Challenges on real-time monitoring of patients through the Internet

Duarte Pereira and Adriano Moreira
Department of Information Systems
University of Minho
Campus de Azurém
4800-058 Guimarães, Portugal
adriano@dsi.uminho.pt

Ricardo Simões
Institute for Polymers and Composites
University of Minho, 4800-058 Guimaraes, Portugal
and
Polytechnic Institute of Cávado and Ave
Campus do IPCA, 4750-810 Barcelos, Portugal
rsimoes@ipca.pt; rsimoes@dep.uminho.pt

Abstract— **Real-time remote monitoring of patients aims at improving and facilitating the accessibility to data about the vital signals of hospitalized patients, or patients that need to be closely monitored while at their own homes, by medical care professionals. By resorting to remote monitoring systems, the need for clinical staff to physically move next to each patient to supervise their clinical condition is reduced.**

This paper focuses on the key issues and challenges faced in the design and implementation of a distributed remote patient monitoring system using the Internet as communication platform, while describing some of the technologies that can be employed for this purpose. To optimize the systems' usability by medical care professionals, intuitive interfaces were developed using Web technologies, with an Internet browser as the only tool required to access the system. The study demonstrated that data updating based on database pooling system (periodic queries) is not the optimal solution, as the resulting system is not scalable. Also, monitoring several patients simultaneously (on the same browser) requires considerable processing at the client machine, particularly in the case of ECG signals.

Keywords: Real-time health monitoring; System architecture; ECG signal.

I. INTRODUCTION

Recent advances in wireless networking technologies have opened up new opportunities in a variety of applications including healthcare systems [1]. Although continuous monitoring systems are used in hospitals, these systems require the sensors to be hardwired to nearby bedside monitors, essentially confining the patient to his hospital bed. The advent of wireless technologies like Wi-Fi and Bluetooth enabled higher mobility of the patients by replacing the cable connection between patient sensor and bedside equipment by a wireless connection[1]. However, these systems are often designed only for breaking the cord between the patient and the monitor without supporting the transmission of data through a network for many receivers or monitors [2].

Since its beginning, the World Wide Web has played an important role in information sharing through the Internet [3]. In this context, our team initiated the development of a system aiming to provide health professionals with enhanced information on their patients' vital signs, and increase the mobility of the patients while being continuously monitored. To this endeavor, the objective was to create a patient monitoring system that would allow health professionals to access patients information without the need to move next to the patients, and without requiring the use of specific applications or proprietary terminals, but instead allowing the use of a simple Web browser on a desktop computer or mobile device.

However, monitoring biomedical signals through the Internet, using wireless networks, sets new requirements and raises a number of challenges still difficult to overcome. This paper describes some of these challenges that were identified through the implementation of a noncritical patient monitoring system using Web technologies.

II. PATIENT MONITORING SYSTEM

In the past, other researchers developed similar systems to that being developed by our team, where the visualization of vital signs is done using a Web browser [3-8], and where the main advantage of these systems is the remote access to data.

The web based Intelligent Online Monitoring System for Intensive Care Units (IMI) [3] is a system that collects ECG data. This is based on a client/server architecture and uses Java Applets to represent the ECG data in a Web browser.

Another system [4] that only analyzes data from ECG was developed with the goal of collecting data remotely from patients at home using the Internet. Through a Web browser, the ECG values of each patient can be monitored in real time.

Lee [5] also describes a system that consists on remote monitoring of patients, in real time, using Web browsers for monitoring the collected data.

The system we developed includes the acquisition of patients' vital signs through wireless sensors, including heart rate (HR), blood pressure (BP), pulse, temperature and electrocardiogram (ECG), all of which were completely developed by our team [9-11]. The system also includes data storage, remote data access, and data visualization. The developed sensors are equipped with a ZigBee network interface that sends the collected data to a central server through a wireless network created for that specific purpose.

The central server contains applications that allow data to be received, stored (in a database) and made available for visualization The stored data is made available to health professionals by web applications that can be accessed using a web browser running on a PC or PDA, both through the hospital network (WiFi or Ethernet) or from the Internet. The server can also receive data from the Internet when data acquisition takes place outside the hospital environment, as in the case of patients monitored at home.

