HL7 based Real-Time Clinical Data Integration System Using Advanced Database Technology

Sooyoung Yoo[a], Boyoung Kim[a], Heekyong Park[a], Jinwook Choi[a], Jonghoon Chun[b]

[a]Dept. of Biomedical Engineering, College of Medicine Seoul National University, 28 Yongon-Dong Chongro-Gu Seoul, Korea
[b]Division of Computer, Myongji University, Young-In Kyonggido, Korea

Abstract

As information & communication technologies have advanced, interest in mobile health care systems has grown. In order to obtain information seamlessly from distributed and fragmented clinical data from heterogeneous institutions, we need solutions that integrate data. In this article, we introduce a method for information integration based on real-time message communication using trigger and advanced database technologies. Messages were devised to conform to HL7, a standard for electronic data exchange in healthcare environments. The HL7 based system provides us with an integrated environment in which we are able to manage the complexities of medical data. We developed this message communication interface to generate and parse HL7 messages automatically from the database point of view. We discuss how easily real time data exchange is performed in the clinical information system, given the requirement for minimum loading of the database system.

Keywords:

HL7, Data Integration, Real-time System

Introduction

With the recent movement toward shared clinical data in health care, a number of models, methods, and evaluative strategies have been developed. Data integration, especially in the medical environment, is the most important issue that must be considered. In 2000, the Institute of Medicine (IOM) in the United States reported that medical errors result in as many as 98,000 deaths each year. This implies that medical errors are the eighth leading cause of death in the U.S., and that they nearly as high as those caused by motor vehicle accidents (43,000), breast cancer (42,000) and AIDS (17,000) combined [1]. The IOM reported that the decentralized and fragmented status of the health delivery system is the main cause of such medical errors.
The mission of biomedical informatics is to enable people to use information to improve health care. The integration of data from a variety of sources will also improve the clinical decision making process [2].

This paper describes the design and implementation of real-time clinical data integration from a viewpoint of database. Also, an HL7 interface that is developed to solve the problem of plug and play interoperability is discussed. HL7 (Health Level 7) has been proposed as a standard for electronic data exchange in medical environment [3], to understand data communication and system interoperability among the various systems.

Background

Considerable research effort has been directed towards acquiring integrated data, reducing redundancy, and merging results in multiple devices including portable devices. In our first stage multiple devices computing project (the LEX project), we developed a clinical information system composed of three parts: an interface for the multiple device, a central data repository, and an HL7 message server. According to this system, the HL7 message server retrieves data from the hospital's information system, such as basic patient information and the clinical test results, and directs the data to the central data repository in HL7 version 2.3.1 format [4]. We assume multiple device environment in which clinicians will check the clinical data via multiple device supporting interface. The implementation of HL7 is essential in such a heterogeneous IT environment.
To fetch and store the patient data in the central data repository, the data should be retrieved from the legacy hospital information system regularly. Because a hospital information system should be fault tolerant, the newly induced system/application should not interfere with the performance of the system. On finding new data in the system, it generates a new HL7 message and attaches the data in a second. In the LEX project, we adopted a polling method to monitor the new data in the legacy hospital information system. Because of the possible overhead to the system, we set the monitoring interval to be no shorter than one second.

In the second mobile clinical computing project (MobileMed^{TM}), we constructed a triggering function as a stored procedure in the database, which ensures that whenever changes to the database such as data insert or delete or update occur, it transfers data to an external application, the HL7 message server, in real-time. The reason is that the polling overloads to the legacy system and creates the possibility of a time delay. This paper details the method for the communication between the laboratory database and an external application.

Methods

Architecture for Real-Time Clinical Data Integration

We developed this prototype mobile clinical information system called MobileMed™, as an integrated health care delivery system which gathers distributed and fragmented patient clinical data into a central clinical repository that allows healthcare providers to access a patient's clinical data using multiple devices (Figure 1). If test results from the various kinds of LIS equipment including, blood, chemistry and urinalysis test equipment stored in the LIS database at a medical institution, this data is collected (or pushed) into the central clinical database in real-time, and may be easily accessed anywhere via available internet devices.

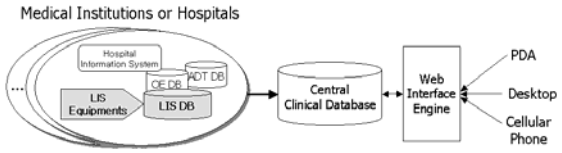

Figure 1: Overall Architecture of MobileMed™ System

The architecture of real-time data integration using the HL7 standard is shown in the Figure 2. We designed the HL7 Message Server (HMS) as an interface at each medical institution and the HL7 Message Archiver (HMA) as an interface for the central clinical database. These two interfaces communicate with each other by HL7 messages, currently version 2.3.1 messages, on the TCP/IP network. Also, to minimize the database communication overheads and the amount of data to be transmitted to the HMS, the LIS database usually sends only small-sized data sets including several key identifiers, to the HMS whenever interesting transactions occur. The HMS queries again to obtain complete data needed for fulfilling the HL7 messages if necessary.

