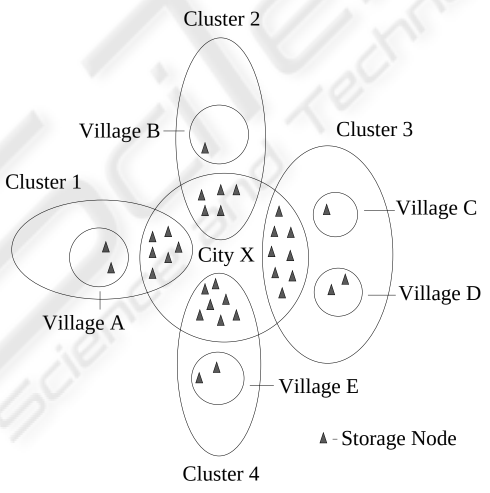

Fig. 2. Clustering Example.

4.2 Two-Layered System Model

A recent approach to large scale file system organization has been to decouple file data and metadata [11][15]. By distributing data and metadata management loads among a large number of nodes, bottlenecks are prevented and system performance is enhanced. While data refers to the actual EMR files, metadata contains useful information about the files such as the addresses of nodes on which they are stored.

For systems handling billions of EMR files, a large number of Metadata Servers (MDS) are required to handle the load without affecting performance. A few reliable and capable edge nodes from each cluster are chosen to be the metadata servers for that cluster. We call these servers *Supernodes*. Capabilities such as network bandwidth, processor speed, storage space, memory capacity, etc. are used to choose supernodes. The reliability of a storage node is measured in terms of the proportion of time the node is reachable and available for use. As a result, static nodes are made supernodes more often than mobile nodes, which tend to have higher disconnection rates.

Apart from being metadata servers, supernodes also monitor and manage the storage nodes in their cluster. Information about patients, doctors and hospitals belonging to their cluster are also managed by supernodes. For instance, doctors usually are associated with a set of hospitals, use/own a set of devices, etc. Similarly, patients usually visit a regular set of hospitals and doctors. Such information is useful for making EMR placement decisions.

Supernodes from all the clusters form a single system-wide structured P2P overlay network or Distributed Hash Table (DHT) [17], [18] (Fig. 3). The overlay is required to connect up all the clusters in the system. It helps users discover the location of records managed by other clusters, enables the efficient discovery of appropriate storage nodes in other clusters, etc.

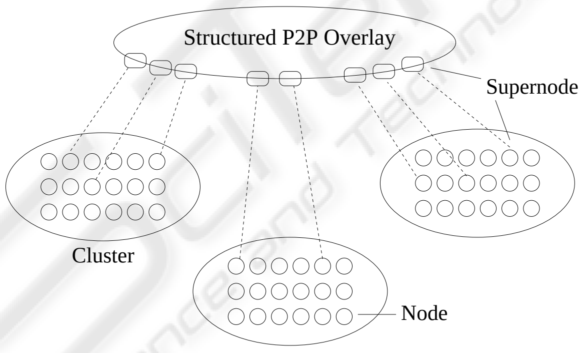

Fig. 3. Two Layered Platform.

4.3 Load Balancing

Metadata management and resource monitoring loads within a cluster are distributed among the cluster supernodes in accordance to their capabilities. A technique similar to the usage of virtual servers in Chord [19] is adopted. A large virtual identifier space is distributed among the supernodes of a cluster in accordance to their relative capabilities. When the load on a supernode increases, the responsibility for a portion of the virtual identifier space can be transferred from the heavily loaded supernode to a lightly loaded one. We use the Paxos algorithm [20] to achieve consensus among a set of supernodes, on the adjustment of the virtual identifier space.

The mapping between the virtual identifier space and the physical address of supernodes in a cluster is made known to the entire system using the structured layer. A file containing the association is stored in the structured layer with the cluster identifier as the key. We will refer to this file as the *supernodes_map*.

In order to balance load, files, nodes, patients, doctors, hospitals, etc. determine the virtual supernode identifier with which they must associate themselves by applying standard hash operations on their unique identifiers (name, address, location coordinates, MAC address, IMEI number, file path, etc.). Entities can then use the *supernodes_map* file to determine the physical address of the supernode responsible for that virtual identifier.

Supernodes in a cluster periodically share their load information with each other. When a supernode failure is detected or when all supernodes in a cluster are heavily loaded, a new supernode can be added to the existing set. On the other hand, when all the supernodes in a cluster are lightly loaded, one of the supernodes can be removed.

