Low level communication management for e-health systems

Guillermo Riva, Carlos Zerbini, Javier Voos, Carlos Centeno, Eduardo González
Clinical Engineering Group, National Technological University, Córdoba Regional Faculty
Maestro M López St and Cruz Roja Argentina St, Córdoba, Argentine
E-mail: griva@scdt.frc.utn.edu.ar, czerbini@electronica.frc.utn.edu.ar

Abstract. The heterogeneity of e-health systems encourages the use of standards such as Health Level 7 (HL7v3) to ensure interoperability. Many actual implementations address this problem by unoptimized high level programming of top-range portable computing platforms. However, this approach could pose excessive demands on battery-powered mid-range terminals. In this work, we propose low-level support for portable HL7v3-compatible embedded systems in order to better exploit their limited processing and communications capabilities. In particular, we present our experience in mobile communication management through two different approaches, which proves the feasibility of this proposal.

1. Introduction
E-health systems have been broadly adopted in last years to take advantage of technology for medicine applications. Particularly, telemedicine systems are used for supplying of sanitary attention, prevention, research and education wherever geographic location is a critical factor. Telemedicine is of particular interest in our country due to long geographical distances between medical centres. A generic e-health scenario is shown in Figure 1.

Two main technology advances are driving actual developments on telemedicine. On the one hand, computing resources enable engineers to design standard, smart, and secure data processing and management applications for embedded devices. On the other hand, communication technology enables both increasing bandwidth and mobility, which paves the way for innovative e-health systems. While much work focuses on high-level abstraction implementation of telemedicine systems, we rather discuss low-level design for efficient use of locally available computing and communication resources.

The rest of the paper is organized as follows. In Section 2, we compare communication and digital processing platforms suitable for our purposes. Based on this analysis, in Section 3 we propose approaches to cope with our particular needs. Section 4 presents experiences based on these proposals. Finally, Section 5 reviews obtained results and its application in future work.

2. System Analysis
The presented ideas emerge as enhancements of medical data acquisition and transmission system previously implemented in our laboratory [1]. In order to enable interoperability, on the one hand, adoption of the HL7v3 Standard was proposed [2]. On the other hand, connectivity...
improvements through additional communications technologies are addressed. In the following, we survey available choices for this implementation.

2.1. Communication Technologies

At the moment, our existing platform supports Electrocardiogram (ECG) data transmission to a remote server over Ethernet links. Available communication technologies are discussed in this subsection.

2.1.1. Dial-up Access. Because of its wide availability and robustness to outdated physical transmission media, dial-up access to Internet through the analog Plain Old Telephone System (POTS) is still an option to consider. In this case, a permanent link is first established with a DSL provider through the point-to-point protocol (PPP), which provides an IP address to the user for subsequent data transaction with time-based billing approach. Its very narrow bandwidth restricts its application to low volumes of data, while fixed access to the transmission line restricts mobility.

Our actual application, namely transmission of 12-lead ECG studies, deals with files in the 100 Kilobytes range, therefore dial-up looks like a suitable communications technology. However, adoption of standards such as HL7v3 adds considerably to the volume of transmitted data. As a consequence, Dial-up could turn into an inadequate technology in the near future. From this reasoning, we nowadays consider dial-up only as a last resource.

2.1.2. Ethernet Networks. Ethernet (IEEE 802.3) family of standards, has been for years the preferred technology for local-area networks (LANs), providing relatively high bandwidth, robustness and easy of use. Embedded solutions increasingly include Ethernet as standard communications interface, which enables portable telemedicine devices to attach to any Ethernet socket without the need of additional hardware. In the last years, enhanced Ethernet versions with up to 40-100 Gbps bandwidths are in development for local, metropolitan and wide area networks, which makes this technology a very promising option for wired, nomad Internet access.

2.1.3. Wi-Fi Networks. Wi-Fi (IEEE 802.11) wireless networks enables high-bandwidth interconnection between embedded devices and the Internet using unlicensed bands.

Actually, no open access everywhere is available, even though low-cost network devices are making it increasingly feasible. For example, an ambitious project is in development to provide
open access to Wi-Fi in 66 cities of Córdoba province [3], which pretends to be one of the most important of our country.

Wi-Fi is suitable for medium-capacity battery powered devices. In the case of ultra-low-power mobile e-health devices, however, the IEEE 802.15.4 standard [4] arises as the best option due to its smart modes and enhanced energy management.

