19

New Technologies For Measurement Systems Distributed On A Wide Area

Giovanni Bucci, Fabrizio Ciancetta and Edoardo Fiorucci
Università dell'Aquila (Dip. Ing. Elettrica e dell'Informazione)
ITALY

1. Introduction to Distributed Measurement Systems

In different technological and information fields there is the need for multipoint measurement systems, to monitor a process evolution and to take proper controlling actions; several applications require for a *Distributed Measurement System* (DMS) able to measure the same or different parameters at different points.
Traditional test and measurement applications have been based on the use of centralized control and data management; usual implementations involve a central controller that handles instruments and sensors, polls for any results and processes the acquired data.
In these systems, the essential determination of the system's behaviour resides just in the central controller. The primary advantage of centralized systems is their simplicity: because all data is concentrated in one place, these systems are easily managed and have no problems of data consistency or coherence.
However, in some cases the physical area over which carry out the measurements is so wide that the implementation of a single measurement system will be unacceptable; main reasons are cost, reliability, and the distance between the measurement points. The measurement system must be split in a number of measurement devices, able to carry out each single measurement, to process the acquired raw data and to transmit the information to the final user.
Today, with the advent of dedicated microcontrollers and digital signal processors, complex data processing and transmission can be performed in a fast and easy way. These new technologies permitted the implementation of DMSs that addresses these issues: instead of having one single powerful system, distributed systems employ multiple systems communicating to each other via a common network.
DMSs usually refer to systems comprising one or more controllers each with one-to-one connections to sensors, instruments and possible actuators. This approach is always based on a number of smart measuring systems, where the nodes determine the system behaviour; the control protocol must therefore support each node internally, managing the application details occurring at that node. In addition, nodes can communicate directly with each other or with groups of other nodes, without any restriction to one-to-one communication links. The control protocol must support the transmission of synchronization messages between nodes to produce the correct overall system behaviour. Synchronization includes not only

the timing of measurements, but the overall progress of the application from one sequence of events to another.
In a DMS the management of data must be more structured than in traditional ones: in centralized systems many data management tasks, such as identifying the source and time of a measurement, are based on the properties of point-to-point communication links; in distributed systems using multicast, other techniques must be used for linking the various pieces of information in the system.
This chapter discusses about the new technologies that have been proposed, in recent years, for applications involving measurement systems distributed on a wide area.

1.1 Basic of DMS

A DMS can be seen as a collection of autonomous measurement systems linked by a network and equipped with distributed system software (Coulouris et al., 1994). The system employs intelligent controllers to perform message editing, data collection, dialogue with remote computer clients, some security functions, and message packaging.
The DMS software enables the measurement systems to coordinate their activities and to share system resources. A well-developed distributed system software provides the illusion of a single and integrated environment, although it is actually implemented by multiple measurement systems positioned in different places. In other words, is the software that gives a distribution transparency to the systems. Given the independent and distributed nature of these systems, it is important to underline the importance of having distributed software to provide a common and transparent view of the systems (Tari & Bukhre, 2001).

1.2 Classification of DMSs

Most of the architectures of DMSs include local or remote measurement systems, linked by communication connections with a central unit that provides for the system management and data acquisition. A centrally controlled distributed system is viewed as a collection of peripheral small measurement system, which might be capable to completely execute the measurement task and process the obtained results. Each of them is subordinate to a higher level central unit in the overall system structure. The main interconnection topologies are: the star, hierarchical or ring structure.
As shown in Fig. 1, each remote measurement system in the star structure is connected to the central unit via a network front end (e.g. a modem).
Fig. 2(a) illustrates a DMS interconnected in a ring structure, consisting of autonomous measurement systems linked in a peer-to-peer fashion.
Fig. 2(b) depicts a DMS with hierarchical structure. One or more locations have their own pre-processing unit, each of which performs the data processing and allows the communication of the measurement system with the central unit; the pre-processing unit periodically sends the required summary data to the central unit.
The idea of distributed systems represents a structure ranging from separate, geographically dispersed applications cooperating with each other, to a single application formed by relatively independent, stand-alone components. Another classification for the DMSs refers to four types of structural configurations.

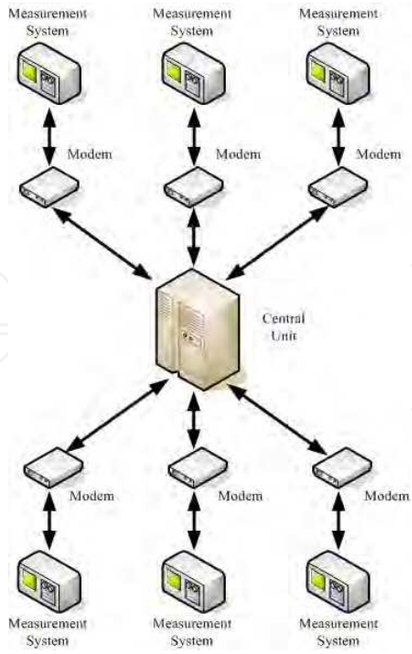
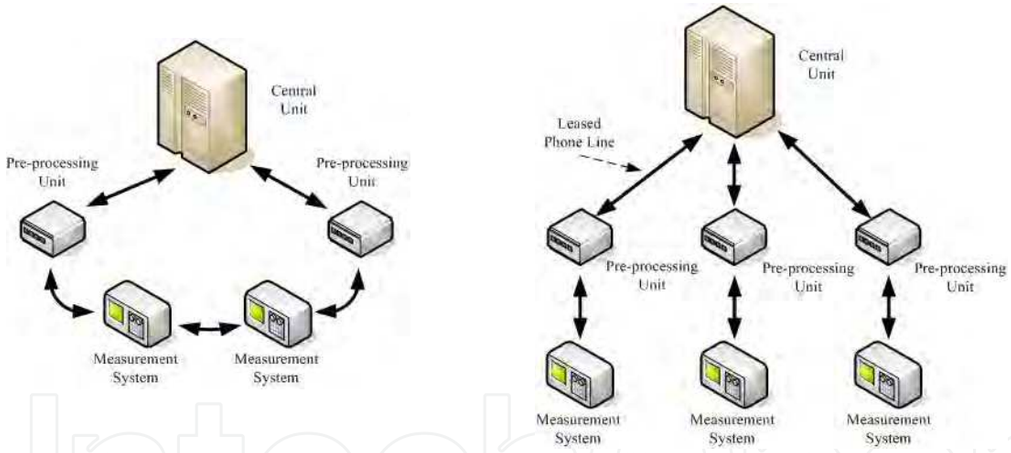


Fig. 1. Star structure



(a) Ring structure (b) Hierarchical structure

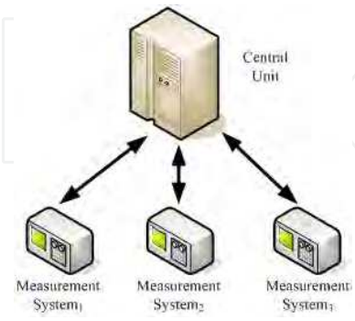
Fig. 2. Architectures of distributed measurement systems.

The first one, which is shown in Fig. 3(a), is a central unit connected to measurement systems with certain functions or applications.

