Choosing the Right Protocol Stack for an Open and Flexible Remote Unit

Jaime Benjumea, Verónica Medina, Isabel Gómez, Enrique Dorronzoro, Gemma Sánchez, Sergio Martín
Departamento de Tecnología Electrónica
Universidad de Sevilla
benjumea@dte.us.es

***Abstract*-This paper presents some works made in the development of communications software for an embedded open core system. By using a Linux-based processor implemented on a FPGA, we are developing the appropriate software in order to implement a remote unit to be used in a telecontrol network. We present an analysis of the physical devices needed and a performance report of them. After that, we analyze the requirements of the telecontrol network and the possibility of re-using already implemented protocols in Linux instead of using standard telecontrol protocols.**

I. INTRODUCTION

In the field of telecontrol or telemetry networks, the use of closed systems has been common. These systems usually implement a series of protocols specified by the International Electrotechnical Commission (IEC) called IEC-60870 [1]. This document, which specifies a suit of protocols, is divided into six parts and specifies an application-level protocol and a link-layer protocol.

However, as these systems are closed, neither software nor hardware can be freely used. This leads to two problems; first, the user is usually tied to a vendor; second, it is not possible to convert or adapt these systems to a specific need as the user cannot access the source code of software.

Although open source software is widely used in Internet, it is not very common in the telecontrol networks area [2]. The main problem is hardware platforms are also closed so the development of software for them is difficult due to lack of information. However recent developments in FGPA's have made it possible to implement different architectures resembling a standard workstation (a PC, for example). One of these developments has lead to the specification of LEON [3], a SPARC-based processor, which can be implemented on a XILINX Spartan-3 FPGA. The main advantage with this kind of systems (some of them are open cores) is the possibility of developing software using standard development tools.

The "Open Flexible Unit" project (called OFU), funded by Junta de Andalucía and "Multimedia Operatives Techniques applied to Supply Electric Networks" project (called TOMARES), funded by the Ministry of Education and Science of Spain, seek the development of a remote unit using other alternatives, distinct from those usually found in current networks with this kind of devices.

In the OFU project, the main purpose is to develop a flexible unit that may be used in various scenarios with no or little limitations; to make this possible, both hardware and software are open and, as a result, the programming environment used is standard so the software can be easily modified. In fact, since we used Linux as a software development platform, the software development kit (SDK) used was Eclipse [4], a standard SDK available in many platforms.

This paper presents some works made within these two projects. Even though the purpose of the first project is to design a flexible unit, it is a good idea to treat it as a remote unit that will operate on a telecontrol network. By doing that way, we develop something "tangible" that will allow us to focus on a specific scenario. Additionally, our software design could also be used in a telemetry network and other scenarios so, in fact, we are not deviating from initial first project expectations.

This paper is organized as follows; first, an introduction to the available hardware platform (section II) and communication devices (physical layer) used in our tests (section III) is given. Once hardware platform is defined, the raw tests made to physical layer devices (section IV) are described; from these tests, some interesting conclusions are extracted. Section V analyzes the standard (serial-port-based) data-link layer protocol and its possibility of use in our environment (a telemetry/telecontrol network). At the end, some conclusions are presented.

II. HARDWARE PLATFORM

Although hardware platform is out of this paper's interest, it is necessary to give some details because the hardware used will constrain the development of software. Hardware platform is an embedded system, a SoC-type design using FPGA. The FPGA itself has been programmed with an open core called LEON, an SPARC compliant system capable of running Linux

for SPARC. The processor is an open core (i.e. an open hardware) meaning that hardware platform is open, so it is the operating system running over it (Linux Debian for Sparc has been installed in the system [5])

So, the system we are working with is, in essence, a Linux-Sparc system. This means that we may use every program available for this platform and, more important, we are able to do the software development in a very similar way than any standard Linux programming environment.

The hardware has been designed to be very portable (in fact, it should operate as a remote terminal unit) so the final (not the development) system itself is very simple: no Ethernet interface, no VGA card … The only available I/O is a RS232-like port.

III. Choosing a Physical Layer

Given the hardware constrains mentioned before and the portable nature of our system, the next step is to choose an appropriate physical layer for the unit. As it is expected the system will operate in a wide area (e.g. telecontrol/telemetry network system), two alternatives were evaluated. First alternative was to use Radio Frequency (RF) equipment and operate in any ISM available band; the second alternative was to use a standard GSM-based modem.

As RF or GSM considered independently are not always available or desirable, both physical-layers will be used. This means that our software must be prepared to run under both physical-layers. In order to simplify the software development, both systems will be accessed from a RS232 interface which, after all, is the only I/O port available in our hardware platform.