2.1.4. Satellite Systems. Satellite-based systems are suitable for areas which lack of any terrestrial communication infrastructure, as well as aircrafts and ships. This communication system is also the only available platform in case of natural disasters such as earthquakes, tornadoes, tsunamis, volcanic eruptions, etc. Main advantages of satellite systems are worldwide coverage and relatively high data transfer speeds [5].

Drawbacks of satellite communication are expensive equipment, dedicated RF links, and need of skilled operators. In addition, high RF power is required in satellite systems with the consequent size and autonomy impact, which restricts its application to very special cases.

2.1.5. Mobile Phone Networks. The General System for Mobile (GSM) technology enables digital communication in not only urban but also rural areas. While 1800/1900 Megahertz (MHz) band is used for short-distance high-bandwidth communication in cities, long-reach 850/900 MHz band is used in rural areas. Available GSM data services are:

- **Circuit-switched Data (CSD).** The CSD service is very similar to a voice call, requiring a dedicated digital channel with a remote system. Its implementation is quite straightforward and requires no digital interfaces which makes it possible to use any mobile phone. However, bandwidth is low (i.e., 9600 bps) and billing is time-based, which restricts it to special applications.

- **Short Message Service (SMS).** SMS service has been widely deployed in recent years for systems such as billing systems and event alert. It can be deployed between mobile stations (MSs) inside the provider’s network or between a MS and external networks by means of a SMS server. It is quite easy to implement and cheap for low data volumes. Drawbacks are no delivery time guarantees and need for multiple messages for mid-sized data volumes [6].

- **General Packet Radio Service (GPRS).** GPRS is the specific GSM service for broadband mobile data transactions. MSs first establish a connection to a Gateway, which serves as interface between GPRS network and internet. From this point, application data can be transparently carried on using the TCP/IP protocol stack, as illustrated in Figure 2. This service was introduced after second generation (2G) GSM, but it wasn’t until recently that it was broadly adopted and implemented as mobile Internet platform in Argentina. Its main advantages are data volume-based billing and enhanced bandwidth, i.e., 56-114 Kbps. Drawbacks are, service quality is strongly time-and place-dependant, latency is high, and cost is not low enough yet in our country.

As demand and competition increases, it is expected that GPRS system is going to evolve to better services and coverage with decreasing costs. In addition, several development platforms include GPRS interface, that’s why it was adopted as communications platform for our work.

2.2. Digital Processing Technologies

Two main requirements drove a review of our processing platform. On the one hand, adoption of the HL7v3 standard demands parsing of Extensive Markup Language (XML) structures requiring dedicated processing and memory. On the other hand, additional processes for communication management are required. As a consequence, the natural question arises about whether to add new tasks to the existing platform or executing them on an external one.
Previous software was based on a PIC18F4550 microcontroller from Microchip, which takes care of user interface, visualization and formatting of raw ECG samples. It also manages Serial Peripheral Interface (SPI) with programmable amplifiers and serial A/D converter, as well as raw data transfer through the isolated serial V.24 port to a Personal Computer (PC). The PIC microcontroller runs 12 Mega Instructions per Second (MIPS) at 40 MHz, and featuring 2 Kbytes of RAM and 32 KBytes of FLASH program memory, which may be enough for HL7 and GPRS modem management. However, in the near future a real-time operating system (RTOS), additional memory and enhanced speed could be needed for additional processing tasks such as ECG storage, compression, encryption, and new protocols support.

Advanced microcontrollers such as ARM cores with appropriate peripherals, meanwhile, can fulfill current needs while providing extra processing power for future applications. In particular, the TS-7200 Single Board Computer (SBC) from Technologic Systems [7] features a powerful Cirrus EP9302 ARM9 processor running 200 MIPS at 200 MHz; Ethernet, SPI, Keypad, USB, V.24, GPIO, and Compact FLASH interfaces; as well as on-board 32Mb DRAM and 8 Mb Strata Flash memories. This processor includes a Memory Management Unit (MMU) which supports high level operating systems such as Linux and Windows CE. Despite all of these features, drawbacks such as cost and power consumption call for a deeper analysis.