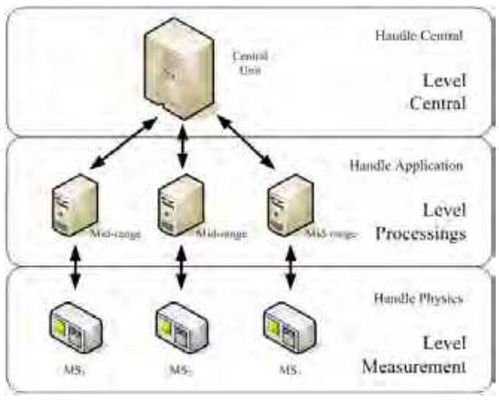
Another type is a measurement system connected to mid-range units, which, in turn, are connected to a central unit; as depicted in Fig. 3(b), all of them are organized in a hierarchical structure. The mid-range units deliver data to the central unit which handles the central level processing.

Fig. 3(c) illustrates a distributed system arranged as a connection of peers which have different details of interconnection and dependency; however, it is no evident a central control unit in this system.

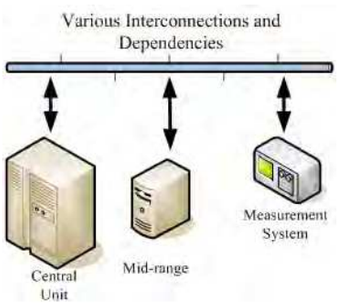
The last type, shown in Fig. 3(d), is a collection of peer hierarchies. The central unit, or each of the mid-range units, exhibits a hierarchical configuration and is connected to a collection of measurement systems.



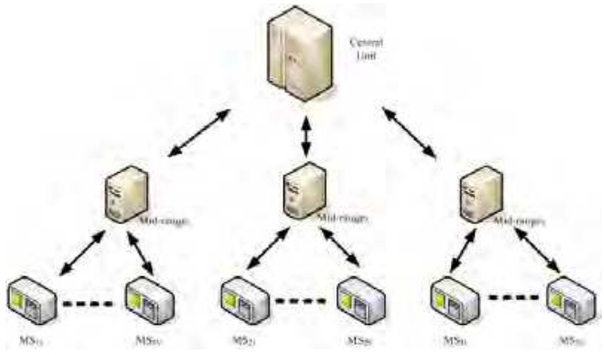
(a) Central unit and interconnected measurement systems



(b) Hierarchical structure and interconnection configuration



(c) Connection of peers structure and interconnection configuration



(d) Connection of hierarchical peers structure and inter-connection configuration

Fig. 3. Structural configurations

1.3 Main characteristics of a DMS

The trend of DMSs is motivated by the potential benefits that they could yield (Ozsu & Valduriez, 1991); (Tanenbaum, 1992). The first advantage of a DMS over a centralized system is cost effectiveness; other important characteristics are (Coulouris et al., 1994): (1) resource (provided by the processors) and data (measured parameters) sharing, (2) system expandability (hardware and software), (3) concurrency (simultaneous execution of multiple tasks), (4) scalability, (5) fault tolerance (carried out by hardware redundancy and software recovery), and (6) transparency (the users perceive the DMS as a whole system rather than as a collection of independent components). These characteristics are not automatic consequences of distribution; instead, they are acquired as a result of a careful design and implementation.

Beside these advantages, a DMS has some disadvantages: network reliance (problems on the network would disrupt activities in the system as a whole); complexities (a DMS

manipulates resources of computers with a wide range of heterogeneities); security (private resources would be exposed to a wider range of potential hackers, with unauthorized accesses). Decentralization contributes to the extensibility, fault-tolerance, and lawsuit proofing of the system, while the partial centralization makes the system more coherent than a purely decentralized system (Makarenko et al., 2004); (Berkes, 2003).
In a DMS all concurrent access must be synchronized to avoid problems such as lost update (two concurrent accesses update the same data, but one of the updates is lost), dirty read (one access updates the data read by another access, but the former fails and affects the latter), incorrect summary (a set of data is updated by an access while the set is being processed by another access), and unrepeatable read (an access reads data twice, but the data are changed by another access between the two reads) (Elmasri & Navathe, 1994).

2. Implementation techniques

2.1 Smart Web sensors

In this field, a new revolutionary technology is that of the microprocessor driven (smart) sensors (Yong et al., 2004), (Hamrita et al., 2005). The increased availability of communications and networking systems (both wired and wireless) is likely to bring about a crossing over of price and functionality between sensor networking technology and communications technology. This makes possible the installation of smart sensors on remote places, transmitting the measured information to the final user (a client).
A simplified block diagram of hardware implementation of a *remote smart sensor* is shown in Fig. 4. In the first section the physical quantity under measurement is transduced and then conditioned for the A/D converter input. The samples, acquired by the A/D converter, are pre-processed (i.e. averaging the measurements, changing the scale) by a local processor (microcontroller). A smart sensor can also include high hierarchical level applications, such as electronic data sheets, self-identification, self–testing, self-adaptation, smart calibration and compensation.

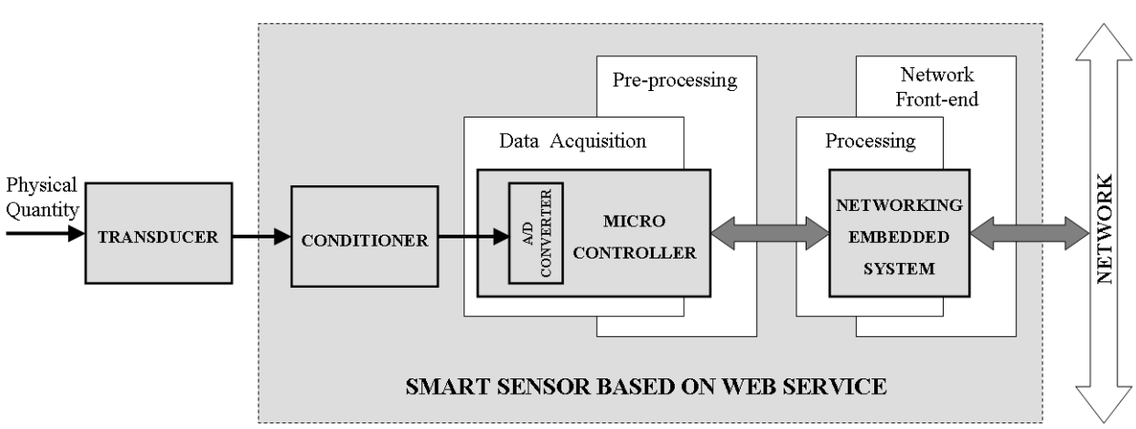


Fig. 4. Simplified block diagram of a Smart Sensor

The sensor needs to be able to communicate for remote monitoring and remote configuration; in Fig. 4 is the last block, the networking embedded system, that performs the final processing, storing and data transmission.

