ELSEVIER

Sensors and Actuators A 101 (2002) 194–202

SENSORS AND ACTUATORS A PHYSICAL

www.elsevier.com/locate/sna

Sensor interfaces: from field-bus to Ethernet and Internet

A. Flammini*, P. Ferrari, E. Sisinni, D. Marioli, A. Taroni

Department of Electronics for Automation, University of Brescia and INFM, Via Branze 38, 25123 Brescia, Italy

Received 8 November 2001; received in revised form 27 May 2002; accepted 12 June 2002

Abstract

Traditionally, the 4–20 mA analog signal used in industry to interface sensors in distributed process control has been replaced with relatively simple digital networks, called ''field-buses'', and recently by Ethernet. In this work, three interfaces for a simple sensor, based on the same architecture and microcontroller, have been implemented and tested: Profibus-DP, CANbus2.0B and Ethernet IEEE802.3. The experimental results show that the Ethernet interface, although less efficient than field-bus interfaces, requires about the same resources, in terms of computing power and memory space for code and data storage. As an Internet-capable sensor could be virtually reached from everywhere, for remote control and diagnosis, the feasibility of an Internet interface has been investigated. A full-compliant implementation of high-level protocols as TCP/IP or HTTP needs many resources to be implemented, resulting too expensive for a ''smart sensor''. For this reason, a simple UDP-based solution, that preserves Internet visibility, has been proposed and implemented. Obtained results match with a low-cost smart Internet-sensor.

Keywords: Sensors; Fieldbus; Ethernet; Internet

1. Introduction

Nowadays, many field devices have a bit of computing power allowing each device to be ''smart'' [1] and digitally interfaced [2]. Field devices, as sensors or transducers, usually make use of field-buses, while control devices, as industrial PC, PLC or SCADA, are often interconnected using Ethernet IEEE802.3 [3,4]. There are several open standards with pros and cons describing these networks; for instance, CANbus2.0B (ISO 11519 and ISO 11898) [5] or Profibus-DP (EN50170) [6,7] are quite simple and can be easily integrated in low-cost microcontrollers, reducing the need for external components (e.g. Motorola 68HC05 or Microchip PIC18Cx5x, that provide a CANbus2.0B interface).

Despite the effort to develop standardized connectivity for smart transducers to new or existing networks, by IEEE Standard 1451 [8–10], the existence of several ''multivendor field-buses'' and the appearance of powerful devices with an Ethernet controller (like National Instruments FieldPoint, Web cameras, etc.), have made bridging necessary, as depicted in Fig. 1, thus increasing the overall cost. Moreover, these cross-points could result in a bottleneck for data stream.

Ethernet has been used in factory communication systems at plant and cell levels, but the earlier version was not able to satisfy the real-time requirements needed by sensor/actuator levels. Recently, new improvements in data transfer speed and the introduction of switches practically have overcome limits due to non-determinism of Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocols [11]. Several Ethernet compatible protocols have been proposed to support real-time traffic [12]. Implementing Ethernet even at the lowest level is suitable for a better interface to the last-generation PLCs [13], as Soft-PLC, allowing the reuse of existing infrastructures. Standard protocols, as Internet Protocol (IP) [14], User Datagram Protocol (UDP) [15] or Transfer Control Protocol (TCP) [16] should be used; the obvious evolution of this approach is the direct visibility of sensors in Intranet or Internet [17]. In this way, it is possible to remotely acquire real-world data from production equipment: data may be used to control a process or machine, be saved for later processing, or be graphically displayed for a human operator. The main obstacles are low-efficiency of Ethernet-based protocols when few data are transmitted and, mainly, the cost. In fact, although there are single-chip Ethernet interfaces as cheap as the field-bus ones, a full-compliant implementation of high-level protocols,

* Corresponding author. Tel.: +39-30-3715627; fax: +39-30-380014.
E-mail address: flammini@ing.unibs.it (A. Flammini).

0924-4247/02/$ – see front matter
PII: S0924-4247(02)00201-7

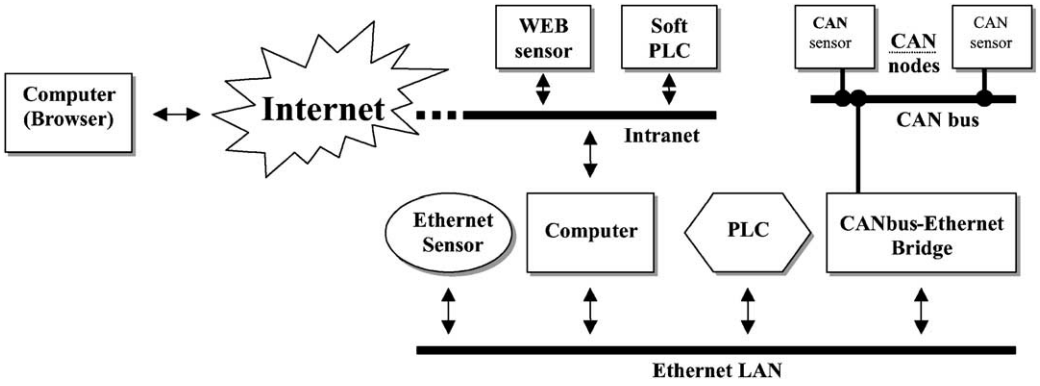

Fig. 1. Sensor interconnection: Fieldbus, Ethernet and Internet.

such as IP, UDP or TCP, needs high-performance processors and requires a lot of resources, as memory for code and data storage.