![GPRS-based e-health architecture](image)

**Figure 2.** GPRS-based e-health architecture

3. Implementation

3.1. Discussion

Our ECG acquisition platform consists of a 12-lead analog processing stage, followed by a microcontroller which provides extra processing, user interface and transmission to a PC through an isolated standard RS232 serial port. Software running on this PC is in charge of transmitting raw ECG data to a server through Ethernet interfaces. Custom storage and visualization applications, developed in C++, are required for management and later access to ECG studies. Even if this platform proved to be useful for our previous goals, it suffers of some drawbacks. First, it does not support any standard for medical information, which makes it very difficult to interoperate with larger systems. In second place, transmission of raw (i.e., not-compressed) data is bandwidth-inefficient and does not provide enough confidentiality, which is essential while handling patient data. Third, the transmission equipment is based on a PC, which is far much complex and costly hardware than is demanded for our case. Finally, Ethernet technology could not fit mobility requirements for applications such as field data collection, remote consulting and emergencies. We focus on solving these problems while keeping in mind the cost-efficiency tradeoff.
As mentioned, the most appropriate communication solution seems to be GSM/GPRS. Communications infrastructure is already installed and managed by external players, while increasing demand pushes for better service and reduced cost. Multimedia applications are already present in most devices, and some development platforms are suitable for our work. Despite these advantages, developers must cope with some problems before achieving a functional module. On the one hand, development platforms sometimes lack of enough documentation or are launched to market without being completely debugged. On the other hand, interacting with such platforms involves many firmware-specific details.

3.2. GPRS Connectivity Platforms
Main discussion arises about whether to use mobile phones or modems for transmission of medical studies. In the following, we compare the use of mobile phones with respect to two different modems available in Argentina.

3.2.1. Mobile Phones. The first considered platform due to its availability, low cost and proper user interface was a mid-range mobile phone. At the time of our survey, interfaces and protocols used to be proprietary. In addition, coverage could no be enhanced by external directional antennas, which limits its application for e-health services in remote areas.

Although many models offer standard Wi-Fi, Bluetooth and USB interfaces nowadays, they are not intended as interfaces for custom hardware development, and they still do not offer detailed hardware and programming documentation. Therefore additional effort is needed in order to get familiar with every available model.

3.2.2. Siemens TC65T Modem. TC65T combines the Siemens TC65 GSM/GPRS module with a powerful ARM9 microcontroller, which enables applications running on Java 2 Micro Edition (J2ME) to establish so-called Machine-to-Machine (M2M) communications. Quad-band operation is included, which is appropriate for roaming in different countries, and multiple general-purpose I/Os (GPIOs) serve as analog and digital interfaces to external platforms.

Despite all its features, this modem is strongly oriented to stand-alone use and therefore offers poor documentation and difficult programming guidelines for interaction with external devices. Finally, it is rarely adopted in Latin-American e-health projects, and its under-utilized features do not justify its high cost for our project.

3.2.3. Motorola G24 Modem. This modem offers Quad-band operation for international use, CSD, SMS, and GPRS services. The G24 modem also includes an internal TCP/IP stack for FTP, SMTP, SSL, or HTTP data transfer, and Java support is provided in most recent versions. The internal G24 module offers two UARTs, one USB and some general-purpose interfaces, of which only one UART is available in the modem pinout.

Both, our previous PIC-based platform, and a new TS-7200 SBC were used with GPRS modems. Although a TC65 modem was first tested for our implementation, the G24 was finally preferred due to good support from local suppliers, ease of use and low cost. Mentioned features are briefly compared in Table 1.

For transactions between any terminal device (e.g. a PC, our embedded platform, etc) and communication device, namely the GPRS modem, the AT command set is commonly used. These commands are formed by a set of ASCII characters starting with the AT prefix, which stands for attention. By using these commands, both configuration and information requirements are possible. It is to note that most of these commands are standard, while a small subgroup is proprietary of each modem provider [8].
Table 1. GSM/GPRS Modems features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Modem G24</th>
<th>Modem TC65T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Documentation</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Assembly</td>
<td>Argentine</td>
<td>Germany</td>
</tr>
<tr>
<td>Interfaces</td>
<td>V.24</td>
<td>V.24, Audio, I2C, SPI, GPIO</td>
</tr>
<tr>
<td>Power Cons.</td>
<td>2.5 mA - 3.3-4.2V (Sleep)</td>
<td>3 mA - 8-30V (Sleep)</td>
</tr>
<tr>
<td>GPRS Class</td>
<td>Multi-slot Class 10</td>
<td>Multi-slot Class 12</td>
</tr>
<tr>
<td>Embedded TCP/IP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Processor</td>
<td>MC9S08GT32 (G24MCU)</td>
<td>ARM7</td>
</tr>
</tbody>
</table>

3.3. Methodology
We tested transmissions of ECG studies to a HTTP server hosting a web service application through the well-known port 80. For the purpose of this work, previously generated HL7v3 messages were used. We note that the inclusion of additional studies to the existing XML scheme is possible without substantial software modification.