These new sensors are smaller, cheaper and more versatile than any other measurement device; their deployment in intelligent, interacting networks is opening up a new age in the DMS field.
The more revolutionary aspect is that related to the communication capability, which is addressed to link the sensors using a data transmission network. A lot of effort has been invested in order to overcome the obstacles associated with connecting and sharing these heterogeneous sensor resources. An interesting solution is to embody each sensor with tools capable to transfer the processed information through the network of sensors to a central computer.
Smart sensors can use different communication technologies. From network side, Internet is a widely adopted network where any user is uniquely identified with its IP (Internet Protocol) address. So, to implement a DMS, it is necessary to use a common and open communication protocol to exchange information and a methodology to auto-configure any smart sensor is linked to the network. This technology, known as *smart Web sensor*, makes various types of Web-resident sensors, instruments, image devices, and repositories of sensor data, discoverable, accessible, and controllable via the World Wide Web (Chu et al., 2006), (Lee et al., 2007).
DMSs based on smart Web sensors represent an interesting solution to many different measurement problems (Castaldo et al., 2004). These smart devices can transmit data to a remote processor for implementing remote monitoring of production processes, R&D experiments, environmental, security, or a wide-range of other, sensor-based monitoring tasks. In these applications, a computer client interacts with one or more smart sensors to download the measured parameters with a browser or an application capable to receive information from the Web server.
Interfacing transducers to all communication networks and supporting the wide variety of protocols is time-consuming and costly for manufacturers. To simplify this problem a standardized connection methods to interface smart transducers to the existing control networking technology has been proposed by the IEEE 1451 family of standards (IEEE Std 1451.1-4). The heart of the IEEE 1451.4 standard is the definition of the TEDS (Transducer Electronic Data Sheets), the information structure that contains the critical sensor information to enable plug-and-play operation.

2.2 Architecture of a DMS based on smart Web sensors

The architectures of a DMS based on smart Web sensors can be basically grouped in two categories.
The first, widely adopted, approach is based on a number of devices, linked with a centralized system, a central server keeping a list of users and shared resources (Fig. 5**Error! Reference source not found.**(a)). The primary advantage of centralized systems is their simplicity. Because all data is concentrated in one place, centralized systems are easily managed and have no problems of data consistency or coherence. During a search, every client sends a request to the central server that consults its lists providing results of IP user addresses. The file downloading happens between the two interested users from outside-centralized network. So, the server does not keep up any files. Each system is an independent server and must be selectively interrogated by the clients. The client needs to know the server position on the network (IP address) before starting the operations.

The second approach, the decentralized system shown in Fig. 5(b), is still based on a number of smart measuring devices, but presents the advantage to make easier the interrogation by the clients. This gives more extensibility to the network in which any node can join the network and instantly make new data available to the whole network. Another important feature of decentralized networks is that the failure or shutdown of any particular node does not influence the rest of the system. On the other hand, the intrinsic nature of this network gives two problems: the difficulties to manage the network because all the nodes have the same hierarchic level and the possibility to establish a packet loop that causes useless traffic.
To implement a network of smart Web sensors, two are the main problems to solve: i) how discover the smart Web sensors and ii) which interface is published to consume the services. From developer side, smart Web sensors present always a closed approach to interact with them, so Web services are adopted (Viegas et al., 2007), in order to give a standard approach in developing a Service Oriented Architecture. The user can search all the measurement related information available on the network using dedicated (special) services. Then he can ask to transfer the needed data form one or more of the measuring systems able to carry out the required measurement. There is no need to know any information related to the server (address list) before starting the search.
A hybrid schema is also possible combining centralized and decentralized systems (Fig. 5(c)).

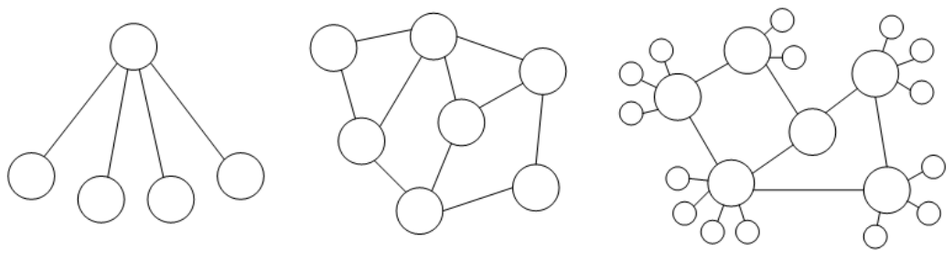


Fig. 5. Type of distributed measurement networks

2.3 Technologies for the DMSs

Ethernet is a cost-effective communication medium for DMSs; the growing dissemination and use of World Wide Web browsers and Java programming language also has a significant impact on sensor networking. DMS is frequently based on software packages that support Web-based data interfaces.
Despite the adoption of smart Web sensors is becoming a more and more suitable solution for different metrological applications (Grimaldi & Rapuano, 2005), the devices proposed in these years are still limited in the performance (Coulouris et al., 1994).
If the protocol used to exchange information between server and client is developed by the manufacturer, the smart sensor becomes a close system in which only the producer can modify the service for the client (Bucci et al., 2005). Moreover, proprietary solutions make impossible to correlate different measures supplied by two different systems using a single system.
Software tools play a fundamental role to define the performance of advanced DMSs; for this reason in the following paragraphs we will analyse the different possibilities.

2.3.1 HTML, Java and XML

One of the most popular solutions adopts the HTML (*HyperText Markup Language*), the encoding scheme used to create and format a web document. All strings which constitute the markup either begin with the character "<" and end with a ">", or begin with the character "&" and end with a ";".

```
1 <html>
2       <body>
3             <p>Here is a paragraph.
4             <p>And here is another.
5       </body>
6 </html>
```

Fig. 6. Example of HTML style

HTML is used to generate static Web pages in which both formatting tags and information requests are present (Morelli et al., 2004). This involves that, usually, only the manufacturer knows the exact protocol and data format used by server and client to exchange information (Bucci et al., 2005); this implies insurmountable problems to develop different user applications (Bertocco et al., 1998).

A smart Web sensor based on the HTML technique creates, at every request or page refresh, a new complete Web page, rather than update only a specific page field, with a consequent useless data retransmission. Moreover, because the information is transmitted only in text format, other data formats (e.g. an image) require ad hoc solutions (Hrushal et al., 2005).

For example, to obtain a graphical representation of a data streaming it is necessary to leave HTML and use a system that allows the creation of sockets to exchange data. The socket is a standard technique to send data over a communication system; its use is more efficacious because the server sends only the specific data requested by the client.

Another limitation of HTML technique is evident in multisensor metrological applications where it is necessary to synchronize all the measurements; in this case it is often necessary to adopt a further protocol to synchronize the data transfer between servers and clients. But the synchronization protocol is non standard, complicating the implementation of some client applications.

The reach of these browsers is complemented by their support of the *Java* programming language, which can help implement sharing and distribution of functionality across a network (Grimaldi et al., 1997); (Michal & Wieslaw, 2001); (Knyziak & Wieslaw, 2003). Although usually perceived as a technology to animate web pages or to develop portable client applications, Java was designed to be a portable, clean, object-oriented language for small embedded systems. Java is network-aware (TCP/IP, Transmission Control Protocol/Internet Protocol) and provides support for dynamically downloadable code, as well as for communication between applications. Java gives more advantages, compared with HTML, and offers a method to delegate the graphical processing to the client with the Java Virtual Machine (JVM).

The combination of browser- and Java-based systems has tremendous benefits for integration of networked sensor applications into the enterprise.

2.3.2 XML structure of the sensor and communication protocol

On the Web, the *XML technology* is growing in importance, spreading as a new system for the data exchange between different platforms and different software tools (Han et al., 2000). XML stands for the eXtensible Markup Language (Han et al., 2000); (W3C, 2006) and is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. A markup language is a text language that enables to describe the data content in a way independent of hardware, software, formats, or operating system. Markup encodes a description of the document's storage layout and logical structure (Rusty, 2004).

XML describes a class of data objects called XML documents and partially describes the behaviour of computer programs which process them. XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and others form markup. XML provides a mechanism to impose constraints on the storage layout and logical structure.