4.4 Fault Tolerance

In order to handle supernode failures, the metadata stored in a supernode is replicated in a constant number (k_r) of other supernodes. The number k_r is selected in such a way that the simultaneous failure of $k_r/2$ supernodes within a cluster is highly improbable. A supernode, S^n, that updates some metadata has to multicast the changes to all its replicas, so that the replicas are always in a state similar to that of S^n.

Whenever the responsibility for a virtual identifier is transferred from S^i to S^j, appropriate metadata transfer must happen. The metadata of all the resources that map to that virtual identifier must be transferred from S^i to S^j and to S^j's replica set.

Storage nodes periodically send information about their vital characteristics, such as available storage space, network connectivity, location, etc., to their corresponding supernodes. These status messages help supernodes detect storage node failures. The frequency with which status messages are received also helps in gauging the reliability of a node. Cluster supernodes periodically exchange node status information among themselves.

4.5 Security and Privacy

Patient data privacy and security are important issues in a telemedicine system. In *Arogyashree*, different kinds of cryptographic techniques can be used to encrypt patient data and preserve its privacy. Standard practices such as the usage of Kerberos authentication protocols for the servers and clients to authenticate each other and establish secure sessions, using Access Control Lists (ACLs) to restrict the usage of files, etc. can also be adopted. The Artemis [21] project presents a mechanism for the secure transfer of patient information between health care organizations in a Peer-to-Peer (P2P) setting.

4.6 File Placement

The objective of our file placement strategy is to place files at locations where they are most often accessed. A patient's medical records are most often accessed at the hospitals that he/she frequents. Many a times, a patient is attended to by a regular set of doctors from nearby locations. Thus, when a new file is added to a patient's records, *Arogyashree* attempts to replicate the file on storage nodes belonging to the hospitals and doctors associated with the patient.

The set of hospitals and doctors a patient is associated with may change with time. *Arogyashree* autonomously migrates patient records to the new locations. In order to avoid excessive replication, replicas are maintained in the proximity of a limited set of recently and frequently visited hospitals and doctors only.

When the ideal storage nodes do not have enough storage space, nodes nearby are used to replicate the file. Network coordinates are used to discover proximal nodes. We employ Vivaldi [22], a decentralized network coordinate system, to assign network coordinates to storage nodes. We require only an approximate estimate of network distances to discover nearby storage nodes. Therefore, the accuracy provided by Vivaldi [23] is sufficient for our purpose. Vivaldi vertices consist of Euclidean coordinates augmented with a height and can be represented as (x, y, h). Network distance (or round trip time) between nodes is calculated in a way similar to Euclidean distance measurement. Mobile devices may need to maintain more than one set of coordinates, since their latencies from other nodes may vary based on their location and the connection used (WiFi, cellular network, WiMAX, etc.).

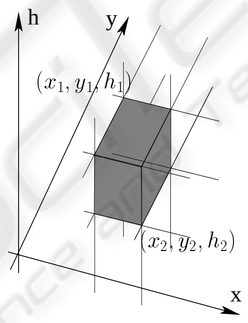

Fig. 4. Location Space Partitioning.

In each cluster, supernodes take responsibility for different blocks of the coordinate space. The distribution is done using hash functions, as discussed in section 4.3. Each block is uniquely identified by the endpoints of its diagonal (Fig. 4). The list of cluster nodes in a block is maintained by the corresponding supernode. Nodes proximal to a particular node can then be found using simple geometric algorithms.

Supernodes ensure that files are replicated in a minimum number of storage nodes with reasonable reliability measures. When failed nodes remain inaccessible for abnormally long periods of time, their contents are replicated in other nodes. An outline of the replica placement mechanism is shown in figure 5.

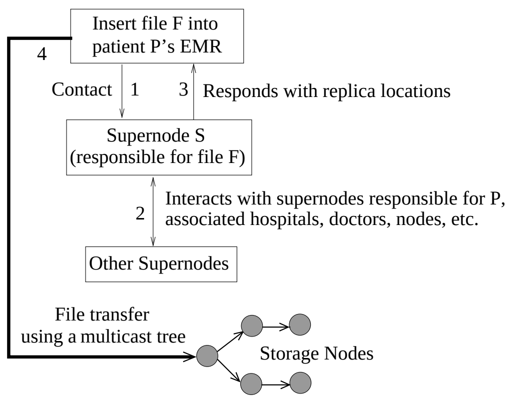
Insert file F into patient P's EMR
4
Contact
1
3
Responds with replica locations
Supernode S (responsible for file F)
2
Interacts with supernodes responsible for P, associated hospitals, doctors, nodes, etc.
Other Supernodes
File transfer using a multicast tree
Storage Nodes

Fig. 5. Replica Placement.

