POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Client-Server Architecture for Distributed Measurement Systems

Original
A Client-Server Architecture for Distributed Measurement Systems / Bertocco, M.; Ferraris, Franco; Offelli, C.; Parvis, Marco. - In: IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT. - ISSN 0018-9456. - STAMPA. - 47:5(1998), pp. 1143-1148. [10.1109/19.746572]

Availability:
This version is available at: 11583/1400295 since:

Publisher:
IEEE / Institute of Electrical and Electronics Engineers Incorporated:445 Hoes Lane:Piscataway, NJ 08854:

Published
DOI:10.1109/19.746572

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

25 April 2024

A Client–Server Architecture for Distributed Measurement Systems

Matteo Bertocco, Franco Ferraris, Carlo Offelli, and Marco Parvis, *Member, IEEE*

***Abstract*—This paper describes a client–server architecture for the remote control of instrumentation over the Internet network. The proposed solution allows multiuser, multi-instrument sessions by means of a queueing and instrument locking capability. Client applications can be easily developed by using conventional high-level programming languages or well-assessed virtual instrumentation frameworks. Performance tests are reported; they show the low overhead due to network operation with respect to the direct control of instrumentation.**

***Index Terms*—Client–server systems, education, intelligent sensors, internetworking, measurement system data handling, remote sensing.**

I. Introduction

THE increasing use of programmable instruments has brought about changes in the operating procedures which are commonly adopted for the solution of practical measurement problems. Before the availability of programmable instrumentation, the user was required to set up the test system by connecting the device under analysis to proper instruments; then one had to choose the best excitation signals and configure the instruments by interacting with their front panels. The results were obtained by reading the data-displays of the instruments and, sometimes, after some further simple calculations made by hand. Nowadays, this operating scheme is outdated and often cannot be applied. Many intelligent instruments are not provided with a front panel; moreover, modern tests are performed first by writing a suitable program for a host computer that controls the instruments, and then by running that program with the device under test (DUT) connected to the test station.

In addition, in the last few years a surprisingly rapid growth of fast and reliable communication networks has allowed an easy interchange of information and commands between computers both connected to local networks and connected to far sites of wide area networks (WAN) such as the Internet. Thus, network services and programmable instrumentation now permit the development of measurement laboratories distributed on a wide geographical area and simultaneously available to several users variously located in the territory. Such a kind of distributed measurement laboratory can satisfy two main requirements.

Manuscript received May 21, 1998; revised November 9, 1998.

M. Bertocco and C. Offelli are with the Department of Electronics and Computer Science, University of Padova, Padova 33131 Italy.

F. Ferraris and M. Parvis are with the Dipartimento di Elettronica, Politecnico di Torino, Torino 10129 Italy.

Publisher Item Identifier S 0018-9456(98)09754-X.

The first one consists of an easy and efficient use of expensive or complex instrumentation systems. Typical examples are the anechoic chambers used for electromagnetic compatibility (EMC) experiments or the complex instruments that are used to test the integrated circuit wafers during the design of new devices. In these cases, the remote access to measurement systems allows the interaction between designers or application engineers and instrumentation without the need to physically move people or bulk instrumentation, thus increasing the speed of the development process and also improving the interaction between many experts in the field. One should also note that the rental of costly measurement services is possible in a rather simple way, because only the target device needs to be moved while a better exploitation of valuable instrumentation can be achieved.

A second advantage coming from the availability of a remote instrumentation is the possibility of arranging distributed measurement systems which are both controlled from a single position and able to perform complex measurements strictly related to the site where instrumentation is located. An important example is the control of environmental parameters where scientists or government agents have the need to perform several measurements in many locations distributed on the earth. In this last example more users possibly would like to perform their own measurements by sharing the instruments with the others to gather different data or compare different post-processing analysis techniques.