XML gained widespread acceptance because of its simplicity in the application development. Even if XML's rules are strict, they make the burden of parsing and processing the files more predictable and simple.

XML presents an important advantage: the tags are not defined and the user itself can define the set of markup necessary for its application. In fact, the XML does not use a closed, limited and fixed set of markup tags, as defined for HTML, interpreted and used by the browser to represent the information. In XML the tags are defined by the user, which is free to represent any kind of information and any tree structure of the data source for the application that requires the service.

Defining a scheme that represents the information provided by the smart sensor in a complete and exhaustive way and defining the policy to follow for the correct exchange of information between the server (smart Web sensor) and the client (high level application) all based on XML, we have a complete Web product offering a measurement service through the internet network, platform and application-oriented independent (Amiano et al., 2006) .

The smart sensor needs to be visible to the client through an XML structure that describes its functionality, its use and its characteristic. In fact, the smart Web sensor architecture uses a hierarchical approach: the smart sensor can manage many transducers becoming a sensors node in a sensors network. The client will receive not only the information about the smart Web sensor but also the information about the transducers connected to it.

The smart Web sensor packages the measurements using the XML structure and sends all to the client, via the internet network. This device is charged for the communication between the client and the transducers and executes the commands imparted via the Web.

As an example of application we suggest a communication protocol that can be implemented for the definition of the packet sequence during a request. This protocol defines the structure of every XML streaming file involved in the communication and has been developed to reduce the overhead of XML tags indispensable for the communication (Benz & Durant, 2003). The main packets of the communication protocol are:

- CONNECTION: When a host needs to download data from the smart Web sensor, it will establish a connection resolving the IP address of the server.
- SETUP: When the client performs a CONNECTION, it receives a XML SETUP data streaming reporting the number of transducers connected to the smart Web sensor, its nature and the data type that they use as response for a request. In this streaming file more general information are present such as the description of the application, the

location of the sensor, the date and the release of the last firmware upgrade, the author of the system and the organization that developed the device.
- ACK: If the SETUP packet has been received correctly by the client and it interprets the data in the right way, the client sends an acknowledgment packet to the server.
- REQUEST: When the server receives an acknowledgment packet, it waits for a request packet from the client for the accessing of the data provided by a transducer connected to the server.
- DATA-SEND: After receiving the request packet, the server begins to process the acquired data from the sensor involved in the calling and then transmits a new datagram with the result. The form of the datagram depends on the dimension of the data type that represents the measurement.

3. The Web service technology

The use of the XML as streaming support of measurement results is a good solution for all the remote measuring applications. However, XML presents a limitation: even if the streaming support is open, well organized and cross platform, the way used by client and server to exchange XML streaming data is proprietary. These problems present important limitations in the development of complex sensors network (Ferrari et al., 2003).
The basic requirement beyond smart Web sensor is the needing to have in some way the accessibility to some measured value (Bucci et al., 2003). The supplying of this value can be seen as a service done by an embedded server that is accessible on Internet. Every server allows the client to access the information acquired from a sensor.
A different approach to Web sensor development is based on the new concept of server that has been developed by the W3C (World Wide Web Consortium) (http://www.w3.org/): the idea is to consider a Web server not only as a stand alone server that a client can access to download files or HTML pages, but also a Web component that supply a service on the Internet network (Mielcarz & Winiecki, 2005). This solution, known as *Web service* approach, transforms a smart Web sensor into a server of measurement functions. In this way it is possible to offer great possibilities in terms of easy access for measurement data, integration of large complex Web sensors networks, realization of flexible custom applications and services reusability. Every client or developer can use this service to obtain information or to develop new complex services starting from the received information.
It is important to underline that Web services are similar to the local components used to build Windows applications (COM Object) with the method and attribute that the COM (Component Object Model) Object provides to the developer, but they aren't physically present in the local machines.
In the past, clients accessed these services using a tightly coupled, distributed computing protocol, such as DCOM (Distributed Component Object Model), CORBA (Common Object Request Broker Architecture), or RMI (Remote Method Invocation). While these protocols are very effective for building a specific application, they limit the flexibility of the system. Specifically, is the tight coupling used in these protocols (dependencies on vendor implementations, platforms, languages, or data encoding schemes) that limits the reusability of individual services.
The Web service architecture takes all the best features of the service-oriented approach and combines it with the Web, supporting universal communication using loosely coupled

connections. Web protocols are completely vendor-, platform-, and language-independent. Web services support Web-based access, easy integration, and service reusability.

3.1 Smart Web sensors based on Web services

As previously discussed, the today's smart Web sensors present in literature adopt a micro-embedded Web server to transfer data and information to the clients that perform the request. As an application, starting form a low cost widely adopted smart Web sensor (Castaldo et al., 2003) ; (Castaldo et al., 2004) (Testa et al. , 2004) , a new kind of smart Web sensor with the Web service functionality is proposed; its simplified block diagram is shown in Fig. 7.

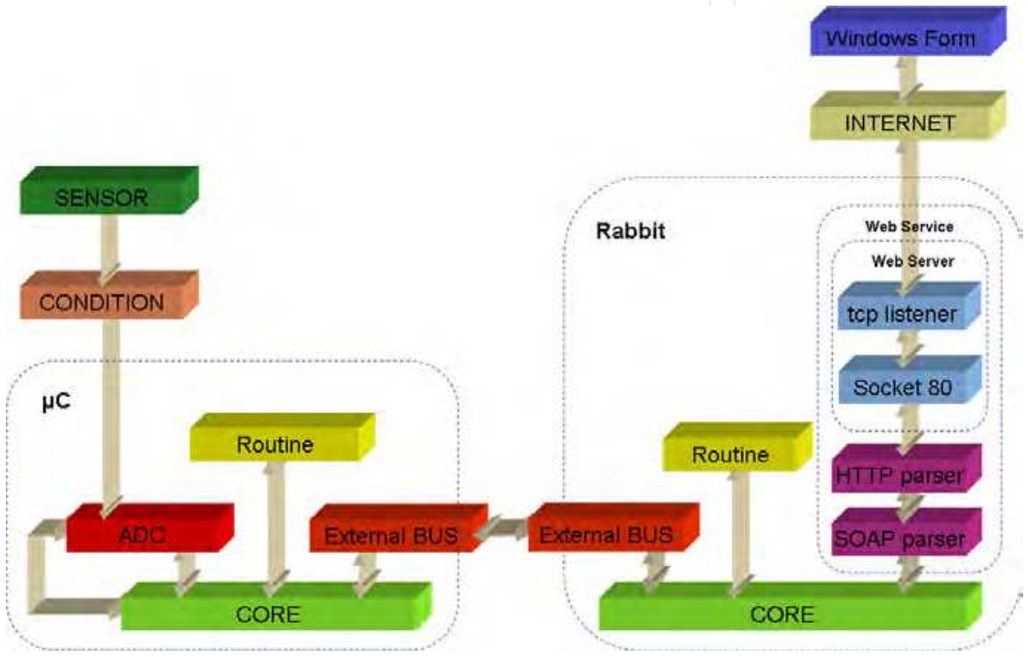


Fig. 7. Simplified block diagram of a smart Web sensor based on Web services

For including a Web service in a server environment, the main and widely adopted software architecture is ASP.NET, available in Microsoft Visual Studio .NET.
However, the use of a real Web service determines hard constraints on a general embedded architecture in term of cost, portability and power consumption. For these reasons, a possible solution for the developed of embedded Web service server is the use of a low cost embedded Web server.
In general a Web server does not have the same functionally of a Web service because of the use of HTTP (as protocol for sending data packets), HTML (to display information to a browser) and SOAP, Simple Object Access Protocol, (to exchange data with a client or with a Web service), while a Web server manages only HTTP and HTML.
As reported in Fig. 7, the communication system emulates a Web service opening a socket on port 80 for the listening of all the packets; then, a HTTP and SOAP parser controls and responses to the SOAP messages.