The devices used in the tests were the following:

- RF equipment: An ICOM IC-V82 (VHF transceiver) with the digital unit UT-118. This device is D-star [6] capable, and can be connected to any RS232 device for data transmission. The specifications for this equipment can be found at [7].
- GSM equipment: Wavecom Fastrack M1306B GSM Modem. It is a device behaving as a standard AT-command modem via a RS232 port. According to device's specification, it allows a data transmission (GSM) up to 14.400bps [8] but this feature is dependant on the GSM operator used, so it might not be available (in fact, our tests ran at 9600bps).

We have undergone tests (next section) in order to verify each option usefulness in our scenario. The maximum throughput (i.e. bytes/s) calculation given by device's manufacturer is not appropriate because in our scenario, frame size may be small and, in this case, other factors (as propagation delay and overheading) have a significant impact in effective throughput.

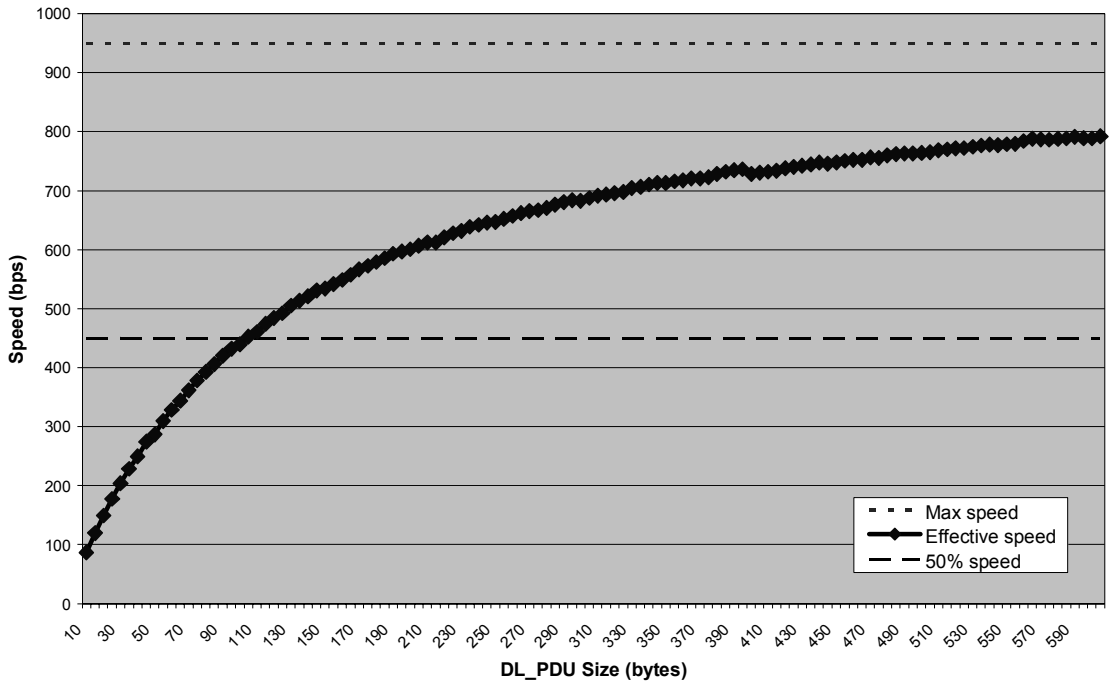

Fig 1. RF-RF throughput graph

IV. TESTING THE PHYSICAL LAYER

In order to test raw throughput in both physical media, we designed a program that allows measuring some important parameters; to be precise we measured the following:

- Real Data Link Protocol Data Unit (DL_PDU) transmission time: This parameter measures the time required to transmit a frame, not considering overhead introduced by the physical layer.
- Observed (effective) DL_PDU transmission time: This parameter measures the time required to transmit a frame but in this case, considering the overhead introduced by the physical layer. This parameter is important because in the case of RF, the physical layer adds some headings (preamble and synchronization stuff) which size might be significant compared with upper-layers data size. As we measure time required for a frame to be received by the other side, effective DL_PDU transmission time also includes propagation delay which in some cases, may be significant. So, this parameter allows analyzing how a physical layer behaves when handling small data.