The monitoring system includes the generation of alarms, both audible and on-screen, and also the display of historical data for the indicators of each patient.

III. DEVELOPED SYSTEM

The system being developed by our team includes: A) the design and implementation of a set of wireless sensors that measure the vital signs and send the corresponding data over a wireless (ZigBee) network to a gateway; B) the integration of data originated from multiple sensor networks into a central repository; C) the remote visualization of the vital signs of one or more patients; D) the management of the complete system, including users' management, remote configuration of the sensors, and management of the patients information. The overall architecture of the system is illustrated in Figure 1.

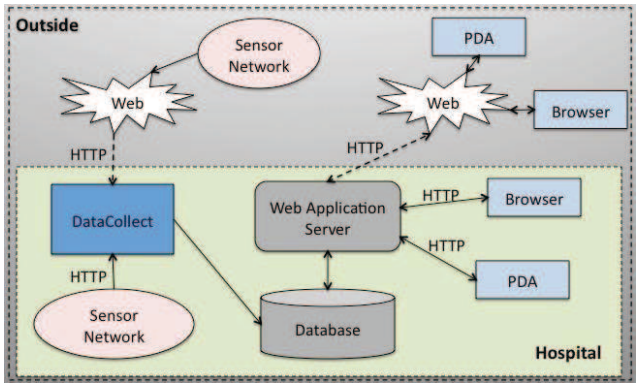

Figure 1: System Architecture

This paper only addresses parts B and C of the global system: data integration and data visualization (several aspects of part A are described in [9-11]). Parts B and C consist mainly of three services: 1) a Data Collection Application (DCA) responsible for continuously receiving data from the network of sensors, either from the local network or from the Internet, and store it in a database, 2) a Web Application Server responsible for providing an interface that allows users to interact with the system, including visualizing the monitored vital signs, and 3) a database (DB) where all the data is stored (such as information on patients, sensors, users, alerts and sensor data).

The DCA is a server application and was developed to meet the demands of the sensor networks. A gateway, in each sensor network, collects data frames from the sensors and transmits them to the DCA, over the local network or Internet, using the HTTP protocol. During the design phase, several options have been considered for the communications protocol between the several components of the system. Among them, UDP and HTTP have been considered since some parts of the system were to be connected through the public Internet. The final choice was the HTTP protocol since it is the one that presents the lowest probability of being blocked by proxies, or firewalls. .The DCA has been implemented using Java and uses Java Servlet technology to respond to requests from the gateways. To support its implementation, an Apache Tomcat Web server was used mostly because it is known for being fast, stable and designed for multi-tasking, being able to serve multiple simultaneous requests [6], being free and supporting Java Servlets and Java Server Pages [12].

The Web Application Server supports three functionalities: the User Interface, Monitoring and remote Data Access. The User Interface is a Web application that allows users to manage the system. This provides features such as authentication, management of patients, sensors and users, setting up alerts, association of sensors to patients, and monitoring of indicators for each patient, all through a Web interface. The Monitoring functionality allows the user to graphically visualize each of the patient signals. This functionality is embedded into the above mentioned Web application, but due to its requirements deserves special attention. This feature was developed in Java, using Java Applet technology, due to the possibility of incorporating existing Java APIs, such as graphical components (e.g. JFrame, JPanel and JLabel). It can be embedded in HTML pages, and is supported by any web browser [13]. Figure 2 illustrates the interface where the user performs patient monitoring. In this scenario, six patients are being monitored with sensors for ECG, temperature and HR.

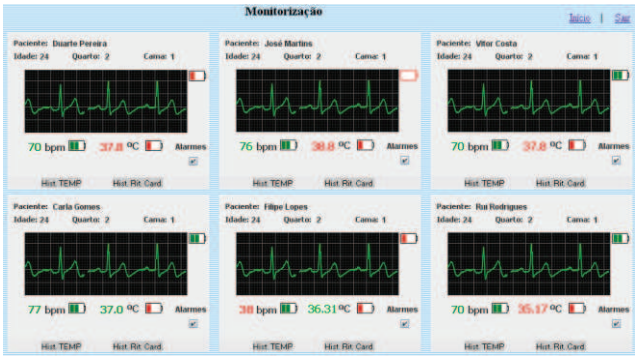

Figure 2: Monitoring System Interface (Portuguese version)

For each patient, the corresponding data is rendered by an embedded Java Applet, which constantly receives data from the central application server.