5 Conclusions

In this paper, we discuss the design of a distributed file system suitable for large scale Internet-based telemedicine projects. *Arogyashree* uses the resources of large numbers of unreliable Internet edge nodes to provide reliable file services to doctors and patients. A two layered platform, comprising of clusters of proximal nodes and a system-wide DHT, enables the distribution of metadata and data management loads among the edge nodes. The working of our system assists in optimizing EMR access times, which is a critical parameter that affects the performance of a telemedicine system. At our lab, we are currently implementing a prototype model of *Arogyashree* as part of a sponsored project. The prototype will be used for the evaluation of the system's performance and scalability.

Acknowledgements

We thank Intel Research and Indian Council of Medical Research for supporting this work.

References

1. Bagchi, S.: Telemedicine in rural India. PLoS Medicine 3 (2006)
2. Website: Belgium-hf. (http://www.belgium-hf.be/)
3. Website: Center for connected health. (http://www.connected-health.org/)
4. Fraser, H. S., Biondich, P., Moodley, D., Choi, S., Mamlin, B. W., Szolovits, P.: Implementing electronic medical record systems in developing countries. Informatics in Primary Care 13 (2005)
5. Kim, H.S., Tran, T., Cho, H.: A Clinical Document Architecture (CDA) to generate clinical documents within a hospital information system for e-healthcare services. In: CIT '06: Proceedings of the Sixth IEEE International Conference on Computer and Information Technology, Washington, DC, USA, IEEE Computer Society (2006) 254

6. Dolin, R.H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F.M., Biron, P.V., Shabo Shvo, A.: HL7 clinical document architecture, release 2. J Am Med Inform Assoc 13 (2006) 30–39
7. Bray, T., Paoli, J., Sperberg-Mcqueen, C.M., Maler, E.: Extensible Markup Language (XML) 1.0 (Fourth Edition). Technical report, W3C (2006)
8. Papadakis, I., Poulymenopoulou, M.: Medgrid: A semantic-capable grid for medical data. The Journal on Information Technology in Healthcare 4 (2006)
9. Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., Lyon, B.: Design and implementation of the Sun Network Filesystem. In: Proc. Summer 1985 USENIX Conf. (1985) 119–130
10. Howard, J.H.: On overview of the Andrew File System. In: USENIX Winter. (1988) 23–26
11. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google File System. SIGOPS OSR 37 (2003) 29–43
12. Schwan, P.: Lustre: Building a file system for 1000-node clusters. In: Proc. of Linux Symposium. (2003) 380–386
13. Website: Kazaa. (http://www.kazaa.org/)
14. Sarmenta, L.F.G.: Bayanihan: Web-based volunteer computing using java. In: In Second International Conference on World-Wide Computing and its Applications. (1998) 444–461
15. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C.: Ceph: A scalable, high-performance distributed file system. In: OSDI. (2006) 307–320
16. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. SIGARCH Comput. Archit. News 28 (2000) 190–201
17. Stoica, I., Morris, R., Karger, D., Kaashoek, F.M., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for Internet applications. In: Proc. of ACM SIGCOMM '01. Volume 31. (2001) 149–160
18. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: Middleware '01. (2001) 329–350
19. Rao, A., Lakshminarayanan, K., Surana, S., Karp, R.M., Stoica, I.: Load balancing in structured P2P systems. In: Proc. of IPTPS 2003. (2003) 68–79
20. Lamport, L.: The part-time parliament. Trans. Comput. Syst 16 (1998) 133–169
21. Boniface, M.J., Wilken, P.: Artemis: Towards a secure interoperability infrastructure for healthcare information systems. In: HealthGrid 2005. (2005)
22. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: Proc. of the ACM SIGCOMM'04 Conference. (2004) 15–26
23. Lua, E.K., Griffin, T., Pias, M., Zheng, H., Crowcroft, J.: On the accuracy of embeddings for internet coordinate systems. In: Proc. of the 5th ACM SIGCOMM conference on Internet Measurement, Berkeley, CA, USA, USENIX Association (2005) 11–11