Experimental data shown is used to analyze how a physical layer behaves when handling small data; this is important because data size in telecontrol network is about 250 bytes. The data obtained are in figures 1 and 2 and allows coming to some interesting conclusions:

- Experimental data for RF: Observed (i.e. effective) transmission speed for RF is shown in figure 1, is poor. Some references from the manufacture [9] states that maximum achievable speed for this band is 950bps. However, our tests indicate that DL_PDU of sizes less than 100bytes, are effectively transmitted at no more than 450bps (efficiency is about 50%). As the device adds not only physical layer overhead but also data-link layer protocol overhead (D-star protocol), further testing must be carried out with other devices implementing this or other data-link protocol, to see if the same behavior is present.
- Experimental data for GSM: Effective transmission speed for GSM (figure 2) is not good either, 50% efficiency is only reached with DL_PDU larger than 500 bytes which exceeds the 250 bytes of our typical DL_PDU size. In this case, the poor performance is caused by the propagation delay associated with the use of a GSM network; our tests indicate a propagation delay in the range of 250-500ms. If we consider that 250 bytes at 9600 bps are transmitted in about 200ms, the propagation delay introduced is too long.

So given the experimental data obtained, it seems that with a DL_PDU payload (data) of 250 bytes, neither of the systems are very efficient. RF systems are inherently slow whereas GSM might be faster but it is not very efficient. Anyway, GSM

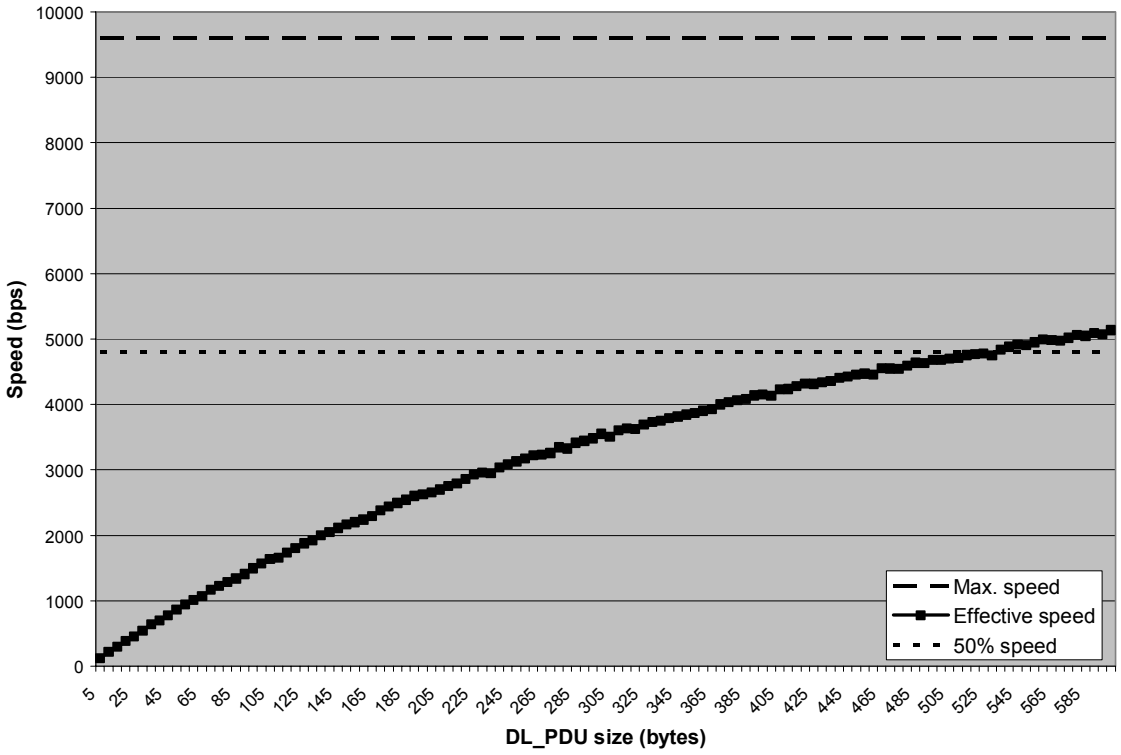

Fig 2. GSM-GSM throughput graph

is still capable of effectively transmitting DL_PDU faster than RF equipments used.

However, it is important to notice that propagation delay observed in GSM is very high so, in some cases, GSM could achieve worse results than RF (for small frames, for example). In fact, some preliminary tests made with other RF equipments show us that, for small frames, RF might achieve a better transmission speed than GSM. This behavior must be taken into consideration in the following stages of the project (now, it is too early to know the exact frame average size).