Our purpose is to verify the feasibility of an Internet-capable sensor whose cost is comparable with a sensor interfaced to a field-bus. In the following section, an experimental comparison among Ethernet, CANbus2.0B and Profibus-DP is presented; then, main problems related to Internet interfacing are analyzed, and in the end, a solution is proposed to obtain a low-cost Internet-enabled sensor.

2. A comparison among the considered protocols

Actually field-buses are preferred to Ethernet-based protocols thanks to their simplicity, robustness, efficiency and low-cost. In the following paragraphs, cost and characteristics of sensors interconnected to Profibus-DP, CANbus2.0B and Ethernet IEEE802.3 are compared. Field-bus protocols and Ethernet IEEE802.3 define only the Physical and Data Link layers of the ISO/OSI stack. For this reason, all these networks support one or more upper-layer protocols that run on top of it, providing sophisticated data transfer and network management functionality. An overview of several configurations is shown in Table 1, referring to ISO/OSI standard communication levels [18].

Physical level is generally structured in more sub-layers. If Ethernet is considered, medium dependent interface (MDI) and physical medium attachment (PMA) define the transceiver, attachment unit interface (AUI) defines the cable and Physical-layer signaling (PLS) is devoted to the Data Link-layer interface. Regarding the Data Link layer, access to the shared channel is determined by the Medium Access Control (MAC), while Logical Link Control (LLC) manages the interface to Network layer. It should be said that, in the industrial context, proprietary protocols are often built ‘‘ad hoc’’ over the Data Link layer, the Network layer or the Transport layer.

Referring to Ethernet, MAC sub-layer is based on a system called Carrier Sense Multiple Access with Collision Detection. Each node must wait until there is no signal on the channel, and then it can begin transmitting. If some other interface starts simultaneously to transmit there will be a collision. Every station then stops transmitting and remains silent for a random period of time before attempting to transmit again. This process is repeated until the frame is eventually transmitted successfully: this leads to unpredictable waiting time that, if high-traffic affects the network, is not suitable for the sensors/actuators level. The use of switches allows implementation of Ethernet networks in which collisions between frames are practically absent. In addition, the typical organization of devices avoids collision: usually, a controller sequentially polls a device at a time waiting for response before passing to the next.

The Ethernet frame consists of a set of bits organized into several fields: it has a minimum length of 72 octets and allows a payload of 1500 bytes (1526 octets), plus 12 octets of inter-frame gap. If we define T_{bit} as the time required to transmit one bit, that is $T_{bit} = 1/\text{bit-rate}$, d as the number of data-bytes that contains information and T as the total time

Table 1
ISO/OSI levels of considered protocols

Layer	Profibus-DP	CAN2.0B	Ethernet	Internet
User	DDLM			
Application		CANopen DeviceNET		FTP HTTP
Presentation				
Session				
Transport				TCP/IP UDP/IP
Network				IP ARP ICMP
Data Link	FDL	LLC MAC	LLC MAC	
Physical	RS485	PLS PMA MDI	PLS AUI PMA MDI	

required by the selected protocol to send d data-bytes, Eq. (1) results. It should be noticed that T_{Eth} remains the same for information encoded by data d from 2 to 46 bytes. As d-value is rather small for sensors into consideration, T_{Eth} can be considered equal to $672 \times T_{\text{bit}}$:

$$T_{\text{Eth}} = [(8 + 14 + \gamma + 4) + 12] \times 8 \times T_{\text{bit}},$$
$$\gamma = \begin{cases} 46 & (d \leq 46) \\ d & (d > 46) \end{cases} \quad (1)$$

If CANbus2.0B is considered, the used bus access method is a non-destructive bit-wise arbitration, called Carrier Sense Multiple Access with Collision Avoiding (CSMA/CA). Bus allocation is provided on the basis of need, with minimum delay and with maximum possible utilization of the available bus capacity: in fact the dominant message is sent anyway. Hardware synchronization is reached using a mechanism called "bit-stuffing": if five consecutive identical bits have been transmitted, the transmitter will automatically inject (stuff) a bit of the opposite polarity into the bit stream. Receivers will automatically delete (de-stuff) such bits before processing the message in any way. The effective bit-rate is decreased by a factor that, in the worst case, is equal to 4/5 [19]. Some fields are not affected by bit-stuffing and we indicate as n_{stuff} the maximum number of bits stuffed. CAN2.0B has a maximum data-field of 8 bytes within a 154 bits frame (worst case). The maximum time T_{CAN} to transmit the d-bytes wide information $(d \leq 8)$ is indicated in Eq. (2):

$$T_{\text{CAN}} = (67 + d \times 8 + n_{\text{stuff}}) \times T_{\text{bit}} \quad (2)$$