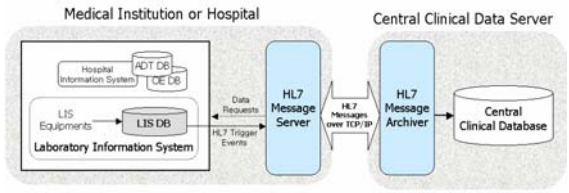

Figure 2: HL7 Message-based Standard Interface Architecture

Sending Real-Time Data from LIS DBMS

Java's safety and automatic memory management allows for tight integration with the RDBMS. Thus, Java and RDBMS support the rapid assembly of component-based, network-centric applications that can evolve gracefully as business needs change [5]. For this reason, many of the major database vendors - Oracle, Informix, Sybase and IBM, are supporting Java by embedding a Java Virtual Machine (JVM) in their servers [6]. In this paper, we implemented our system using Oracle 9i database technology. Given the advantages of Java and database synergy, we leveraged Java network programming into the database system to send data from the DBMS to an external application.

Having used this technology to design the architecture of the database components, we examined how to identify a specific HL7 triggering event and how the identified event is notified to the external application without creating database overhead.

In our system, we needed to integrate real-time patient recent lab test results data and service provider information into a central clinical database. From the viewpoint of the database, we were able to identify the triggering event (event R01) served by the ORU (Observational report - Unsolicited) message in combination with the ACK message using SQL insert statement into a specific table (a recent lab results table in the LIS database). Because the triggering event occurs whenever an LIS equipment interface inserts newly results into a recent lab results table (see Figure 3). Likewise, the SQL insert, delete and update statements performed on a staff/practitioner master table is represented as an M02 trigger event for which MFN (Master File Notification) message exchanged in combination with an MFK (Master File Application Acknowledgement) message.

Figure 3 shows the relationships between key tables of LIS database.

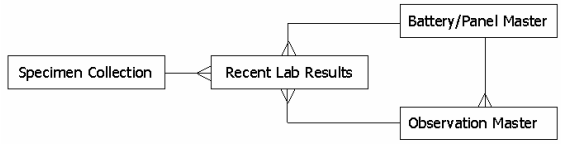

Figure 3: Partial LIS Database Schema

And the embedded database modules are composed of a set of Java stored procedures and database trigger, as can be seen in Figure 4.

Java Stored procedures are Java methods published to SQL and stored in databases for general use [5]. The reason we imported a Java stored procedure is that Java is becoming the stored procedure language of choice, promising portability and safety. In addition, by using a portable stored procedure language, code can be transferred between servers from different vendors, vendor-specific training is reduced and database-independent applications can be distributed with application-specific stored procedure codes [6].

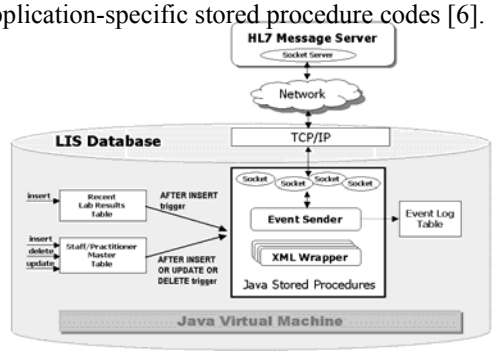

Figure 4: Architecture for transferring real-time data

The database triggers and stored procedures shown in Figure 4 functions as follows;

XML Wrapper: This is a Java stored procedure that returns XML formatted ASCII text on the basis of the value of its parameters. The reason we choose XML encoded messaging for communications between DBMS and the external applications is that XML provides a consistent, language-independent interface for programmers while providing full flexibility in determining the granularity and consistency of the information as accessed.

Event Sender: To send real-time data from DBMS to an external program, we implemented network programming using a TCP socket in Java. This is a Java stored procedure that sends an XML string received by its parameter to an external program using an instance of socket upon the TCP protocol. In addition, to assuring communication reliability without any data loss or failure, we verified the successful receipt or failure of the transmission by checking the received acknowledgement. That is, if the socket connection fails or doesn't receive any acknowledgement, it inserts a data failed to send into an *event_log* table, which is designed to keep data temporarily until the connection is re-establish successfully. To resend the data in the *event_log* table without database load, we defined a new HL7 trigger event "Z01" which is triggered only once to indicate that the *event_log* table has more than one record. Using this trigger event, we transferred only small-sized XML data like "<?xml version='1.0'?><doc event='Z01'></doc>" when the network is available. We then expected the external application to select and delete all records from the *event_log* table after receiving Z01 events.