The most remarkable aspect of the entire flow is the waiting time of the Windows Form during a request; this time depends on the network load and on the number of samples acquired by the microcontroller. When the Windows Form sends a request on HTTP with a SOAP message to the Light Web service, it waits a SOAP response (an XML streaming file) in which the waveform is serialized. During this time, the Windows Form doesn't execute any other thread and it waits for the SOAP message.
To continue to use the Windows Form, it is necessary to control the thread of the Windows Form otherwise the process seizes up and any operation can be run (see Fig. 8).

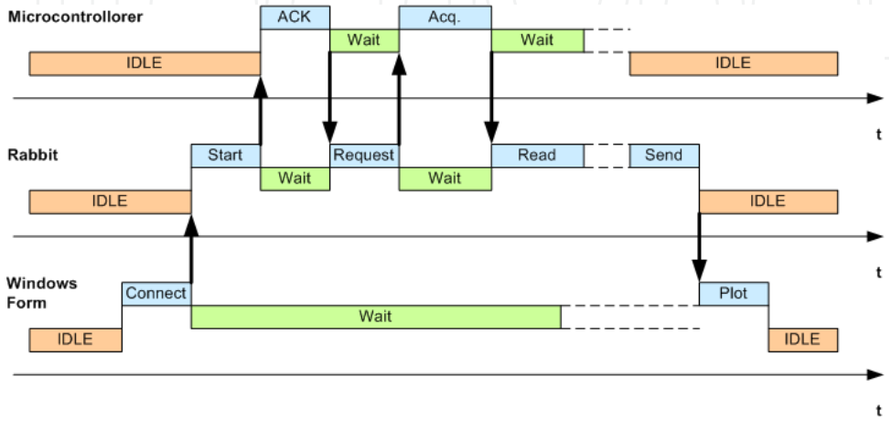


Fig. 8. Time analysis of the tasks present in the whole system

4. Plug-n-play smart Web sensors based on Web services

In a DMS, based on this technology, the services published by a Web service are reported in the WSDL (Web Services Definition Language) file.
Unfortunately, the Web service technology does not give any mechanism to refresh the services published and to manage dynamically the new services exported or deleted (Mielcarz & Winiecki, 2005). For instance, the access to a deleted service by a distributed application can generate an exception, collapsing the whole system and switching off the application. This is a stiff limitation, especially for a network of sensors that are often reconfigured to perform different measurements (Bucci et al., 2001), that require an appropriate run-time control for managing these service problems. Therefore, it is very important to develop a methodology to create a network in which smart Web sensors (network nodes) can be plugged without the need for an external configuration (Bucci et al., 2007).
A suitable solution is that every sensor sets an IP address and communicates its ability to the network master, who has two functions: master of the entire network and gateway (Ciancetta et al., 2007) Besides, the network master provides a Web service interface to every client that wants to use the sensors network: the entire network is equivalent to a single dynamic Web service (Ciancetta et al., 2006).
The core of this new approach is the adoption of two different tables in the smart sensors network: IP Routing Table and Services Table. The IP Routing Table is a table necessary to route a request from a client. This table stores the IP address and the services of every node; allowing the server to join the network node with its services. So, every request from a client

can be sent to the specific node. However, the client request has a different approach: the client sends a request to the server that, consulting its IP table routing, decides if it can execute the request. Next, the server sends a request to the network node present in the table to await the response and re-sends it to the client. This operation works well if there is a request to a specific service present in the network.
The main advantage of this solution is the possibility to merge more services to implement another new service. For example, we can imagine having a sensors network with two nodes: a voltage measurement sensor and a current measurement sensor. Besides the voltage or current services, the server can create other "virtual" services by fusion of the existing services. As an example, power or resistance can be "virtually" measured starting from these two services and the server can show four different services stored in the Service Table. This table, showing all the services available to the client and how they can be implemented, is upgraded every time a new sensor, executing new services, is plugged.
The service table describes whether the service is direct (not virtual) or virtual as shown in Fig. 9. A direct service is directly connected to a node, so, the Web service consults its IP table routing to resolve it. On the contrary, if a client sends a virtual request, the Web service consults an execution table, where the service is linked with a specific function related to actual devices.

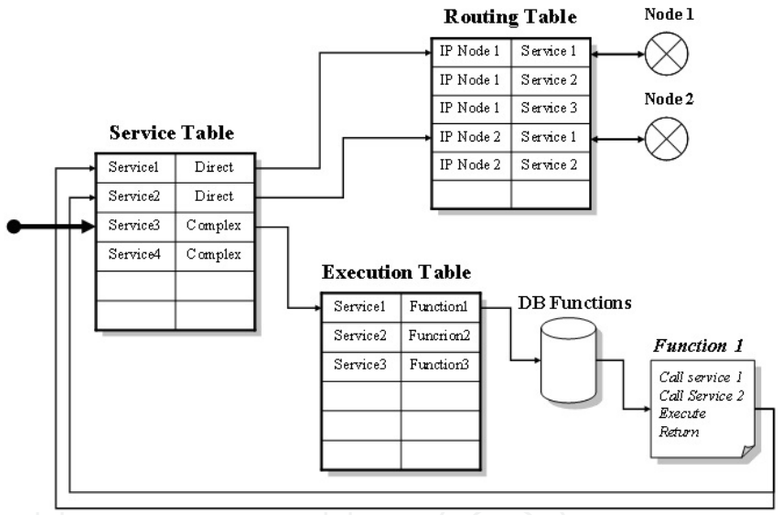


Fig. 9. IP Routing Table and Service Table

The Web service presents a DataBase (DB) storing all the executable functions. A typical function executes these tasks: i) reserves the required memory to every element; ii) receives all the values from the services involved in the function; iii) performs all the operations necessary to have the correct result; iv) gives the result to the Web service that resends it to the client, using SOAP.
The Fig. 9 illustrates how a Web service deals with a virtual service received from a client. The execution table is consulted to know whether the Web service can perform the function. Then the service table is consulted, to find the services it requires. Adopting this technique, it's possible to execute a virtual service by means of other virtual services. The service table has an important role in this approach. Every time a new network node is plugged in a sensors network, the Web service maps all the direct services available on the node,

upgrading the IP routing table. Moreover, it scans all the DB functions that can be performed, to correctly execute virtual services.

5. A peer-to-peer distributed system for multipoint measurement techniques

To implement a DMS based on smart Web sensors, it is necessary to use a common and open communication protocol to exchange information and a methodology to auto-configure any smart sensor is linked to the network. Peer-to-peer networks allow individual computers to communicate directly with each other and to share information and resources without using specialized servers. A common characteristic of this new breed of applications is that they build, at the application level, a virtual network with its own routing mechanisms. The topology of this virtual network and the adopted routing mechanisms has a significant influence on the application properties such as performance and reliability (Ripenanu, 2001). Significant advantages can be gained using a freeware and widely adopted technology, such as the *Gnutella*.
The Gnutella protocol (The Gnutella protocol specification v4.0) .is an open, decentralized group membership and search protocol, mainly used for file sharing. The term Gnutella also designates the virtual network of Internet accessible hosts running Gnutella-speaking applications (this is the Gnutella network) and a number of smaller, and often private, disconnected networks. The graph in Fig. 10 depicts the topology of peers forming a connected segment of the Gnutella network.