This paper presents a technique for the remote control of distant instrumentation in a multiple-user, multiple-laboratory environment. Such a solution has been developed under a common project that involves the University of Padova and the Polytechnic of Torino, which are located in the eastern and western parts of Italy.

The proposed technique is based on a client–server architecture which is described in Section II. With this technique, many users can simultaneously share remote instruments by using suitable client procedures that interact with a server program running on a computer physically connected to the instruments. This paper also shows that client programs can be easily built by using a set of library functions provided with the developed environment.

Section III reports meaningful examples of end-user applications. The examples show that measurement problems can be solved in a remote-controlled environment with a moderate overhead due to network operation.

0018–9456/98$10.00 © 1998 IEEE

II. Client-Server Architecture

A. The VXI-11 Solution

Techniques for the remote access to instrumentation have already been proposed in the literature. One interesting proposal comes from the VXI Consortium, which has designed a powerful extension to the IEEE-488 bus [1] referred to as VXI-11 specification. Such an extension allows remote hosts and an instrument controller to communicate over a network to perform measurements as if the instruments were connected directly to the clients.

The VXI-11 specification relies on the TCP/IP communication protocol and therefore is suitable for use on both local area networks and on the Internet, thus allowing addressing long-distance connections. Moreover, it suggests the use of a technique referred to as remote procedure call (RPC). According to this technique, the server computer, which is connected physically to the instruments, makes available a set of remotely callable procedures that perform all standard IEEE-488 activities (addressing, read, write, status polling, etc.). The client or "remote" computer asks for a procedure to be executed and receives the results.

Software solutions exist that can be used to embed the RPC approach in already developed applications. This approach therefore allows for the distribution of the measurement tasks among several computers, but it has a major drawback when employed in a multiuser environment. In fact, the IEEE-488 was designed for a scenario where a single user has complete control over all the available instrumentation, while a multiuser approach implies the simultaneous access of different computer tasks to the same network-shared instruments.

To solve this problem, the VXI-11 specifies a locking mechanism. Each resource (i.e., each instrument) can be "locked" by a user so that any other request of access to the same resource is denied until the locking expires. This approach actually prevents abnormal operations, but has a limited flexibility. In fact, a user that cannot gain access to a measurement subsystem due to a pending lock operation is not able to gain any information concerning the duration of the lock or about the number of other clients potentially able to access the same resource. Furthermore, deadlock can arise, since two users can simultaneously lock two instruments and then they can mutually require the instrument the other user already locked to complete execution.

B. A Multiuser Multi-Instrument Proposal

To overcome the limitations of the RPC mechanism, an alternative technique has been developed. It consists of a communication protocol and a set of procedures for the interconnection in a wide area network of a measurement system controlled by a "server" computer, with more concurrently operating remote "client" computers possibly running different operating systems.

Since network connections could be established between different computing environments or operating systems, a message-based protocol has been adopted, based on the well-accepted TCP/IP suite. This latter choice overcomes the problems arising from the physical location of the RPC server. Moreover, by employing specific TCP "ports" for the message interchange, the limitations due to firewall hosts can be easily solved.

The second quoted drawback of the VXI-11 proposal due to possible multiuser interaction is addressed by establishing and handling a queue of client requests, and by allowing the clients to receive fast responses to requests for information forwarded to the server, such as the queue status or the server actual load.

The block diagram of the proposed solution is shown in Fig. 1 where one client and one server are reported for the sake of clarity. Both client and server computers run programs which are logically split into two layers. One layer in both client and server sides deals with the network interconnection while the other layer deals with the instrument management and user interface.