After insert trigger on recent lab results table: This trigger is fired after a row is inserted into the recent lab results table. Because a database trigger can invoke the Java stored procedure, a triggered observation reporting event can be notified to an external application using the previously discussed *XML wrapper* and *Event Sender* procedures. That is, just after encoding data elements in XML format using its appropriate *XML wrapper*, we can easily send the encoded XML data to external application by calling the *Event Send* procedure.

After insert or update or delete trigger on staff/practitioner master table: This trigger is fired after insert or update or delete on our staff/practitioner master file. This trigger notifies the occurrence of an event to an external application using above *XML wrapper* and *Event Sender* procedure, as explained before.

To minimize database communication overhead, we used the following two methods to notify specific events in real-time. First, If the trigger events cause frequent data communications or the data elements for its HL7 message are dispersed on several tables, with a complex relationship, we transfer only key elements to identify each trigger event from the DBMS. For example, Figure 5 shows the key data tagged with XML attributes and elements including the trigger event type, observation code, patient number, service of battery code and specimen number. This XML document is sent from the database, and then the receiving application retrieves all the additional data it needs from the database. In addition, using the database view approach, we could design a receiving application logically independent of the database. Second, If the trigger events occurs infrequently and the triggered table has enough data for its HL7 message, with a size of less than 1000 characters, all the data elements required for the HL7 message can be transmitted without database load. For example, when the master file notification event are triggered, we sent the XML documents contain all the information needed for the staff/practitioner master file notification message.

```
<!ELEMENT doc (patno, service_code, obs_code, specimen_no)>
<!ATTLIST doc
        event CDATA #REQUIRED
        date CDATA #REQUIRED
>
<!ELEMENT obs_code (#PCDATA)>
<!ELEMENT patno (#PCDATA)>
<!ELEMENT service_code (#PCDATA)>
<!ELEMENT specimen_no (#PCDATA)>
```

Figure 5: DTD of XML document generated after insert trigger on a recent lab results table

HL7 Message Server (HMS)

The HMS is the interface that automatically generates HL7 V2.X messages from the LIS database and exchanges messages with the central clinical database. Figure 6 shows the overall architecture of the HMS.

We designed two approaches for generating messages. One is based on database query and the other is based on XML. First, the database query approach is appropriate when the HMS receives an event that contains only key identifying information such as Figure 5, from database. In this case, a database query is necessary to obtain the actual data needed to fulfill a complete message like unsolicited observation reporting message. At this time, the database query is issued to views for HL7 messages as shown in Figure 5. We defined views to make HMS independent of a specific database schema. The second approach based on XML is suited for master file notification messages, because the event coming from the database contains all information needed for this message.

For either of each message generation mechanism, additional message segments can be added easily to address institution specific requirements. Because message contents can be different according to the view data or XML formatted data.

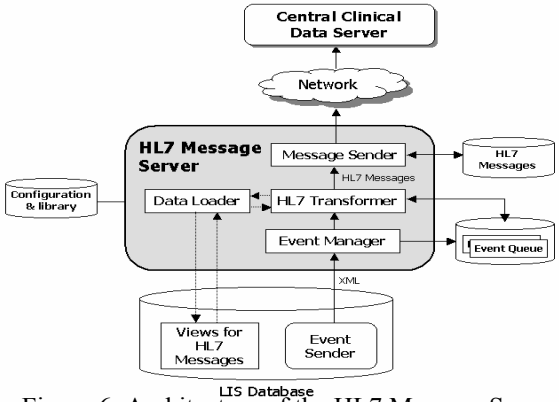

Figure 6: Architecture of the HL7 Message Server

HL7 Message Archiver (HMA)

The HMA is an interface on the central clinical data server side that processes a large number of HL7 messages from the HMS, and interacts with the central clinical database. It is composed of two components, a message *receiver* and a *parser&mapper*. To process large messages from heterogeneous medical institutions, the *receiver* notifies the events about incoming messages to the *parser&mapper* using UDT protocol.