ProfiBus-DP, instead, is a master–slave network and it utilizes a two-wire RS485 physical medium. Up to 32 stations can be connected in one segment, but the bus can be easily expanded using inexpensive line amplifiers (repeaters). Maximum bit-rates of 12 Mbit/s can be accommodated by segments up to 100 m. The Fieldbus Data Link layer (FDL) consists of two sub-layers: the lower portion is the Fieldbus Media Access Control (FMAC) and the upper portion is the Fieldbus Data Link Control (FDLC). The Fieldbus Media Access is a fusion of the token passing and polling principles. Several devices on a network may be master stations. Only the station that has the token is permitted to initiate communication. The master may poll (master requests, slave responds) the slave devices (e.g. a sensor) while it has the token. For that reason it is hard to speak in terms of data-rate. Considering a simple network implementation made up of one master and one slave, in order to receive sensor data, two messages must be sent: the master request and the slave reply. The total cycle time $T_{\text{P-DP}}$ can be seen as the sum of the request time (T_{req}), the reply time (T_{rep}) and the inter-frame gap (T_{i}) as in Eq. (3):

$$T_{\text{P-DP}} = T_{\text{req}} + T_{\text{rep}} + T_{\text{i}} = 6 \times 11 \times T_{\text{bit}} + (9 + d) \times 11 \times T_{\text{bit}} + 48 \times T_{\text{bit}} \quad (3)$$

The Direct Data Link Mapper (DDLM) provides the user interface easy access to layer 2 (ISO/OSI Data Link). The application functions, which are available to the user as well as the system and device behavior of the various Profibus-DP device types, are specified in the user interface. Profibus is a proprietary standard, instead of CAN and Ethernet, and slave interfaces must be certified.

A useful feature, named efficiency η, can be defined as depicted in Eq. (4):

$$\eta = \frac{d \times 8 \times T_{\text{bit}}}{T} \quad (4)$$

Figs. 2 and 3 show how the total time T and efficiency η change as a function of the number d of useful data-bytes $(0 \leq d \leq 8)$ for the three considered protocols. Fig. 2 has been obtained considering typical bit-rates: 1 Mbit/s for

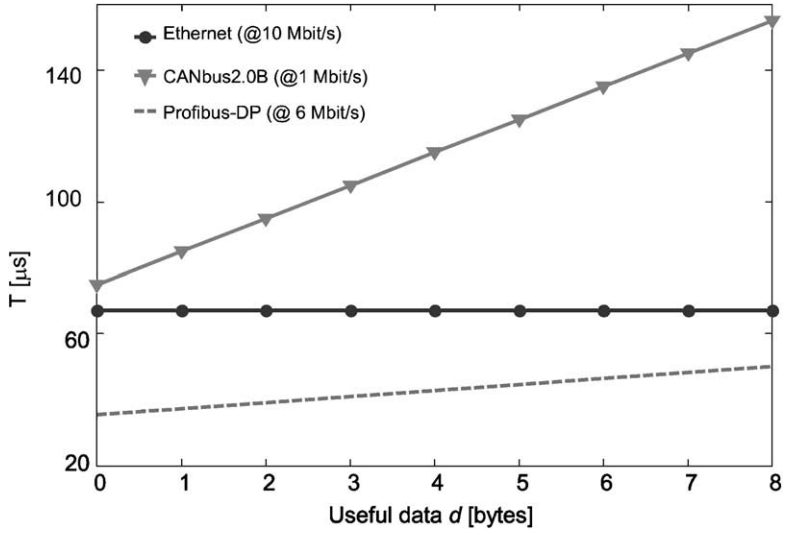

Fig. 2. Total time T as a function of the number d of the useful data-bytes for the considered protocols.

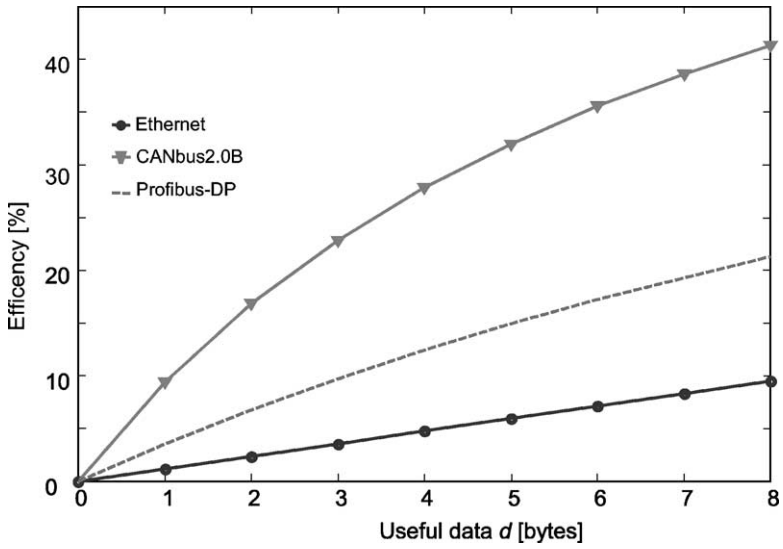

Fig. 3. Efficiency η as a function of the number d of the useful data-bytes for the considered protocols.