The database stores all the important information for the operation of the system, including patient information (such as name, birth date, room number, bed number, and any diagnostic information), sensors, and measurements (for each sensor type). It uses MySQL as Database Management System (DBMS) [14], for being multi-user, multi-task and robust, and featuring speed, strength and flexibility to store large amounts of data, such as that produced by ECG sensors [6].

While implementing the system, several challenges and problems were identified for which it was necessary to find the appropriate solutions.

One of the first challenges resulted from the large amount of data produced by each sensor network, and the respective storage of that data in the database in a small amount of time, particularly in the case of the ECG data. The pace at which data is transmitted by each sensor to the DCA server is presented in Table I.

TABLE I. SENSOR DATA RATE

Sensor	*Data Rate*
ECG	250 milliseconds (50 samples)
Oximetry	500 milliseconds
Heart Rate	500 milliseconds
Blood Pressure	10 milliseconds
Temperature	10 milliseconds
Battery charge for each sensor	60 seconds

Although the transmission of the data over the local area (Ethernet) does not raise significant issues, its reception and storing by the DCA is a demanding task. To overcome this challenge, the DCA server resorts to an input queue (a FIFO - First In First Out) to isolate the processes of reception and storing. The reception is supported by a servlet running on Apache Tomcat, and accepts HTTP POST request. When the DCA receives a request from one gateway, it validates the variables that comprise the message sent by the sensor network and store them in the input queue. At the same time, a running thread picks up messages from the queue and stores them in the database, as illustrated in Figure 3. This allows the DCA to be constantly receiving data frames from the sensor networks, regardless of whether or not they entered the database.

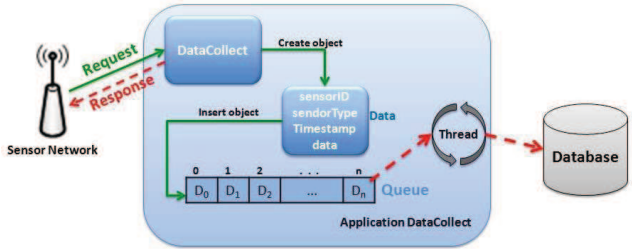

Figure 3: Data Flow of Data Collection Application

Testing the DCA with a large number of simultaneous sensors revealed that the amount of time spent to store the received data into the database was too high. A MySQL DBMS was being used, and the response time was exceeding what was expected given this product specs. Therefore, tests were carried out to determine the maximum number of insertions per unit time, in order to identify the bottleneck. These testes consisted in measuring the time required to insert data into the DB, and are shown in Table II. As shown, for all the three tests, more than one second was being spent for 50 insertions in the database. In conducting 500 insertions, the time spent was approximately ten times larger than before. With these results, it became obvious that storage of the sensors' measurements was not feasible using the current solution. For example, if one takes into account the pace of ECG, HR, and Oximetry sensors, presented in Table I, eight DB inserts are required per second for each patient (four insertions for ECG, two for HR and two for Oximetry). In a scenario of ten patients being monitored, that would result in 80 insertions per second, and this only for the three sensors mentioned above. The database would not be able to respond adequately under this scenario, since at this stage the maximum capability was 40 insertions per second.

TABLE II. DATABASE INSERTION TIME – BEFORE CONFIGURATION

50 insertions in Database			**500 insertions in Database**		
Test	*Time (milliseconds)*		*Test*	*Time (milliseconds)*	
1	Total	1219	1	Total	12204
2	Total	1172	2	Total	11828
3	Total	1203	3	Total	11671

The cause for this slow performance was found to be the logging defaults of MySQL which records every transaction in the log file and saves them to disk immediately. To solve this problem, the option "*innodb_flush_log_at_trx_commit*" was changed to the value of 2. This way, the record in the log records every transaction, but the log is only flushed to disk approximately once per second [15]. With this change, the process of integration become much faster, and the results after these changes are shown in Table III.