- The client application (CA) contains the procedures that are related to the user interface as well as to the measurement request and processing. This is the program part that would be required if the measurement session were carried out by means of instruments directly controlled by the client computer.
- The measurement client (MC) contains the procedures that are related to the network management: server contact and log-in, data flow control, and recovery from network failures. The communication between the MC and the CA can be designed according to the different operating system opportunities.
- The measurement server (MS) contains the network-related procedures on the server side and the queuing management. It accepts the connection requests from the clients and processes their messages according to the kind of request: the messages regarding the queue and server status are acknowledged immediately, while all the requests regarding a measurement operation are queued and dispatched to the instrument manager as soon as possible.
- The instrument manager (IM) contains the procedures related to the instrument management. It can be designed to operate by means of IEEE-488 boards as well as with serial ports and can be used to manage acquisition boards and other daughter boards, which can be employed by the remote clients for control and configuration purposes.

The development of the MC and MS modules, which have to deal with network issues, requires a considerable skill and the knowledge of several network-related topics. The choice of splitting both client and server into two layers that are operated by different software modules allows the MC and MS to be developed independently from the user and instrument interfaces.

C. Interconnection Protocol

The interconnection protocol relies on messages composed of a header and a body. All the messages have to pass through the MC and MS and the headers are used to efficiently identify the modules that have to process the message body. The header need not be encoded and is composed of a sequence of ASCII

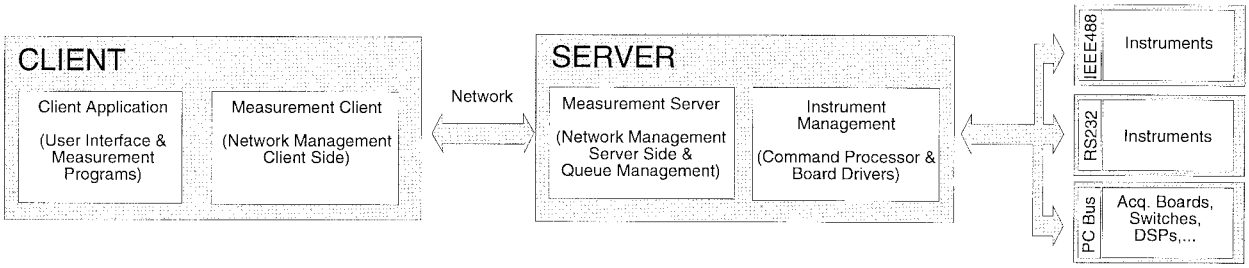


Fig. 1. Block diagram of the client–server architecture for the remote control of instrumentation.

characters, while the message body, which can be composed of a long stream of data, can be either a binary stream or a character sequence. This allows both a simple program development and an efficient use of the network bandwidth.

The messages directed to the IM have been classified into two categories which have been referred to as "instrument" and "experiment." The instrument messages refer to operations that have to be performed on single instruments and are the natural extension of the IEEE-488 messages as in the VXI-11 approach. The experiment messages refer to more complex environments where a more complex measurement involving several instruments can be set up with a single encoded message. Each experiment requires a procedure in the IM capable of decoding the message, setting up the instruments, and encoding back the response. The experiment procedures have to be designed by the end user of the experiment and need to be installed in the IM by the server manager. This extension, though not as simple and flexible as the simple instrument driver, has been designed to allow both a substantial reduction of the network traffic and efficient instrument use where complex measurement procedures are required.

III. Experimental Results

Experiments have been performed both to investigate the degree of difficulty and skill required to port existing applications in the remote environment and to test the environment performance in term of measurement throughput.

The porting has been carried out on programs currently in use for training and research in the authors' laboratories [2]. In all tests the computer that hosted the server was running Windows 95 or Windows NT, while some clients were tested in the UNIX environment also. The IM was written in VisualBasic, and both Hewlett-Packard and National Instruments IEEE-488 boards were used on different computers. The IM was designed to provide basic ⟨send⟩ to instrument and ⟨receive⟩ from instrument capabilities and to manage an "experiment" involving the simultaneous use of different instruments.

In all cases the modifications required to existing programs were rather small. Programs that were originally developed in VisualBasic or VisualC required only the addition of a very small number of statements necessary for establishing and closing the network connections, together with the substitution of the calls to the interface-related functions to corresponding network functions.