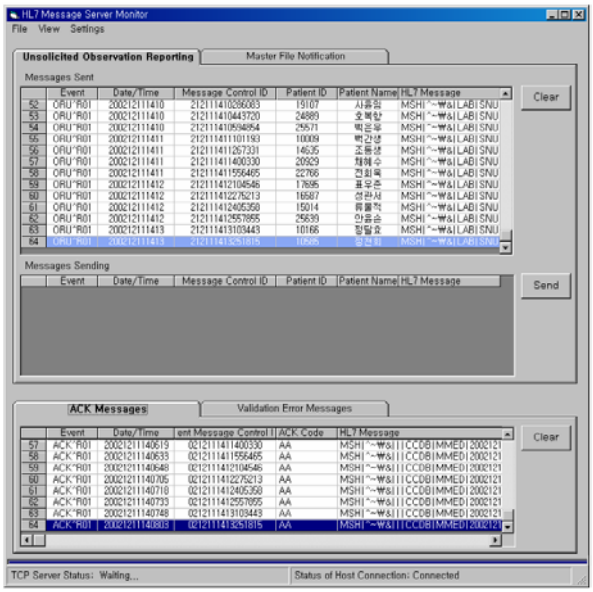

Figure 7: UI for monitoring the HMS

We implemented the HMS and the HMA using MS Visual Basic 6.0 on a Windows 2000 Server. Figure 7 shows the user interface used for monitoring HMS's operation. Each of the three sections of Figure 7 displays sent messages, sending messages, received ack messages. Figure 8 shows the two components of the HMA, receiver and parser&mapper.

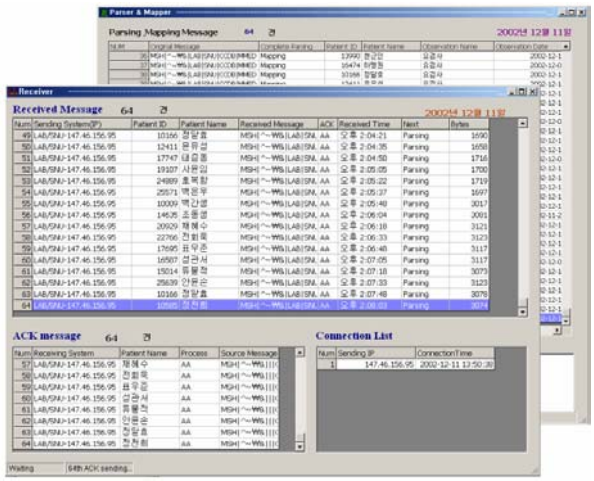

Figure 8: Receiver, Parser & Mapper of the HMA

Result

The described system is a prototype that has been developed over 2 years from 2001. During this project, each of the six kinds of the clinical laboratory test results - routine CBC, blood coagulation test, chemistry test (I), chemistry test (II), serology, routine urinalysis – were sent to the LIS database from LIS equipment when it was generated. To share clinical results with other institutions, the data in the LIS database is gathered into CCDB using HL7 messages. Currently we exchange with HL7 V2.3.1 message, however, the HL7 message version can be extended to any 2.X message.

Conclusions

In this paper, we present a mobile clinical information system MobileMed^{TM}, which integrates distributed patient clinical data using a standard interface and advanced database and information technologies. To achieve seamless integration in real-time, we implemented a socket network between DBMS and external applications using a Java stored procedure.
This system represents the future of the health care system, where patient clinical data can be easily shared among authorized practitioners/institutions and easily accessed anywhere via the internet.

In future work, we plan to solve the lack of interoperability for information representation with unified ontology in regard to having the same meaning in codes, vocabulary, terminology, context, and other means of information representation.

Acknowledgements

This study was supported by a grant of the Korea Health 21 R&D Project, Ministry of Health & Welfare, Republic of Korea. (01-PJ1-PG4-01PT06-0002)

References

[1] RE Rouse, JS Charlson. The evolution of eHealth. Lehman Brothers report, 2000
[2] W. Stead, R. Miller, M. Musen et. al. Integration and beyond. JAMIA 2002. 135-145
[3] http://www.hl7.org
[4] Sooyoung Yoo, Boyoung Kim, Seungbin Han, Youngchul Lee, Jinwook Choi, Jaeheon Cheong, Minkyung Lee, Jonghoon Chun. Automatic RIM (Reference Information Model) Wrapper for LEX : Lifelong Electronic Health Record Based on XML. AMIA 2001.
[5] http://otn.orcle.com/
[6] http://www.firstsql.com/

Address for correspondence

Sooyoung Yoo, MS Candidate
Department of Biomedical Engineering, College of Medicine, Seoul National University, 28 Yongon-Dong Chongro-Gu Seoul 110-799, Korea, E-mail: yoosoo0@snu.ac.kr