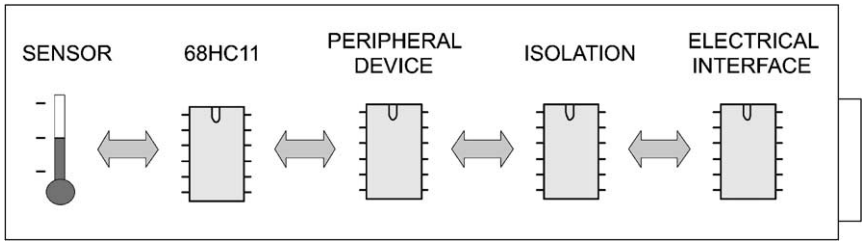

Fig. 4. A simple network-enabled sensor.

CANbus2.0B, 6 Mbit/s for Profibus-DP and 10 Mbit/s for Ethernet (10BASE-T [4], that specifies 10 MHz as the bit-rate and a twisted-pair cable as medium).

The d-value is small for a simple sensor and the Ethernet link is characterized by the lowest efficiency η, as Fig. 3 shows, due to protocol overhead. In industrial applications, a sensor network is generally characterized by the cycle time, that depends on the overall time T and the number of sensors. As shown in Fig. 2, the overall time T is comparable for the three considered protocols, yielding to similar values of the frame-rate, equal to $1/T$. In fact, efficiency is independent of bit-rate and that advantages Ethernet, thanks to its improving technology (10 and 100 MHz, 1 GHz).

To evaluate experimentally costs and performance of the three analyzed protocols, we have considered a humidity sensor, where the data to be sent is a two ASCII character set $\{00 \div 99\}$ representing the percentage of humidity ($d = 2$). To acquire signal, manage and transmit information, a widespread microcontroller, the 8-bit Motorola MC68HC11, has been provided. Nowadays, single-chip implementation of ISO/OSI layers 1 and 2 for Ethernet, CANbus2.0B and Profibus-DP are available as low-cost peripheral devices that can be easily interfaced to a microcontroller. It is important to say that these peripheral devices handle all aspects of Physical and Data Link layers (ISO/OSI 1 and 2), including baseband signal and checksum generation.

Consequently it is possible to develop a network-enabled sensor only adding the isolation and the electrical interface devices. In this way, we have realized three small cards: a CANbus2.0B interface, a Profibus-DP interface and an

Table 2
Overview of the devices used in the three network-enabled sensors

	Peripheral device	Isolation	Electrical Interface
Ethernet (@ 10 Mbit/s)	Crystal CS8900A ($f_{CLK} = 20$ MHz, 4KB RAM)	Halo TG43-1406 mini transformer	Halo TG43-1406 mini transformer
CANbus2.0B (@ 1 Mbit/s)	Intel AN82527 ($f_{CLK} = 16$ MHz, 256 bytes RAM)	6N137 optical isolator	Philips PCA 82C250 CAN transceiver
Profibus-DP (@ 6 Mbit/s)	Siemens SPC3 ($f_{CLK} = 48$ MHz, 1.5KB of RAM)	HPCL7100 optical isolator	MAX485 RS485 transceiver

Table 3
Memory resource utilization

	Memory resource (bytes)	
	Code	RAM
Ethernet	700	15
CANbus2.0B	650	50
Profibus-DP	10200	150

Ethernet interface, all based on the same architecture, as shown in Fig. 4. Table 2 summarizes hardware devices that have been used. These three inexpensive implementations are equivalent in terms of components cost.

Since the same microcontroller has been used, an estimate of the computational cost, in terms of memory occupation, can be performed, distinguishing between program memory size and data-RAM utilization, as reported in Table 3.

If memory requirements are analyzed, the Ethernet interface seems to be competitive with the widespread and simple CANbus2.0B interface. Profibus-DP is a quite heavy interface even in the slave configuration, and hence today there are no microcontrollers with an on-chip Profibus-DP interface. Only PLCs or dedicated hubs use this kind of field-bus, while sensors are connected to a low-cost I/O module, controlled by a resident firmware. This kind of configuration makes it possible to subdivide the high-cost of the Profibus-DP hub among a large number of low-cost sensors.

In conclusion, the Ethernet interface is suitable, in terms of costs and performances, not only for complex sensors, as an industrial camera, but even for those sensors that are interfaced to field-buses. Anyway, it must be remembered that the experimental results are limited to the first two ISO/OSI levels implementation. As we said, main difference between field-buses and Ethernet is that the latter supports standard and widespread high-level protocols (IP, TCP, UDP) that can access Internet.

3. Interface to Internet

To allow visibility from Internet, basic protocols are Internet Protocol, Address Resolution Protocol (ARP) [20] and Internet Control Message Protocol (ICMP) [21]. Their most important features are relatively easy to implement in a low-cost microcontroller, in terms of memory resources and time utilization. A proprietary protocol could be built over IP, but a standard Transport-layer protocol, as Transfer Control Protocol or User Datagram Protocol, is preferable to achieve a better interface to PLCs and industrial computers. Fig. 5 shows how these protocols are built over an Ethernet packet, where header fields of IP, TCP and UDP are represented with letter H. For IP and TCP, the header is composed by a fixed-length part (20 octets) plus the option field, ranging from 0 to 40 octets. Comparing the header sizes, UDP is clearly a simpler and more compact protocol than TCP.