TABLE III. DATABASE INSERTION TIME – AFTER CONFIGURATION

50 insertions in Database			**500 insertions in Database**		
Test	*Time (milliseconds)*		*Test*	*Time (milliseconds)*	
1	Total	16	1	Total	125
2	Total	31	2	Total	141
3	Total	31	3	Total	125

As can be seen by comparing the tests carried out before and after modifying the MySQL configuration, the system performance suffered a significant improvement, making it possible to carry about four thousand insertions per second.

Another challenge arose in the communication between the Applet and the Web server, which is supported by the HTTP protocol. The HTTP paradigm is based on the request/response, where the client makes a request and the server, in turn, returns the data to the client and terminates the communication. In this implementation, it was intended that when an applet was initiated, it ordered a monitoring request to the server that, in turn, consulted the DB regularly. When there were new values, they would be sent to the Java Applet through the open communication channel, as shown in Figure 4. This intention has proved difficult with the HTTP protocol, since the communication is closed once the server finishes its answer, and what was intended was for the channel to remain open so the server would be able to send multiple responses to the same application (periodic updates of the indicators). In order to solve this problem, while the applet is running, it receives a

response composed of various parts, using the delimiter character, '\n'. In that way, each delimiter character lets the applet know that the first part is completed, but the server continues to answer because the link has not been terminated. Thus, it is possible for the server to send values to the applet whenever there are updates to the database. This method proved effective in tests and, as such, was the solution selected and implemented.

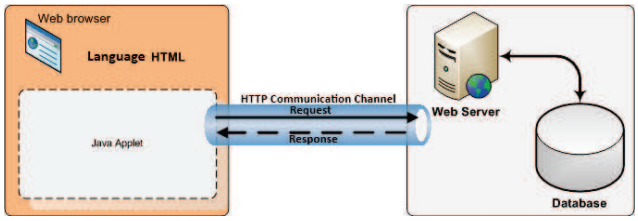

Figure 4: Web Communication between the Applet and Web server

The visualization feature was, undoubtedly, the most complex part of the system. In particular, rendering the ECG waveform accordingly to the medical professionals' requirements showed to be more difficult that was initially expected, which triggered the design of a specific architecture for the applet, as illustrated in Figure 5.

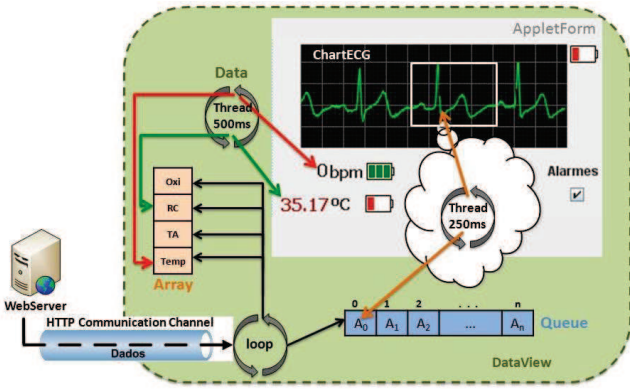

Figure 5: Applet Architecture – Monitoring

The Applet is implemented by the DataView class. When initiated, it creates a HTTP connection with the server and waits for data to arrive. Whenever new data is received, the corresponding values are stored on a list (Array), where each position corresponds to a data type. Concurrently, a thread reads these values (pulse, temperature, BP, HR signals, and the battery level of the sensors) from the array, every 500 milliseconds, and updates the display. For the ECG, the data received from the server is stored in a list (Queue) where each position contains a sample of the ECG time series. Also concurrently, a second thread removes the samples from the queue at a rate of 25 samples every 250 milliseconds and draws the waveform on the screen. The removal of the samples creates room for the new samples arriving from the server.

The most significant challenge was the graphical representation of the ECG in real time. This was because the data are received through the network, and it is necessary to deal with its latency and also with the processing power of the computer where the Applet is running. This underscores the fact that the representation of the ECG signal has strict specifications that are important for its proper implementation. First, it is necessary that the graph represents 2.5 seconds of the ECG signal on a 2.4 millivolts vertical scale. It is also necessary to represent a range that includes two different measurements, as illustrated in Figure 6: 1) small squares that represent 0.1mV/0.04s and 2) large squares (corresponding to small 5x5 squares) that represent 0.5mV/0.2s [16]. Finally, the visual representation must have a refresh rate that emulates the notion of a continuously drawn graphic and a separate line behaviour, that is, when the line reaches the rightmost limit of the graph (the time axis maximum of 2.5 seconds) the representation must return to the beginning of the graph (at 0 seconds), continuously erasing the previously drawn line only as the new one is drawn.