Even simpler modifications were required when LabView programs were considered. In this case, most of network overhead is handled by some library functions that should be installed overwriting the vendor-supplied ones. The modifications which were required to port the existing applications were then limited to the addition of some new global variables which are required to handle the necessary network information.

Specific tests were written to understand the performances of network-controlled instrumentation systems. The tests were designed to investigate the overhead due to network connection and the effect of a concurrent access. A LabView program was written that remotely interacts with a waveform generator (Hewlett-Packard HP8904A) and a sampling oscilloscope (Hewlett-Packard HP54501), the former being directly connected to an input channel of the latter.

The test consisted of a simple remote measurement session, which was repeated 500 times. Each session was composed of four steps: 1) request of the exclusive use of instruments to the server (lock); 2) setup of the waveform generator and the oscilloscope to generate a sinewave and to measure its peak-to-peak value; 3) reading of the measurement results from the server; and 4) release of the instruments (unlock). Three time intervals were determined by using the real-time clock of the local computer during each session: 1) the time required for obtaining the exclusive use of the instruments (T_{lock}); 2) the time required to remotely set up the instruments and obtain the measurement results from server (T_{meas}); and 3) the time elapsed from the request of connection to the server up to the instant in which the connection is eventually closed (T_{conn}). The tests were performed with both client and server connected to the same local area network which is used in the facility.

The first run was performed by activating one client only; the time required to obtain an exclusive use of the instruments from server (T_{lock}) was rather low, spanning in the range of 0.15–0.35 s. The variability is mainly due to the obvious lack of synchronization between client and server activities, as well as to network-dependent parameters such as the data traffic.

The measurement time (T_{meas}) exhibited a lower variability; in fact, its standard deviation was approximately equal to 0.06 s, while the mean value was about 1.5 s. One should note that T_{meas} includes the time required by the instrument to perform the measurements and some overhead due to the transmission over the network of the commands and of the measurement results.

The total connection time (T_{conn}) is equal to the summation of T_{lock}, T_{meas} and of the time required for the client–server

pair to release the exclusive use of instruments granted to the client as well as to close the connection (T_{unlock}). By subtracting from T_{conn} the sum $T_{\text{meas}} + T_{\text{lock}}$, it is easy to see that the average values of T_{unlock} and T_{lock} are about 0.2 s.

A second test was carried out by activating two clients that simultaneously interacted with the server. As the second client activated, the T_{lock} of the first client increased, from the previous recorded values of 0.2 s, to about 2 s. The increase of about 1.8 s is approximately equal to T_{conn} of the other client. For comparison, the measurement procedure described above has been performed in a "local" environment, i.e., by directly using the CA to control the instruments by means of a computer equipped with a IEEE-488 interface. The time intervals T_{meas} which were required for accomplishing single measurements have a mean value of about 0.8 s, which is lower than the mean T_{meas} for a remote client (about 2 s). The difference is due to the intrinsic overhead imposed by the remote-controlled environment driven in "instrument" mode. In this operational mode the client has to await an "operation completed" message before posting a new command so that the synchronization between client and server is assured. This penalty is often negligible when using instruments that take a long time to complete the measurement procedure, but can become significant with complex devices requiring a lengthy setup. In this case the "experiment" approach can be employed that nearly eliminates the overhead by encoding all the requests in a single network packet.

Other tests were performed by increasing the number of clients up to ten. The locking time showed a linear increase with the number of connected clients. This confirms that the MS does not add significant delay due to the queue management.

Another series of tests was performed to investigate the performance of the system over a wide area network. The Italian Interuniversity 2 Mb/s network was used for the connection between a client located in the Torino University and a server located in the Padova University. One should note that the connection required eight network hops involving different routers located along the path between the server and the client computers.