Referring to the total time T discussed in the previous section, Ethernet or Ethernet plus IP, UDP or TCP are comparable, even when maximum length for header fields H is considered ($T_{\mathrm{Eth}} = 67.2\ \mu s$, $T_{\mathrm{UDP}} = 69.6\ \mu s$, $T_{\mathrm{TCP}} = 78.4\ \mu s$ @ $d = 2$, 10BASE-T). On the contrary, required computing power and memory space are very different for a simple proprietary protocol based on IEEE802.3 packets [11], a multi-connection system over TCP/IP [22] or even a Web-server accessible from Internet both in read and write mode [17].

The overall cost depends on the required compliance level: a full-compliant implementation of Internet protocols (IP and UDP or TCP) in embedded systems must deal with several problems like IP fragmentation [14] or multiple TCP connection handling.

IP packets, that have a maximum datagram length of 65,535 octets (64KB), are composed by data plus a header with a variable length from 20 to 60 bytes depending on option field. IP packets travel on different physical networks and IP routers have the ability of fragmenting a packet if it is bigger than the maximum transmission unit (MTU) of some network segment, keeping a minimum IP fragment length equal to 68 bytes. This adds the facility to send packets larger than maximum size permitted by network hardware, but it results in a performance penalty and complicates management at the receiver end. In fact, fragments of the same packet could arrive untidily and must be reassembled before any operation takes place. Thus extra RAM for temporary fragment-buffer is needed. PCs and other complex systems, that have a large amount of RAM, can handle a virtually unlimited number of connections with a large-at-will buffer for each one, while IP fragmentation recovery is

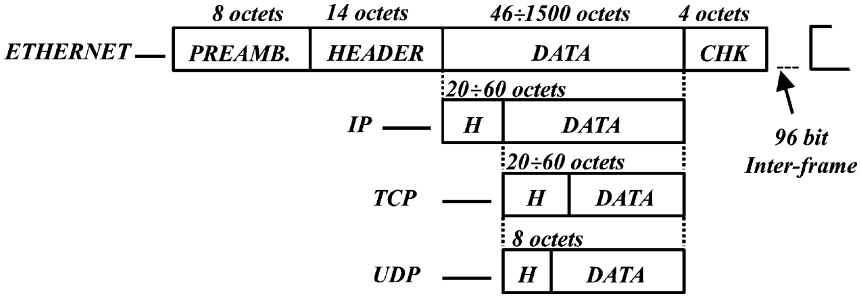

Fig. 5. Ethernet packet.

easily accomplished. On the other hand, low-cost microcontrollers have poor resources, in particular an on-chip RAM less than 1KB. The problem of IP fragmentation is related only with data d incoming to sensor and it should be said that messages sent to a simple sensor are rather small, that is $d < 10$ bytes. Obviously, the IP data-field is composed by TCP or UDP header plus data d. If $d \neq 0$, fragmentation can occur, because IP header can be 60-byte long, TCP header has a minimum length of 20 bytes and UDP header has a fixed length of 8 bytes, therefore total IP packet length can exceed the 68 bytes limit. If several systems send fragmented data simultaneously to the sensor, too many fragments have to be stored simultaneously and the limited amount of RAM yields to lose messages.

Regarding the choice of a Transport-layer protocol suitable for simple sensors, many factors have to be taken into account. TCP guarantees a more reliable connection, performing several tasks at once, as negotiating the connection, providing flow control and handling network failure. Of course, TCP packets needed for managing the connection, which have not been included in the previous calculus because they are not known "a priori", decrease the overall efficiency. One of the main problems to implement a TCP stack is the large quantity of data memory necessary to store messages waiting for acknowledgment: these buffers have to be provided for every simultaneous connection managed by the sensor. Moreover, additional RAM should be provided to store connection-related information, as IP addresses, port number, buffer pointers and so on, if multiple TCP connections have to be achieved. It should be noticed that a TCP/IP stack could be implemented even in a full-hardware fashion. However, this kind of implementation is still more expensive than the software one, which is also more flexible if adequate computational speed is provided. On the other hand, UDP is more efficient (η) because its header field is more compact if compared to TCP; in addition, since UDP is a connectionless protocol, it allows a lighter and faster interface to be used and could be preferable in a low-cost or real-time environment.

For these reasons, in the following sections we refer to UDP as the selected Transport-layer protocol.

4. The proposed Internet-enabled sensor

Since IP fragments have no room to be stored, a solution could be to ignore fragmentation altogether, on the assumption that it can be avoided by forcing all stations to send packets smaller than the MTU. This works fine on a local network, where MTU is known, but it can fail if the Local Area Network (LAN) is connected to Internet. A solution for messages coming from Internet could be to force the IP router of the LAN to eliminate the option field, while is not useful for sensors, and to reassemble fragments. If this work is too heavy for the local IP router, a simple and efficient solution may be to impose the incoming IP packets to respect the 68 bytes limit. As the IP option field is variable from 0 to 40 bytes and no assumption can be made "a priori" about this length, IP data-field must be limited to 8 bytes.