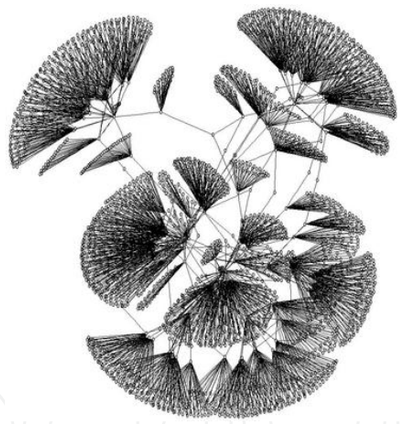


Fig. 10. A representation of the topology of Gnutella Network

Like most peer-to-peer file sharing applications, Gnutella was designed to meet the following goals:

- *Ability to operate in a dynamic environment*. Peer-to-peer applications operate in dynamic environments, where hosts may join or leave the network frequently. They must achieve flexibility in order to keep operating transparently despite a constantly changing set of resources.
- *Performance and Scalability.* The peer-to-peer paradigm shows its full potential only on large-scale deployments where the limits of the traditional client/server paradigm become obvious. Moreover, scalability is important as peer-to-peer

applications exhibit what economists call the ”network effect” (Makarenko et al. 2004): the value of a network to an individual user scales with the total number of participants. Ideally, when increasing the number of nodes, aggregate storage space and file availability should grow linearly, response time should remain constant, while search throughput should remain high or grow.

- *Reliability.* External attacks should not cause significant data or performance loss.
- *Anonymity.* Anonymity is valued as a means of protecting the privacy of people seeking or providing unpopular information.

Gnutella nodes, called *servents* by developers, perform tasks normally associated with both SERVers and cliENTS. They provide client-side interfaces through which users can issue queries and view search results, accept queries from other servents, check for matches against their local data set, and respond with corresponding results. These nodes are also responsible for managing the background traffic that spreads the information used to maintain network integrity.

The *Ultrapeer* is an important concept that was not specified in the original Gnutella protocol, but which has now become a prominent feature of the Gnutella network. The Ultrapeer scheme improves network efficiency and scalability by categorizing nodes into regular clients and super nodes. A super node is a reliably connected host with plenty of network bandwidth that can act as a proxy for a large number of connecting clients. The super node removes the burden of extensive network message routing from the client, which may be a low bandwidth modem user. With this scheme, the Gnutella network mimics the Internet itself: low bandwidth nodes are connected to larger routers (the super nodes) that transmit the majority of the data over high bandwidth backbones.

As an example of using the Gnutella network, we describe a network that allows linked hosts to share arbitrary resources. This is a decentralized peer-to-peer system, consisting of hosts connected to one another using TCP/IP. In this network a client request for a measurement application is addressed to a computer which performs a particular Web service (Gnutella Web Service). This systems use the Gnutella network to search all the users able to perform the specific measurement, called Gnutella Embedded Clients (GECs) as reported in Fig. 11.

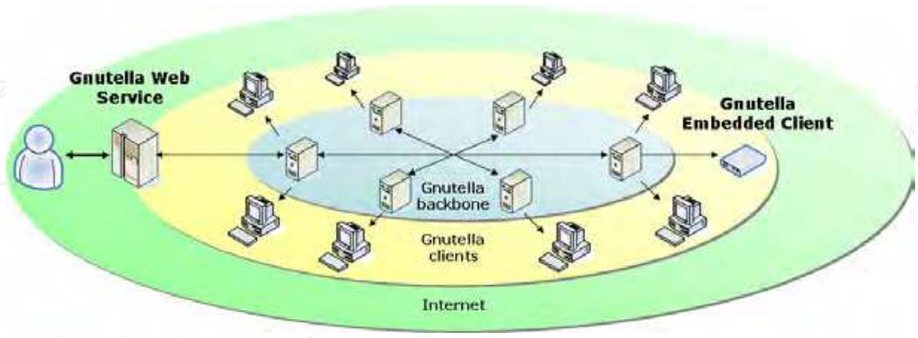


Fig. 11. Distributed architecture of a Gnutella measurement network

The name client for GEC is because it is a client of the Gnutella network. To execute the user search the request (query message) is repeated to all the Gnutella network computers (Fig. 12). When the suitable user is found, this network sends back the GEC address to the client.

At this point, the client can download the measures directly from the GEC, without overloading the Gnutella network (Bucci et al., 2005).
In this system, the measurement points are the GECs; each GEC can perform special measurements, depending on the kind of sensors embodied. This network creates an Internet over-structure from which all clients can perform a free access without external configuration and the GECs are visible without special operations. In order to implement this kind of system, a special Gnutella Web Service, a kind of interface between the client and the Gnutella network (Fig. 13) has been implemented, because the current implementations, referring exclusively on files sharing, cannot support a measurement process. When a measurement operation is asked, GEC sends the results to the *Gnutella Web Service* (GWS). One of the advantages of the proposed solution is the simplification of the activities to search and locate the measurement systems (GECs).

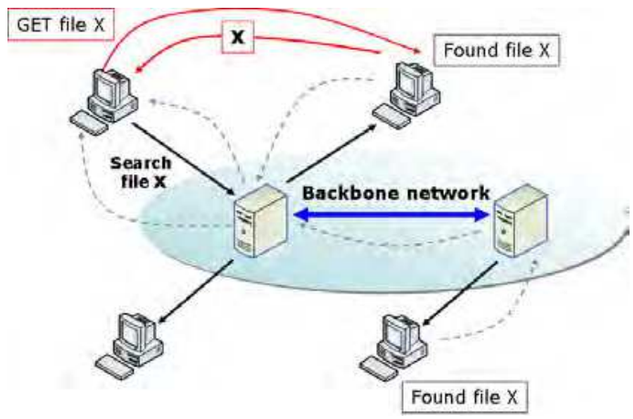


Fig. 12. The measurement server search, route and download

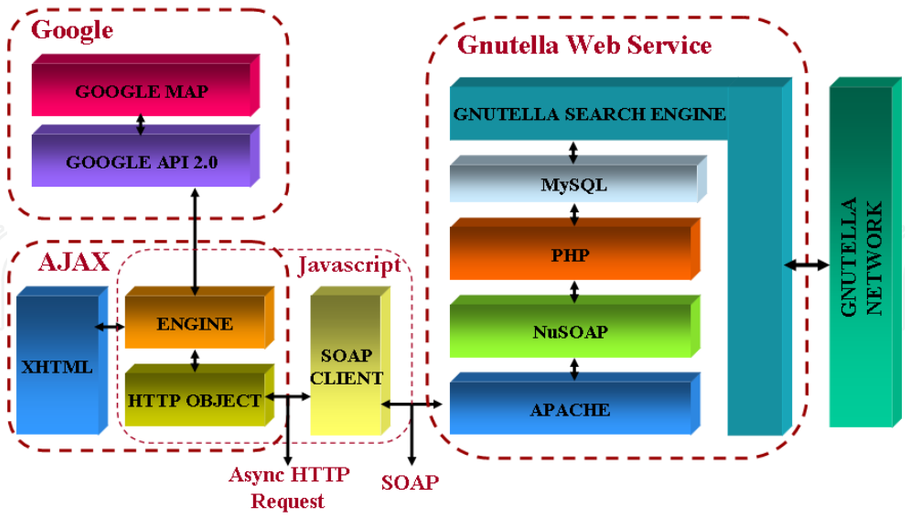


Fig. 13. Architecture of the implemented Web Service and User Interface

The GWS provides a particular implementation of typical Gnutella software, developing an ad-hoc Gnutella Search Engine. The methods are specifically developed for a measurement application; in particular the exported methods are:

1. GetStations: to obtain information about the stations present in a limited geographic area defined by GPS coordinates, in order to restrict the searching. The output of the method gives an array of stations in which every one reports.
2. GetCurrentData: the user calls the method passing the HASHID (hash identification)of the remote station and the service request to obtain the current data.
3. GetHistoryData: is similar to GetCurrentData, but accesses to stored DB data.