Two different implementation approaches were evaluated during our tests, as shown in Figure 3. The first one uses the TCP/IP stack of the Linux kernel, which we call external stack; while the second one takes advantage of the embedded TCP/IP stack of the G24 modem. The first option is easier to implement, because many open development tools are available for Linux, which raises the abstraction level. On the other hand, the second option allows standalone applications, suitable for constrained-resources portable devices. It is worth to note that the TCP/IP stack could be also embedded in the microcontroller included in our acquisition platform [9], however, it makes no sense to do so considering that the modem includes this functionality.

Both approaches (external and embedded stacks) were evaluated by using three testbenches:

- **Modem connected to PC.** This setting enhances fast testing but it is the more expensive solution. The external (i.e. Linux) TCP/IP stack is used, while any PPP dialer such as wvdial can be applied. This software takes care of modem management, hiding AT commands from the user. For communication through the created socket, any Telnet client can be used, such as Telnet, Putty, etc., see Figure 3(a). We mention that embedded TCP/IP stack could also be tested with this testbench by using any available serial terminal (e.g. Hyperterminal, Putty, etc.), to send application data to the modem, emulating the processing platform behaviour.

- **Modem connected to SBC.** In this case, the Linux kernel must be compiled and ported to the TS-7200 board. The general approach is similar to the previous case (Figure 3(a)), but the limited resources of the SBC imposes the use of command-line tools, which demands additional development.

- **Modem connected to custom acquisition platform.** This is the most optimized testbench regarding use of hardware resources, but it requires most development efforts and work at hardware level (Figure 3(b)). Prior to hardware implementation, simulation tools (e.g. Labcenter Proteus and Microchip MPLab) and C compilers (e.g. Hi-Tech C Compiler) are valuable for this testbench.

While the first testbench was used for fast initial application test, the last two solutions are candidates for our final hardware implementation due to their strong optimization.

Two application-layer protocols were used during tests, Simple Mail Transfer Protocol (SMTP) and HyperText Transfer Protocol (HTTP). SMTP is used for electronic mail
transmission across Internet Protocol (IP) networks using TCP port 25. Most of the documentation about the G24 modem is based on SMTP. HTTP, meanwhile, is a networking protocol for distributed, collaborative, hypermedia information systems communications in a client-server model. Although there are nine methods in HTTP, the two most used are GET and POST. GET requests a representation of the specified resource, and POST submits data to be processed. This protocol will be used in our final implementation for storage of ECG records in a central database, therefore our results are based on it.

4. Results
4.1. Test of the External TCP/IP Stack
In this implementation the `wvdial` software is executed in a host PC on the Linux operating system. First, `wvdial` reads the file `wvdial.conf`, which stores information such as modem initialization commands, Packet Data Protocol Context (i.e., packet data protocol and Provider IP Address), modem interface to be used, etc. After that, it dials the Provider’s Gateway and starts the `pppd` daemon, which thereafter manages transactions between both ends. Finally, a Telnet client is executed in the host PC in order to send HTTP commands to the remote process (i.e., a web service) running on the remote server. The following sequence illustrates this procedure:

```
griva@localadmin-desktop:$ sudo wvdial
--> WvDial: Internet dialer version 1.60
--> Initializing modem. // Modem Initialization
ATZ
OK
--> Sending: ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
OK
--> Sending: AT+CREG=1 // GPRS Network attachment
AT+CREG=1
OK
--> Sending: AT+CUCONT=1,"IP","internet.gprs.unifon.com.ar"
AT+CUCONT=1,"IP","internet.gprs.unifon.com.ar"
OK
--> Modem initialized.
--> Sending: ATDT*99#
--> Waiting for carrier.
ATDT*99#
CONNECT
```
In this approach, AT commands are used only for modem configuration and PPP negotiation in order to establish the connection with the server’s application. From this point on, application data is transferred transparently using the Linux TCP/IP stack, so no further AT commands are necessary.