Since the header of an UDP packet is 8-byte long, no space for user data d is available and some tricks have to be used. As depicted in Fig. 6, inside UDP header there are two 16-bit long fields, source port and destination port, that are commonly used to multiplex data between destination and source application. Generally speaking, a destination and source ports must be specified, because every high-level application is identified by its port value, that is a sort of postal address. Many ports can be managed simultaneously realizing a virtual circuit between the two sides of the connection. However, the meaning of these fields still depends on the high-level application, thus data to be sent to a simple sensor can be totally coded into them. In this way, UDP data-field is empty ($d = 0$) and incoming IP fragmentation can be ignored by the sensor.

An easy proprietary protocol, at application level, has been developed using the above mentioned multiplexing feature of UDP. There are the so-called "well-known ports" [16,17] that belong to standard servers. Well-known port numbers range between 1 and 1023 and should be avoided by any non-standard application. The other group of ports (1024–65535) are the so-called "ephemeral ports" and developers are free to use them.

The proposed method assumes destination port field to contain the command and its target address A_{12}–A_0, while

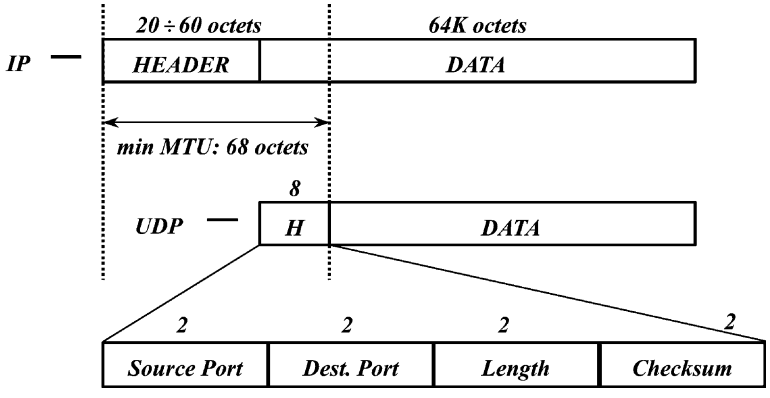

Fig. 6. IP packet.

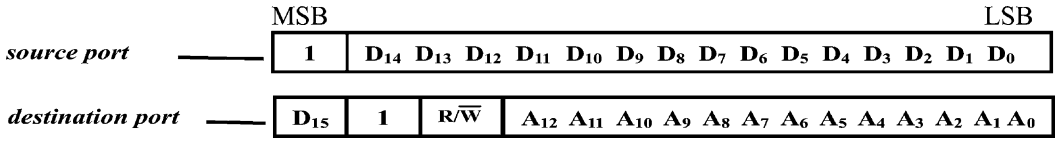

Fig. 7. Format of source and destination ports.

source port holds data value D_{15}–D_0 (optional) if a write operation is performed, as Fig. 7 shows.

To guarantee that none of the well-known ports are, the most significant bit (MSB) of the source port and the 15th bit of the destination port must be set to '1'. The lowest portion of the destination port field contains address bits (called A_{12}–A_0), the 14th bit allows to distinguish between a read or a write operation while the MSB specifies the most significant data bit (D_{15}). The source port contains the remaining of the data bits (D_{14}–D_0). This leads to a very simple and efficient 16-bit read–write protocol with an addressing space of 8192 words. Further acknowledgment function can be provided by echoing the message with swapped destination and source ports.

If a read operation is requested, data can be also sent by sensor in the UDP data-field. In fact, as previously stated, the IP fragmentation is a problem only for incoming data.

5. Experimental setup and results

The Internet-enabled sensor has been realized implementing ARP, IP, ICMP while UDP have been implemented on the top of the Ethernet hardware (see Table 2). Since writing complex applications in a low-level language is quite difficult, the higher-level protocols have been developed in C.

To better analyze protocol cost, in terms of memory space and performance, two experiments have been realized: an Internet-enabled sensor at network level (IP, ARP, ICMP) and an Internet-enabled sensor at transport level (IP, ARP, ICMP, UDP).

First of all, an implementation to manage a ICMP echo request, with a total of 1500 bytes of code, 18 bytes of data memory and 128 bytes of stack, has been realized. The experimental setup, shown in Fig. 8, is a star-type LAN including a network-enabled sensor (as the one indicated in Fig. 4), two PCs and a Linux Net Analyzer equipped with two "sniffer" software: Ethereal and Tcpdump. This architecture allows to grab each bit which is sent or received on the physical link, organizing and decoding both Ethernet frames and high-level protocols. In the following the sensing element has been emulated by a generic analog signal coming from a synthesizer.

The first experiment has been conducted measuring the average response time for a ICMP echo request (i.e. PING). The sensor takes about 5.3 ms to reply a 0-byte PING (a 28-byte long IP packet), while 8.1 ms suffice for a PING size of 32 bytes (a 60-byte long IP packet). It should be remarked that reply delay to a 32 bytes PING between two PCs is about 0.341 ms for a Pentium III @ 800 MHz. All these time intervals have been computed as the difference between time references associated to ICMP reply and request packets as measured by Net Analyzer. The measured performances are worse if compared to the selected field-bus because of two reasons: (i) different functionalities, as field-bus implementations are limited to the Physical and Data Link layers, and (ii) the lack of external glue logic between microcontroller and CS8900A.