V. CHOOSING THE PROTOCOL STACK

In order to make the remote unit to work as a node in a telecontrol or telemetry networks, a communication system must be implemented on it, so one can access every node (remote unit) from a centralized location. There is a protocol designed specifically for this scenario; the IEC60870-series, mentioned in the introductory section, specify some protocols to be used in electrical telecontrol networks. These protocols are used to transmit telecontrol information within an electric utility and, on the other hand, might be used in any scenario in which there is a need to interact between some remote units and a centralized node (so the remote unit still has its flexible ability, as required by OFU project).

So, instead of trying to design a new protocol, we have decided to adapt the IEC protocols in order to suit the requirements of the projects mentioned before. The IEC protocols cover the application and the data-link layers in the OSI model. However, it might be not necessary to implement or use both because we could use a data-link layer provided by the operating system installed in our hardware (Linux Debian).

In addition, current IEC protocols implementations are not open source so we must implement by ourselves, at least, the application-layer of the IEC protocols. As for the lower-level layers, a decision must be made: either use Linux networking Application Programming Interface (API) or implement these layers using the IEC specification.

Using Linux as base for the software development gives us a potential that we usually do not find in other embedded systems. This means that we could use all the software already developed and tested including any existing API present in the Linux Operating System. The main advantage using this option is the fact that Linux distribution used is completely open source. What is more, there is a TCP/IP specification of the IEC protocols that could be used directly over the standard Linux TCP/IP stack.

Next step is to analyze if it is possible to use this API within our restricted system. Given the fact that our interface with outside world (i.e. our “network interface”) is a serial port, the appropriate link-layer protocol to be used is the Point-to-Point Protocol (PPP) [10] as it is the standard way to communicate two computers running Linux thru a serial port. Additionally, although PPP is the natural data-link layer protocol for TCP/IP over serial links, it could be used as a data-link layer protocol by itself (i.e. with any network layer protocol). But before programming anything some tests must be done in both interfaces.

A. GSM interface

This interface consists of a standard GSM modem connected via RS232 to our system. The modem itself behaves as a standard AT-command modem so it is possible to use standard Debian applications directly. Speed is restricted to 9600 bps and propagation delay is very high, so results obtained are not very compelling; in any case we did some testing.

Preliminary tests shown that speed being achievable were in the range from 3Kbps to 7Kbps using TCP/IP. These results were not conclusive because in these tests the GSM-carrier IP network was used (in the final system, the remote unit would call the centralized location). While it is true that more tests should be done, the use of PPP was dropped after the results shown with RF equipment (see below), so more tests using PPP and GSM were not necessary.

B. RF interface

As we said in section IV, RF interface used introduces a high overhead due to the use of the D-star protocol. But in this case we are not testing RF throughput but its ability to be used with the PPP implemented under Linux.

In contrast with GSM, RF equipments operate in Half-duplex mode meaning it is not possible to transmit and receive simultaneously. This is crucial because PPP requires a Full-duplex interface in order to operate correctly.

Tests made show that it is possible to use PPP and RF but if and only if there are only two nodes in the same radio frequency and application-layer is aware that a Half-duplex medium is on operation. This means that although it is possible to use PPP in some scenarios where network use is somehow similar to a stop-and-wait protocol, under general conditions it would not be possible to use PPP in a Half-duplex channel.

No RF throughput tests (with PPP) were made because it was clear that the main trouble is PPP inability to operate in a Half-duplex scenario.

C. Discarding PPP

PPP can not be used without being modified in a Half-duplex scenario. This is true, regardless of the throughput we obtained in section IV[1]. Throughput tests will be useful in the upcoming stages of our project (as we said before), but they are irrelevant to the fact that PPP can not be used in Half-duplex.

So, although PPP could be used in GSM, it is not possible to use Linux PPP implementation in RF. As it is advisable to use a unique data-link layer protocol, it seems appropriate to discard the use of Linux PPP in both scenarios (GSM and RF).

In order to implement a data-link-layer protocol, there are two options: (a) modify PPP implementation and make it compliant with a Half-duplex physical medium or (b) implement IEC data-link-layer.

Both options require to do some programming, in the first case (a), the modification consists in activating the use of P/F bit within a PPP-frame, this means modifying PPP specification; in the second case (b), the implementation of the IEC data-link layer requires to begin from zero. But even though time effort could be greater in case (b), our decision was to implement the IEC data-link layer protocol.

The main reason to do so is the fact that the IEC data-link protocol is specifically designed for our scenario and PPP is not. PPP would have been a good option if it did not require to write a line of code but this is not the case. We believe that once a programming effort must be done, it is much better to concentrate on a protocol implementation specifically designed for our scenario and not to try modifying an already implemented protocol. It is important to notice that there was a chance that once modified, the Full-duplex PPP were still not usable in our scenario.