The Gnutella network is time consuming during the searching. In order to reduce this time, we adopted a caching system: at the end of a search, the authenticated stations are cached and their IP address stored in a DB for a limited period. Therefore, to obtain some information from a particular station, it is not necessary to start a new search, but it is possible to directly perform the download.

5.1 Environment monitoring application

In order to evaluate the feature of the proposed architecture, we implemented a monitoring application able to measure atmospheric values (Manuel et al., 2005), (Simic & Sastry, 2003) developing a remote measurement system (GEC), a GWS and a Web interface between the server and the operator (Ciancetta et al., 2007), (Ciancetta, Bucci et al. 2007).
The Web user interface has been implemented as a XHTML (eXtensible HyperText Markup Language) page that sends a request to Web Service and displays the results using Google Map (Fig. 14).

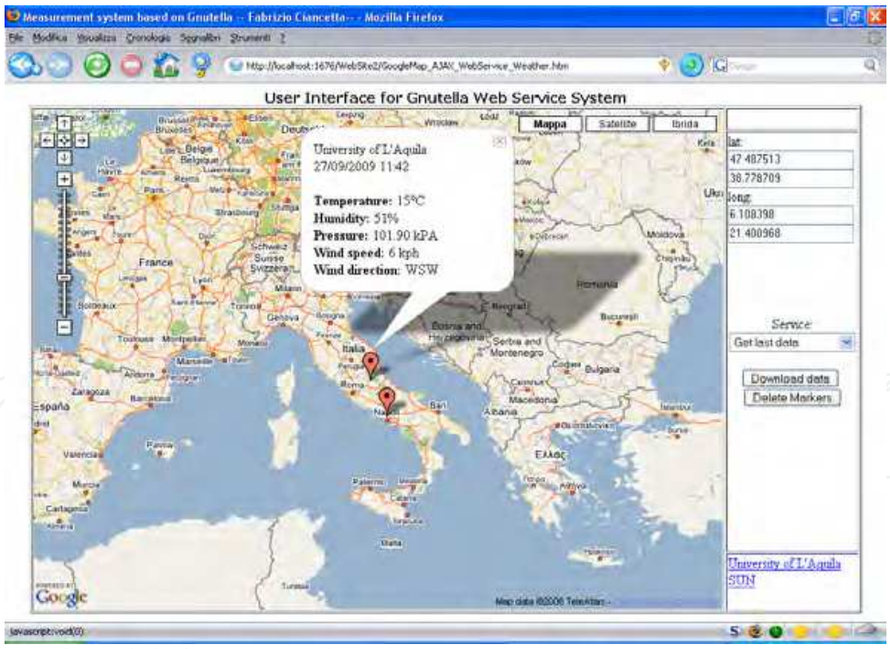


Fig. 14. Screenshot of Web user interface.

The Web user interface gives a more degree of freedom to the whole system, allowing the user to directly access measurement information with a common browser. We used Ajax (Asynchronous JavaScript and XML) technology to create interactive Web applications.

The XHTML page sends asynchronous requests to the Web Service and installs a callback function on the XMLHttpRequest. All the management of the function is done in JavaScript.
To interface the XHTML-JavaScript page with GWS, we adopted a SOAP client, a JavaScript class able to receive/create XML data form XHTML page and create/receive SOAP packet to GWS. In particular, on the remote station we implemented the services: temperature, humidity, pressure, wind direction and speed as shown in the Google Map Balloon accessible directly on the map.
To provide a more powerful mode to represent data from Gnutella Embedded Client we suggest a Windows Form user interface, based on Framework .NET 2.0. In the example, the user interface is divided in two parts: the first part, placed on the right side of the Windows Form, in which the user can: i) list the GECs present in the geographic area limited by the GPS coordinates; ii) select a station, looking at the available services and its GPS coordinates; iii) see a geographic view of all the station involved in the search. On the left Windows Form side there are two panels, reporting the downloaded data.
In the Current Data Panel (Fig. 15) there is a current view of the station with the last stored data acquired by the Gnutella Embedded Client and a graphical view of all the data of the current day from the 0:00 to the current hour retrieved form the GEC DB. The History Data Panel (Fig. 16) performs a direct access to the Gnutella Embedded Client DB, downloading the data.
In this example, all data are accessible directly to the GEC, without using the Gnutella network to reduce the traffic. In order to reduce space there are two DBs: one for the values accumulated during the day and another for an historical trend of the measurements.

6. Sensor synchronization

In a DMS time synchronization is a very important feature; many applications need local clocks of sensor nodes to be synchronized, requiring various degrees of precision. Unfortunately clock devices generate signals with some relative time uncertainties: local clock signals may drift from each other in time, hence sampling time or durations of time intervals may differ for each node in the network.
In general, a DMS can require different clock synchronization. The simplest case is the need to order the measures, that is to determine whether a measure m_1 carried out by a sensor has occurred before or after another measure m_2 carried out by another one. This problem presents simple solutions, because it is just required to compare the local clocks rather than to synchronize them.
Another more important occurrence is when each node embodies an independent clock and it is necessary to obtain information about the deviation from the other clocks in the network. In this way each node has its own local clock, but it is possible to convert a local time to the local times of other nodes. The majority of the synchronization procedures proposed for sensor networks use this technique (Elson et al., 2002); (Greunen &, Rabaey, 2003); (Sichitiu & Veerarittiphan, 2003)
The most complex situation is when all nodes must maintain a local clock synchronized to a remote reference clock. This is, for example, the case of two sensors sampling voltage and current that must be synchronized for calculating the electrical power. The synchronization scheme of (Ganeriwal et al., 2003) conforms to this model.

The synchronization methods are generally based on message exchange between nodes. In effect the problem is complicated by the nondeterminism in the network data access time, typical of Ethernet, characterized by a random access time, and in the variable packet transmission time. If a node transmits a measure with the local timestamp to another node or client, the packet can have a variable amount of delay before it is delivered, precluding the possibility of comparing and synchronizing the two clocks. Other access techniques, such as the TDMA (Time Division Multiple Access) can eliminate the uncertainty on the access time, but not on the transmission time.

Traditional synchronization techniques such as the use of a global positioning system (GPS) are not suitable for use in sensor networks; a GPS device may be too expensive to attach on a small sensor devices, and GPS service may not be available everywhere, such as inside a building. Moreover, this problem becomes important especially for a network of wireless smart sensors, because of their intrinsic properties such as limited resources of energy, storage, and computation.

To solve this problem, several solutions are under study in terms of synchronization algorithms, specifically designed for sensor networks.