As the second experiment, a simple application based on UDP, as described in the previous section, has been realized. The whole program spans about 2300 bytes and needs 52 bytes of data memory and 128 bytes of stack. No buffers have been provided because, according to the proposed protocol, IP fragmentation can not occur and messages are managed directly when they are received. Spurious fragmented or not UDP messages are automatically discarded. To verify the bidirectionality of the protocol, two LEDs (LED 1 and LED 2) have been connected to the microcontroller of the network-enabled sensor shown in Fig. 8. These devices are mapped inside the addressable memory space (8192 words) allowed by the protocol, according to Table 4.

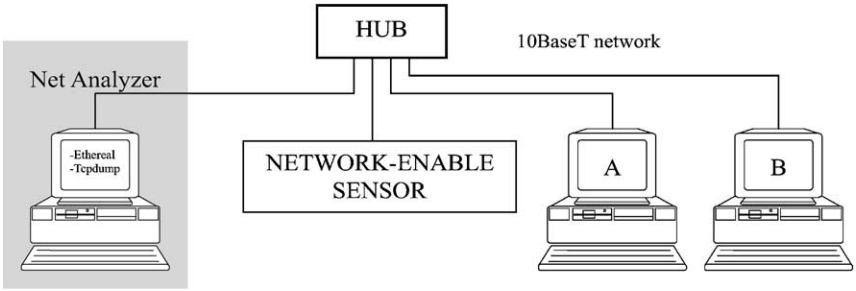

Fig. 8. The network test bench.

UDP request for new data (read from Address 0x0100 sent by PC)

```
0000   00 C0 26 F1 77 D1 00 C0 DF 08 59 E0 08 00 45 00   ..&.w.....Y...E.
0010   00 1C A5 03 00 00 80 11 E8 06 C0 A7 16 29 C0 A7   .............)..
0020   16 4F 80 03 61 00 00 08 71 13 20 20 20 20 20 20   .O..a...q.
0030   20 20 20 20 20 20 20 20 20 20 20 20
```

UDP reply with data (sent by sensor)

```
0000   00 C0 DF 08 59 E0 00 C0 26 F1 77 D1 08 00 45 00   ....Y...&.w...E.
0010   00 1E 20 7C 00 00 3C 11 B0 8C C0 A7 16 4F C0 A7   .. |..<......O..
0020   16 29 61 00 80 03 00 0A 3D CC 33 43 43 43 43 43   .)a.....=.3CCCCC
0030   43 43 43 43 43 43 43 43 43 43 43 43               CCCCCCCCCCCC
```

Fig. 9. Ethernet packets without preamble and CRC. Bold bytes are the UDP packet. Highlighted position are bytes of interest. Grayed-out bytes are the automatic Ethernet packet fill to 576 bit.

Table 4
Memory map of experimental sensor

Address	Data	Operation	Action
0x0100	X	Read	Read analog input value
0x1000	Bit (D_0)	Write	Turn on/off LED 1
0x1000	Bit (D_1)	Write	Turn on/off LED 2
0x1000	X	Read	Read LED status

To verify the compatibility of the implemented protocols with standard equipments, a client application has been implemented using LabView that is installed both on PC A and PC B shown in Fig. 8. This application, formally a ‘‘virtual instrument’’, asks the proposed Internet-enabled sensor for new readouts every 200 ms, then the collected data are displayed in a graph window: Signal with slow dynamics can be correctly reconstructed, allowing non-critical real-time applications in a traffic-free network. LEDs control can be achieved writing the proper address.

Byte transfer during this operation has been recorded by the Net Analyzer (using Ethereal). Fig. 9 shows the Ethernet packets without the preamble and the CRC field. UDP packet is reported in bold. Source port and destination port values have been highlighted both in the request and the reply packet, where the sent data, bytes ‘‘33’’ and ‘‘43’’ (ASCII character ‘3’ and ‘C’), corresponding to an analog input value of 5 V (3Ch/100 h) = 1.17 V, are also visible. Destination port values can be easily verified to be compliant with the proposed protocol (Fig. 7): a read operation at address 0x0100 is encoded into word 0x6100. Source port, whose value is 0x8003 (i.e. data = 0x0003), is ignored during a read operation. Reply time to a UDP request for new data has been estimated by the Net Analyzer running Ethereal, as an average over 100 consecutive identical transactions. A value of 5.7 ms, comparable with the 0-byte PING response, confirms simplicity of the proposed protocol, that is theoretically able to manage up to 200 ms/5.7 ms ≈ 35 instruments that simultaneously request data from the sensor.

6. Conclusions

In conclusion, the theoretical analysis and first experimental results show that implementing an Ethernet-capable sensor is comparable, as regards cost, data throughput and complexity, to a field-bus interface like Profibus-DP or CANbus2.0B. It is to say that Ethernet has gained popularity due to the fact that it is readily available, promoting its introduction even in the ‘‘plant’’ floor for control and device-layer applications. A further analysis has demonstrated the feasibility of an Ethernet-sensor with IP capability with costs similar to the above. Furthermore, to be simply managed by widespread software in the instrumentation area, like LabView, a proprietary smart and low-cost solution based on UDP has been proposed and realized. In effect, implementing high-level protocols leads to object abstraction, allowing an easy compatibility with PLCs and PCs. Experimental results have demonstrated the functionality of the proposed system.