Currently, the IEC data-link layer implementation is under test and the IEC application layer implementation is under development.

VI. Conclusions

Choosing the right protocol stack for an embedded system to be used in a telecontrol or telemetry network is crucial for the development of two research projects we are involved in. As one of the project's constrains is that software in use must be open source, we can not use already implemented protocols due to its closed nature.

On the other hand, the IEC-60870 protocol suit is an appropriate application level protocol because it is specifically designed for telecontrol networks. But the data-link layer could be any of data-link layer protocols already implemented under Linux.

We have shown that the standard PPP cannot be used as data-link layer protocol in our scenario because PPP is not designed to be used in a Half-duplex physical layer. This prohibits the use of PPP if physical layer is a RF system and as a result, the use of PPP for the whole system is discouraged.

On the other hand, modifying PPP or using some link layer protocols as X.25, Amateur X.25 or HDLC, means adapting (reprogram) parts of the original Linux implementation. As IEC protocols specify a data-link layer protocol designed for this specific scenario, we believe it is more appropriate to implement both the IEC data-link protocol and application protocol.

As the IEC data-link protocol already implements any service required by the upper layers, there is no need to have a device that implements its own data-link layer protocol. This is important for RF devices because commercially available RF devices (radio-modems) usually implement a data-link layer protocol on their own which, for our scenario, only adds useless overhead. By using the IEC data-link protocol, the only heading needed is the necessary to support physical layer synchronization.

On the other hand, it is important to mention that GSM high propagation delay could make this device slower (in effective terms) than certain RF equipments, especially if frame size is small. At the current stage of our project, it is not possible to know if this will be an issue or not.

Finally, the use of Linux on the final system has allowed us to make the initial software development in a standard PC, using a standard SDK. Also, initial tests were made with a PC (and not the final system) because all programming used standard C I/O functions. Only in final tests the same source code program was cross-compiled in the PC in order to be transferred into the final system. This is good because there is no need to compile software on the FPGA itself.

To sum up, the stack of protocols to be used in our scenario will consist on implementing both application and data-link of the IEC suit instead of trying to adapt already implemented link-layer protocols. On the other hand, as the implemented data-link protocol offers all services required by the application layer, it is not necessary (and even not advisable) to use equipment that already uses a data-link protocol. However, it might be difficult to find commercial RF equipment that implements no data-link protocol at all. Finally, GSM propagation delay must not be dismissed and should be taken into consideration if we use faster RF-equipment (as RF could surpass GSM for small frame size).

[1] It is worth mentioning that our tests were made using a half-duplex application layer.

ACKNOWLEDGMENTS

This work has been undertaken in the framework of two research projects: OFU (EXC-2005-TIC-1023) - Open Flexible Unit funded by Junta de Andalucía and TOMARES (TEC2006-08430) -Multimedia Operatives Techniques applied to Supply Electric Networks funded by the Ministry of Education and Science of Spain.

REFERENCES

[1] International Electrotechnical comission, "International Standard IEC-60870-5" (6 parts).

[2] Jaime Benjumea, Francisco Pérez, Joaquín Luque: "Encouraging the use of Open Source Software in high-sensitive environments", CIGRE, Study Committee D.2, Colloquium. Rio de Janeiro (Brasil), Sep, 2003.

[3] "GRLIB/LEON3 manual", http://www.gaisler.com

[4] "Eclipse - an open development platform", http://www.eclipse.org

[5] A. Muñoz, E. Ostúa, P. Ruiz, M. J. Bellido, J. Viejo, A. Millán, J. Juan, D. Guerrero, "Un ejemplo de implemetación de una distribución Linux en un SoC basado en hardware Linux", Actas de las IV jornadas de computación reconfigurable y aplicaciones (JCRA'07), pp. 85-92, Sep-2007.

[6] "Dstar system", http://www.arrl.org/FandES/field/regulations/techchar/D-STAR.pdf

[7] "ICOM IC-V82 brochure", http://www.icom.co.jp/world/products/pdf/IC-V82_U82_LM.pdf

[8] "Wavecom Fastrack 1306M User Manual", http://www.wavecom.com/media/files/support/Hard_platforms/Modems/Fastrack_M1306B/User_manual/Fastrack_M1306B_User_Guide_rev003.pdf

[9] "ICOM D-star technical specification", http://www.icomcanada.com/dstar/dstar7.htm

[10] W. Simpson, "The Point-to-Point Protocol (PPP)", ftp://ftp.rfc-editor.org/in-notes/rfc1661.txt