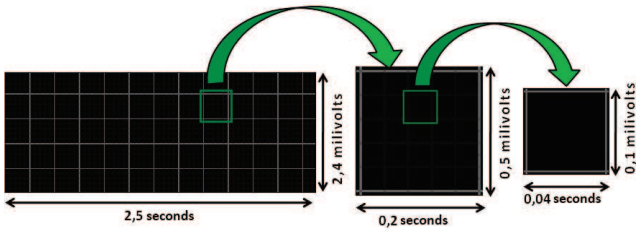

Figure 6: ECG Scale

In order to address the different requirements of ECG representation, we decided first to use free libraries, such as JFreeChart. This is characterized by being 100% free, developed in Java and designed to facilitate the development, containing an extensive list of features [17]. This type of solution was very complex due to the very specific requirements of the ECG chart. This led us to consider that a novel implementation supported by basic drawing would be more feasible. As such, we selected the Graphics2D class in Java API. This class allows the creation of images through the drawing of basic shapes, such as lines, polygons and circles, and its placement on graphical objects, such as the Panel or JPanel. For each monitored patient, an individual ECG is represented with all features described above. This, in a setting of six concurrent patients (the limit set at the beginning of the project), represents high processing requirements on the client's side. However, successful tests were conducted with the final developed solution.

IV. Discussion and Conclusion

During the system's development, many difficulties were encountered and had to be successfully overcome in order to fulfill the initial objectives.

One of the first challenges was to ensure that the server was constantly receiving values from the sensor network. This was achieved using the implementation of an application level protocol based on the HTTP protocol for data communication and the implementation of a Java Servlet on the server side.

The choice of DBMS was also an important point, as a fast and reliable system was needed, particularly for the demanding case of the ECG. MySQL met these requirements with the advantage of being a free solution. However, specific configuration of this DBMS was required.

The Web server used was Apache Tomcat since it supports all the requirements of the project, namely through technologies such as Java Servlet and Java Server Pages. The ECG graphical representation was also a considerable challenge due to the specifications of the representation of the ECG signal. The solution was to the use the Graphics2D class, from Java API, and to implement this representation at the expense of very simple geometric shapes (lines and rectangles).

Another major obstacle was the communication between the applet and the server, because, as stated above, the applet runs on the client and needs to communicate remotely with the server. What was intended was for the applet to make the request for communication to the server, and the server, in turn, sending data to the Applet whenever new data becomes available, through the already open channel of communication. However, this approach became complex when based on the HTTP protocol, since HTTP closes the communication at the end of the response and it was necessary for several responses to be sent to each request. By designing a communication protocol at the application level, where each HTTP response is used to carry asynchronous chunks of data, this problem was solved. The implemented solutions enabled the materialization of the main objective of developing a remote monitoring system. This required the synergic use of Java Servlet technologies, Java Applet, Java Server Pages, MySQL and Apache Tomcat. However, tests have shown that a complex visualization as ECG in real time using a database pooling system (periodic searches) is not the optimal solution. The database requirement was proved to be quite high, since a large number of queries and insertions are made to the database in a fairly short time. It was also concluded that using a Web browser for a complex visualization, such as an ECG signal in real time, might not be the best solution due to the demanding processing requirement on the client machine. This becomes particularly critical with the increase in the number of simultaneously monitored patients. As the number of patients increases, the line representation of the ECG can become slower along time, creating problems with the interpretation of the signal. All these identified features led to the conclusion that the initial system architecture presented is not scalable. Thus, future work should include the analysis of alternative implementations that will scale with the increase in the number of monitored patients.

In the future, we intend to implement other methods that may prove more efficient that the developed solution. One method is to use publish/subscribe technology for data communication. That will prevent the server from periodically checking the database for new values. With this solution, whenever a new value is available, it would be published on the network. In turn, customers would subscribe values they would like to receive. This solution will improve the processing demand (no need for periodic searches to the DB), but it is not known if it will meet response time demands (on the case of the ECG, the value must not take more than 250ms to arrive to the client).