References

[1] M. Bolic, V. Drndarevic, B. Samardzic, Distributed measurement and control system based on microcontrollers with automatic program generation, Sens. Actuators A 90 (2001) 215–221.
[2] G. Smith, M. Bowen, Consideration for the utilization of smart sensors, Sens. Actuators A 46–47 (1995) 521–524.
[3] Institute of Electrical and Electronics Engineers, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, IEEE Standard 802-1990, 1990.
[4] Institute of Electrical and Electronics Engineers, IEEE802.3, Part 3: 2000 Standard, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and Physical Layer Specifications, 2000.
[5] International Standard Organization, Road Vehicles—Interchange of Digital Information—Controller Area Network for High Speed Communication Draft Amendment, ISO 11898-1993/DAM 1, 1994.
[6] European Committee for Electrotechnical Standardization, General Purpose Field Communication System, Vol. 2, Profibus, EN50170, 1996.
[7] G. Cena, L. Durante, A. Valenzano, Standard field-bus networks for industrial application, Comput. Standards Interfaces 17 (1995) 155–167.

[8] Institute of Electrical and Electronics Engineers, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators: Network-Capable Application Processor (NCAP) Information Model, IEEE Standard 1451.1-1999, 1999.
[9] Institute of Electrical and Electronics Engineers, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators: Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Standard 1451.2-1997, 1997.
[10] J. Bryzek, Introduction to P1451: the emerging hardware-independent communication standard for smart transducers, Sens. Actuators A 62 (1997) 711–723.
[11] S. Vitturi, On the use of Ethernet at low level of factory communication systems, Comput. Standards Interfaces 23 (2001) 267–277.
[12] C. Szabo, An Ethernet compatible protocol to support real-time traffic and multimedia applications, Comput. Netw. ISDN Syst. 29 (1997) 335–342.
[13] C.D. Maciel, C.M. Ritter, TCP/IP networking in process control plants, Comput. Ind. Eng. 35 (1998) 611–614.
[14] J. Postel (Ed.), Internet Protocol RFC 791, Network Working Group, 1981.
[15] J. Postel (Ed.), User Datagram Protocol RFC 768, Network Working Group, 1980.
[16] J. Postel (Ed.), Transmission Control Protocol RFC 793, Network Working Group, 1981.
[17] I.D. Agranat, Engineering Web-technologies for embedded applications, IEEE Internet Comput. 2 (1998) 40–45.
[18] Technical Committee TC97, International Standard Reference Model of Open-System Interconnection (OSI), ISO 7498, 1984.
[19] G. Cena, C. Demartini, A. Valenzano, in: Proceedings of the IEEE International Workshop on Factory Communication Systems and on the Performances of Two Popular Field-Buses, 1–3 October 1997, pp. 177–186.
[20] D. Plummer (Ed.), Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48 bit Ethernet Address for Transmission on Ethernet Hardware, RFC 826, Network Working Group, 1982.
[21] J. Postel (Ed.), Internet Control Message Protocol, RFC 792, Network Working Group, 1981.
[22] J. Park, Y. Yoon, An extended TCP/IP protocol for real-time local area networks, Control Eng. Practice 6 (1998) 111–118.

Biographies

Alessandra Flammini was born in Brescia, Italy, in 1960. She graduated with honors in physics at the University of Rome, Italy, in 1985. From 1985 to 1995 she worked on industrial research and development on digital drive control. Since 1995 she has been a researcher at the Department of Electronics for automation of the University of Brescia. Her main field activity is the design of digital electronic circuits (FPGA, DSP, processors) for measurement instrumentation.

Paolo Ferrari was born in Brescia, Italy, in 1974. He graduated with honors in electronic engineering at the University of Brescia, Italy, in 1999. He is a PhD student in electronic instrumentation at the University of Brescia, since 1999. His main activity is the signal conditioning and processing for embedded measurement instrumentation.

Emiliano Sisinni was born in Lauria, Potenza, Italy, in 1975. He graduated in electronic engineering in 2000 at the University of Brescia, Italy. In the same year he started his PhD course in electronic instrumentation. His main field activity is the design of digital instruments.

Daniele Marioli was born in Brescia, Italy, in 1946. He graduated electrical engineering in 1969 at the University of Pavia, Italy. From 1984 to 1989, he was an associate professor in applied electronics, and since 1989 he has been a full professor of applied electronics at Brescia University. Since 1993, he has been the director of the Department of Electronics for automation of the faculty of engineering of the University of Brescia. His main field activity is the design and experimentation of analog electronic circuits for the processing of electrical signals from transducers, with particular regard to S/N ratio optimization.

Andrea Taroni was born in Cotignola, Ravenna, Italy, in 1942. He received the degree in physics from the University of Bologna, Italy, in 1966. He was an associate professor at University of Modena from 1971 to 1986. Since 1986 he has been a full professor in electrical measurements at the University of Brescia. Since 1993 he has been the dean of the faculty of engineering of the University of Brescia. He has done extensive research in the field of physical quantities sensors and electronic instrumentation, both developing original devices and practical applications.