A couple of simple "send-packet" and "echo-packet" programs were specifically developed to track the actual network speed during the measurement tests The programs were designed to measure the trip-time of transmission control protocol (TCP) and user datagram protocol (UDP) network packets having the same size of those required for the remote measurements, in such a way that network overload can be evaluated.

Two measurement experiments were arranged by writing client programs within the LabView environment. The first one consisted of a simple measurement of a resistance value performed with a digital multimeter. The second experiment consisted of a more complex test where a function generator and a sampling oscilloscope were controlled to acquire a 1024-sample waveform. Both tests were performed repeatedly while collecting the data during an experiment that lasted about 24 h.

The resistance experiment involved only three network transactions to carry out the measurement (data request and result report) plus the lock/unlock procedure, while the oscilloscope experiment was composed of 17 network transactions (including lock/unlock procedures) that are required to set up the instruments and to receive the data from the oscilloscope.

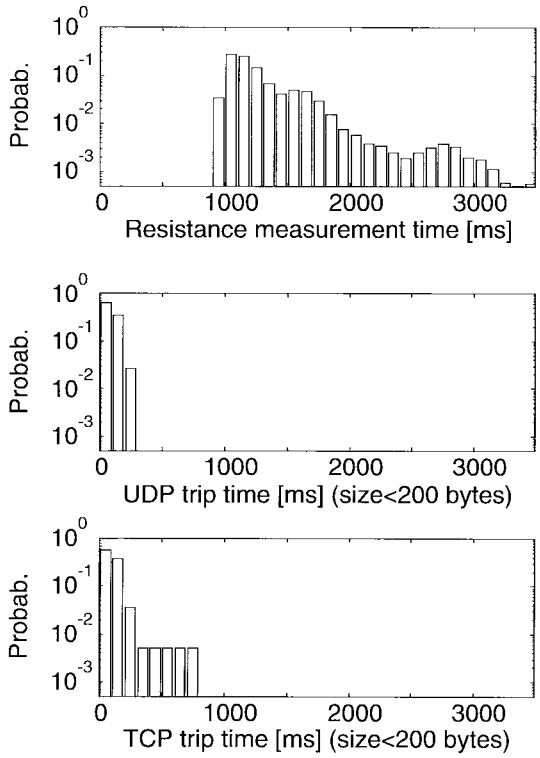


Fig. 2. Probability density function associated with the measurement time required for the resistance test and corresponding transmission round trip times for the case of TCP and UPD protocols.

Fig. 2 shows the results for the resistance measurement test while Fig. 3 refers to the oscilloscope test. In both cases the three plots summarize the total measurement time (including the lock and unlock time) and the TCP and UDP trip times. The average measurement time for the resistance test was about 1.3 s while the mean TCP trip time for packets with dimension below 200 bytes was about 120 ms. The measurement time, therefore, agrees with the sum of the time the multimeter takes to perform the measurement (about 0.5 s) plus the total average network time (five transactions, each of 120 ms), showing that system overhead is limited to about 0.04 s per network transaction.

A similar situation is shown in the oscilloscope experiment where the average measurement time is about 4.8 s and the trip time for TCP packets with dimension of up to 3 kB is about 0.23 s. The oscilloscope requires 0.21 s to perform the measurement with a network time for each of the 17 transactions of about 0.23 ms. The system overhead is therefore again limited to 0.04 ms per network transaction.