Another future goal is implementing fault tolerance mechanisms on the database, because it is important to ensure that it always remains available to receive data from the sensor network and respond to requests from Web applications. Since the database is a critical component of the system, it is vital to create redundancy. Security is another issue to consider in the future, as the system is sending clinical information through the Internet.

ACKNOWLEDGMENT

Casa de Saúde de Guimarães, through Project MOHLL – Mobile Health Living Lab, under a cooperative development protocol with the University of Minho.

Fundação para a Ciência e a Tecnologia, Lisbon, through the 3° Quadro Comunitário de Apoio, and the POCTI and FEDER programmes.

REFERENCES

[1] A. Hande, T. Polk, W. Walker, and D. Bhatia, “Self-Powered Wireless Sensor Networks for Remote Patient Monitoring in Hospitals”, Sensors, vol 6(9), 2006, pp. 1102-1117.

[2] V. Shnayder, B. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh. Sensor networks for medical care. Harvard University Technical Report TR-08-05, April 2005.

[3] K. Wang, I. Kohane, K. L. Bradshaw, and J. Fackler, “A real time patient monitoring system on the World Wide Web”, Proc. AMIA Annual Fall Symp. 1996, pp. 729–732.

[4] F. Magrabi, N. H. Lovell, and B. G. Celler, “Web based longitudinal ECG monitoring”, Proceedings of the 20th Annual International Conference of the IEEE, Hong Kong, 1998, pp. 1155-1158.

[5] H. S. Lee, S. H. Park, and E. J. Woo, “Remote patient monitoring service through World Wide Web”, Proceedings of the 19th Annual International Conference of the IEEE, Chicago, IL , USA, 1997, pp. 928-931.

[6] M. A. Paracha, S. N. Mohammad, P. W. Macfarlane, and J. M. Jenkins, “Implementation of web database for ECG”. Computers in Cardiology, vol 30, 2003, pp. 271-274.

[7] P. J. C. Pizarro, C. Lopes, R. Moraes, and J. L. B. Marques, “Monitoramento remoto de sinais bioelétricos”, Anais II Congresso Latinoamericano de Engenharia Biomédica, 2001

[8] I. Feghali, R. V. Andreão, M. V. Segatto, “Abordagem na Web para o Telemonitoramento do electrocardiograma de Pacientes Domesticos”, Anais do VI Workshop de Informática Médica, 2006.

[9] H. Fernandez-Lopez, P. Macedo, J.A. Afonso, J. A. Correia, R. Simoes, “Extended health visibility in the hospital environment”, Biodevices 2009 – Second International Conference on Biomedical Electronics and Devices, Porto, Portugal, 2009.

[10] H. Fernandez-Lopez, P. Macedo, J. A. Afonso, J. H. Correia, R. Simões, “Performance analysis of a ZigBee based medical sensor network”, Pervasive Healthcare – 3rd International Conference on Pervasive Computing Technologies for Healthcare, London, UK, 2009.

[11] H. Fernandez-Lopez, P. Macedo, J.A. Afonso, J. H. Correia, R. Simões, “Evaluation of the Impact of the Topology and Hidden Nodes in the Performance of a ZigBee Network”, Proc. 1st Intern. Conf. on Sensor Systems and Software (S-Cube 2009), Pisa, Italy, 2009, pp. 256–271

[12] T. A. S. Foundation (1999). Apache Tomcat, Retrieved Mar. 6, 2009, from http://tomcat.apache.org/

[13] Sun Microsystems (1994). Applets, Retrieved Mar. 11, 2009, from http://java.sun.com/applets/

[14] Oracle Corporation (1995b). MySQL: The world's most popular open source database, Retrieved Jan. 10, 2009, from http://www.mysql.com/

[15] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, et al, “Hypertext Transfer Protocol-HTTP/1.1”, Network Working Group, 1999.

[16] M. A. B. Goncalves, Nocoes Basicas de Eletrocardiograma e Arritmias. Senac, São Paulo, 1995.

[17] O. R. Limited (2005). JFreeChart Retrieved Apr. 16, 2009, from http://www.jfree.org/jfreechart/