4.2. Test of the Modem-embedded TCP/IP Stack
This approach, unlike the previous one, makes intensive use of AT commands in order to both establish the PPP negotiation and exchange data with the server application. Two main modes are available for data exchange using the modem-embedded TCP/IP stack:

- **Buffered Data Mode**, using AT+MIPOPEN, AT+MIPSEND, AT+MIPUSH commands
- **Online Data Mode**, which is enabled by using the AT+MIPODM command

![Command sequences: (a) GPRS attachment, (b) buffered mode, (c) online mode](image)

**Figure 4.** Command sequences: (a) GPRS attachment, (b) buffered mode, (c) online mode
As we appreciate in Figure 4(a), an initial negotiation stage with the GPRS Gateway is necessary in both modes. AT+CSQ checks signal quality, before connecting with the Gateway through the AT+MIPCALL command. After successful connection, the Gateway assigns a dynamic IP address to the modem.

In the first mode, AT+MIPOPEN is used to initialize a new socket and open a TCP or UDP connection to a remote server through a specific port. After that, AT+MIPSEND command is required to store application data into the modem buffer. Through AT+MIPPUSH command, data is sent from this buffer to the protocol stack (Figure 4(b)). Drawbacks of this approach are that data must be encoded in hexadecimal format, and segmented in small packet sizes before transmission using AT+MIPSEND. As advantage, this mode allows opening up to 4 simultaneous TCP/IP or UDP/IP connections. The following sequence of commands, which posts a HL7v3 message to our server, illustrates this fact:

```
AT+MIPCALL=1,“internet.gprs.unifon.com.ar”,“wap”,“wap” // Create a wireless link
AT+MIPOPEN=2,3322,“neurotests.frc.utn.edu.ar”,80,0 // Open a TCP socket and connect to server
AT+MIPSSEND=2,“504f535420687474703a2f2f6e6575726f746573” // POST http://neurotes
AT+MIPSSEND=2,“74732e6672632e75746e2e6564752e61722f5465” // ts.frc.utn.edu.ar/Te
... // Keep storing data on 1372-bytes buffer
AT+MIPSSEND=2,“0a3c2f736f6170656e763a456e76656c6f70653e”
AT+MIPSSEND=2,“0d0a0d0a” // End of HTTP POST method
AT+MIPPUSH=2 // Push stored data into TCP/IP protocol stack
```

In the second mode, AT+MIPODM is used to initialize a socket and open a connection with a remote server. This command simplifies data transmission because neither hexadecimal encoding nor segmentation is required (Figure 4(c)). Disadvantage of this case is that only one remote connection can be implemented at a time. The following sequence demonstrates the ease of use of online mode:

```
AT+MIPCALL=1,“internet.gprs.unifon.com.ar”,“wap”,“wap” // Create a wireless link
AT+MIPODM=1,2343,“neurotests.frc.utn.edu.ar”,80,0 // Open a TCP socket and connect to server
POST http://neurotests.frc.utn.edu.ar/TeleMed/services/WSTest HTTP/1.1
Content-Type: text/xml;charset=UTF-8
SOAPAction: "sendMessage"
User-Agent: Jakarta Commons-HttpClient/3.1
Host: neurotests.frc.utn.edu.ar
Content-Length: 271

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
xmlns:test="http://test.ws.gic.com">
<soapenv:Header/>
<soapenv:Body>
<test:message>HL7_message</test:message>
</test:message>
</soapenv:Body>
</soapenv:Envelope>
```

GPRS communication management with our PIC microcontroller-based platform was simulated before firmware download to target hardware. The environment used for this work is the Labcenter Proteus software, which enables direct access to the physical RS232 port of a PC for transmission to the G24 modem (Figure 5). In this way, the exact behavior of the microcontroller was evaluated.
5. Conclusions
In this work we propose taking advantage of actual data processing and communications technologies available in our country for mobile e-health applications. Special emphasis is put on efficient use of constrained resources through low level communication management. We demonstrate the viability of this approach through simulations of our target hardware platform. Future work includes integration to our medical data acquisition system, and debugging of application-level transactions with the process executing at the web server.

Acknowledgment. This work is partially funded by the Universidad Tecnológica Nacional - FONCyT IP-PRH 2007 Posgraduate Grant Program and 25/E134 Project of Ministerio de Educación de la Nación Argentina.

References