It is interesting to note that the use of the UDP protocol could reduce the network overhead of a ratio of about 10%. In fact, the mean trip time for small packets was about 90 ms instead of 110 ms, and 210 ms instead of 230 ms for large packets. This overhead reduction, however, is obtained at the expense of a lower transmission reliability. In fact, it has been noticed that the TCP protocol timed out during

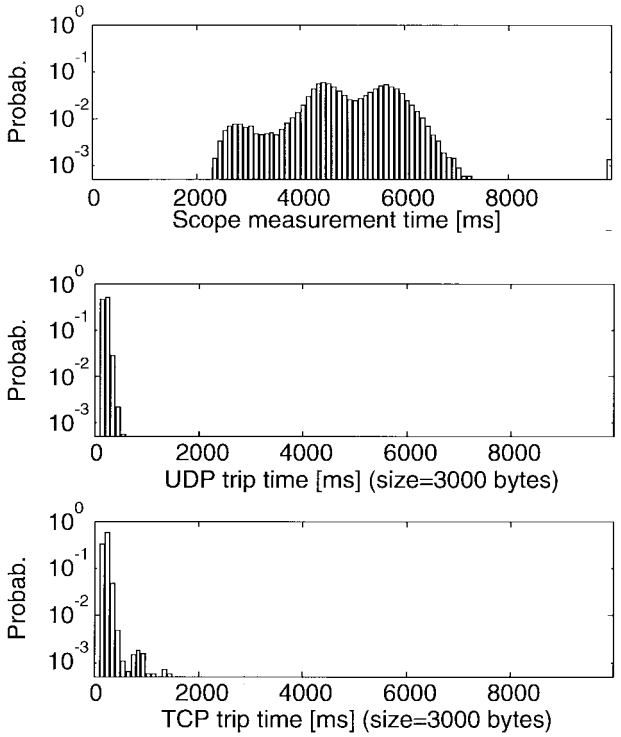


Fig. 3. Probability density function associated with the measurement time required for the oscilloscope test and corresponding round trip times for the case of TCP and UPD protocols.

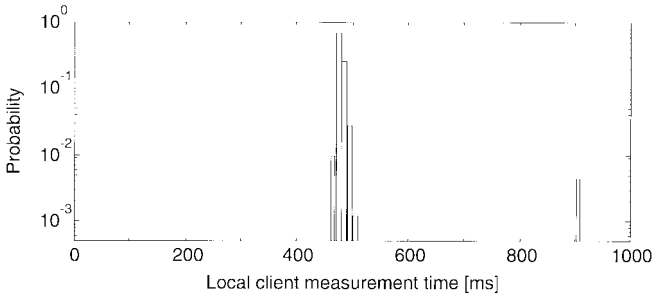


Fig. 4. Probability density function associated with the measurement time required for the case of the evaluation of a resistance value obtained by directly controlling the instruments.

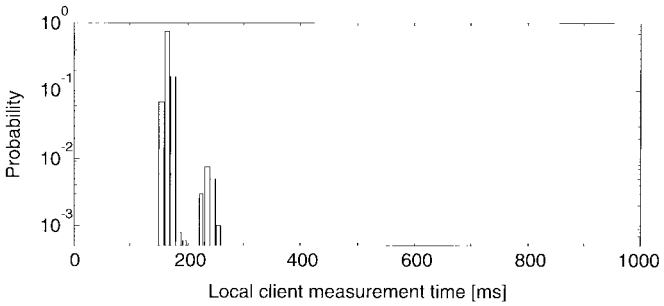


Fig. 5. Probability density function associated with the measurement time required for the case of the acquisition of a sampling oscilloscope waveform, obtained by directly controlling the instruments.

the tests with a probability of less than 0.02% while the UDP did not deliver the packets in 2% of the cases. This great difference in the transport reliability, along with the capability of the TCP protocol of signaling any network failure, suggests that the UDP approach should be used only for local area networks where it can provide a somewhat higher transfer rate.

For comparison, two programs have been specifically designed. They are identical to the above two clients used for the measurement of a resistance value and for the acquisition of a waveform, but instead of interacting with a remote server, they directly control instrumentation and record the time required to perform a single measurement. The results have been used for the evaluation of the probability density function (PDF) associated to the measurement times, which are in turn reported in Figs. 4 and 5. Fig. 4 shows that a mean time of about 0.5 s is required for the measurement of a resistance value, while about 0.21 s are required for the acquisition of a waveform. These latter results agree with the ones reported in Figs. 2 and 3. In fact, by summing the measurement time obtained for the case of the direct control of instrumentation to the TCP round trip delay multiplied by the number of network transactions required to perform a far measurement, approximately the measurement time associated to the remote execution of a measurement is obtained.

IV. Conclusions

The remote instrumentation control is becoming popular since the networks have become reliable and worldwide, and almost every new instrument embeds programmable capabilities. Several proposals have been developed by different vendors to address the remote instrument control issues even though most solutions have been designed as simply "extensions of the cable" which is used to connect computer and instruments.

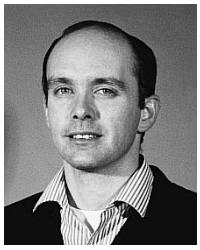
This paper presents a proposal that takes the multiuser problems into account. A queue mechanism has been added to the remote environment along with the possibility for each client to query the actual server load. The communication between server and clients can be obtained either at instrument level or by means of encoded requests in order to reduce the network-imposed overhead. Tests have been performed to estimate this overhead, and it has been found to be reasonably low: about 0.2 s are required for the initial instrument locking and an additional penalty of 0.04 s is experienced for each command with respect to the execution time in nonnetworked environments.

A set of precompiled experiments based on the proposed technique for the control of far instrumentation has been made available to the students of "Electronics and Measurement" courses held in Torino and Padova Universities [2].

References

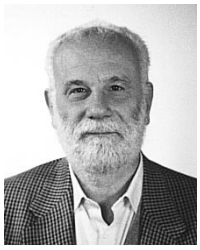
[1] "VMEbus extensions for instrumentation. TCP/IP instrument protocol and interface mapping specifications," *VXIbus Consortium*, Spec. VXI-11, Rev. 1.0, 1995.

[2] M. Bertocco, F. Ferraris, C. Offelli, and M. Parvis, "Training of programmable instrumentation: A student laboratory," in *Proc. XIV Imeko World Congress*, Tampere, Finland, June 1–6, 1997.



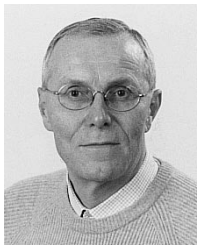
Matteo Bertocco was born in Padova, Italy, in 1962. He received the Laurea degree in electronics engineering from the University of Padova, where he subsequently received the Ph.D. degree in electronics engineering in 1991.

Since 1994, he has worked as a Researcher at the Department of Electronics and Informatics, University of Padova, becoming an Associate Professor of electronic instrumentation and measurement in 1998. His research interests are in digital signal processing, estimation, automated instrumentation, and electromagnetic compatibility.



Franco Ferraris was born in Italy in 1945. He received the Laurea in electric engineering from Politecnico di Torino, Torino, in 1969.

He is a Full Professor of electronic measurement in the Dipartimento di Elettronica of Politecnico di Torino. His main research fields are measurement system modeling, design and development of intelligent and remote measurement systems, and design and metrological characterization of physical quantities sensors.



Carlo Offelli received the Laurea degree in electronic engineering from Padova University, Padova, Italy in 1970.

He is a Full Professor of electronic instrumentation in the Department of Electronics and Computer Science of Padova University, Italy. He has been involved for several years in research work on speech analysis and synthesis. His present research interests comprise application of digital signal processing to measurement problems, real-time analysis, and electromagnetic compatibility.

Marco Parvis (M'98) was born in Italy in 1958. He received the degree in electrical engineering in 1982 from Politecnico di Torino, Torina, Italy, and the Ph.D. degree in metrology in 1987.

He then worked as a Research Assistant of electronic measurements at the Politecnico di Torino and is now a Professor of instrumentation and measurements in the university's Dipartimento di Elettronica. His main fields of interests are intelligent instrumentation, application of signal processing to measurements, and biomedical and chemical measurements.