The most diffused protocol is the Reference Broadcast Synchronization (RBS) (Elson & Estrin, 2001) where the sensors are divided in clusters, each with a cluster-head that transmit a synchronization packet (beacon). A reference beacon does not include a timestamp, but instead, its time of arrival is used by receiving nodes as a reference for comparing clocks. All receivers record the packet arrival time. The receiver nodes then exchange their recorded timestamps and estimate their relative phase offsets. RBS also estimates the clock skew by using a least-squares linear regression. The interesting feature of RBS is that it records the timestamp only at the receivers, thus, all timing uncertainties, including MAC (Media Access Control) medium access time, on the transmitter's side are eliminated. This characteristic makes it especially suitable for hardware that does not provide low-level access to the MAC layer. The main disadvantage of RBS is that it does not synchronize the sender with the receiver directly and that, when the programmers have low-level access at the MAC layer, simpler methods can achieve a similar precision to RBS.

Another protocol is the Flooding Time Synchronization Protocol (FTSP) or Tiny-Sync. FTSP, designed for applications requiring very high precision, utilizes a customized MAC layer time-stamping and calibration to eliminate unknown delays (Mar´oti et al., 2004). Linear regression from multiple timestamps is used to estimate the clock drift and offset. The main drawback of FTSP is that it requires calibration on the hardware actually used in the deployment (it is not a simply software algorithm). FTSP also requires intimate access to the MAC layer for multiple timestamps. However, if well calibrated, the FTSP's precision is less than 2 μs.

The Precision Time Protocol (PTP) is a high precision time synchronization protocol, defined in the IEEE 1588 standards "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems". There are two steps for synchronizing devices using PTP: (1) determine which device serves as the master clock, and (2) measure and correct time skew caused by clock offsets and network delays. When a system is initialized, the protocol uses an algorithm to determine which clock (Master Clock) in the network is the most precise. All other clocks become slaves and synchronize their clocks with the master. Because the time difference between the master clock and slave clock is a combination of the clock offset and message transmission delay, correcting the clock skew is

done in two phases: offset correction and delay correction. Accuracy in the sub-microsecond range may be achieved with low-cost implementations.

7. Conclusions and future trends

Smart sensors are an enabling technology that will influence the future applications of measurement and data acquisition systems distributed on a wide area. The main revolutionary aspect of DMSs is the advanced integration of many state-of-the-art enabling technologies, mainly sensor, wireless communication, positioning, tracking and information technologies.

The first consequence of present trends is the supposition that in the future all sensors will be smart to some degree. Certainly a much higher percentage of them will be self-identifying and communicating. Communication is an important requirement for these devices and Internet, with either wired or wireless links, another widely shared solution.

It will be hard to solve all the problems in a "standard" way, also because there are several different applications with conflicting requirements. Proprietary solutions will be proposed again, especially for industrial applications. A plentiful supply of software tools for information and communication applications can help the DMS developers; even if the needs of a network of measurement systems are substantially different from a network of computers or communication devices.

In this chapter we tried to give an overview of the actual possibilities and trend in this field, even if the evolution run very fast and every day new standards and tools are available.

8. References

Amiano, M., Cruz, C., D., Ethier, K. and Thomas, M., D. (2006*), XML Problem - Design - Solution*, Wiley, 2006. ISBN-13: 978-0-471-79119-5, ISBN-10: 0-471-79119-9.

Benz, B. and Durant, J., R. (2003) *XML Programming Bible*, Wiley, 2003. ISBN-10: 0-7645-3829-2

Berkes, J., E. (2003), *Decentralized Peer-to-Peer Network Architecture: Gnutella and Freenet*, University of Manitoba, Winnipeg, Manitoba, Canada, April, 2003.

Bertocco, M., Ferraris, F., Offelli C. and Parvis, M. (1998), *A Client-Server Architecture for Distributed Measurement Systems* , Proceedings of IEEE Instrumentation and Measurement Technology Conference St. Paul, Minnesota, USA, May 18-21, 1998 pp 67–72.

Bucci, G., Ciancetta F., Fiorucci, E., Gallo, D. and Landi, C. (2005), *A low cost embedded Web Services for measurements on power system*, Proceeding of IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems, Giardini Naxos, Italy, 18-20 July, 2005.

Bucci, G., Ciancetta, F. and Fiorucci E. (2005), *A DSP-Based Wireless and Modular Data Acquisition Unit for Real-Time Measurement*, TechOnline Technical Papers, March 16, 2005 www.techonline.com.

Bucci, G., Ciancetta, F. and Fiorucci, E. (2003), *Unità d'acquisizione dati remota per sistemi di misura e controllo su rete TCP/IP*, Proceedings of Convegno Misure & Energia: l'importanza della metrologia nellindustria energetica italiana, Milano, 25 Novembre 2003.

Bucci, G., Ciancetta, F. and Rotondale, N. (2007), *Rete di sensori Plug-N-Play basata sui servizi Web: applicazioni al controllo di processi industriali*, Proceeding of LI Convegno Nazionale Motion Control, ANIPLA 2007, Milano, Italy, 10-11 Maggio 2007.

Bucci, G., Fiorucci, E. and Landi, C. (2001), *Digital Measurement Station for Power Quality Analysis in Distributed Enviroments*, Proceeding of IEEE International Conference on Instrumentation and Measurement Technology Conference, Budapest, Hungary, May 21-23,2001, pp 368–373.

Castaldo, D., Gallo, D. and Landi, C. (2004), *Collaborative Multisensor Network Architecture Based On Smart Web Sensor for Power Quality Applications*, Proceedings of IEEE International Conference on Instrumentation and Measurement Technology Conference, Como, Italy, 18-20 May, 2004, pp 1361– 1366.

Castaldo, D., Gallo, D., Landi, C., Langella, R. and Testa, A. (2003), *A Distributed Measurement System for Power Quality Analysis*, Proceedings of IEEE Power Tech 2003, Bologna, Italy, June 23-26, 2003.

Chu X., Kobialka T., Durnota B., and Buyya R. (2006). Open Sensor Web Architecture: Core Services, *Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing (ICISIP 2006)*. ISBN 1-4244-0611-0, pp.:98-103. Bangalore, India.

Ciancetta, F., Bucci, G., Fiorucci, E., D'Apice, B. and Landi, C. (2007), *Proposta di un sistema di misura distribuito basato su una rete Peer-To-Peer*, Proceeding of XXIV Congresso nazionale GMEE (Gruppo Nazionale di Coordinamento Misure elettriche ed Elettroniche), Torino, Italy, 5-8 Settembre 2007.

Ciancetta, F., D'Apice, B., Gallo, D. and Landi, C. (2006), *Sistema di misura distribuito basato sui sensori smart e servizi Web*, Proceeding of XXIII Congresso nazionale GMEE (Gruppo Nazionale di Coordinamento Misure elettriche ed Elettroniche), L'Aquila, Italy, 11-13 Settembre 2006.

Ciancetta, F., DApice, B., Landi, C. and Pelvio, A. (2007), *Sistema di misura distribuito per il monitoraggio di rete di potenza*, Proceeding of XXIV Congresso nazionale GMEE (Gruppo Nazionale di Coordinamento Misure elettriche ed Elettroniche), Torino, Italy, 5-8 Settembre 2007.

Ciancetta, F., Fiorucci, E., D'Apice, B. and Landi, C. (2007), *A Peer-to-Peer Distributed System for Multipoint Measurement Techniques*, Proceedings of IEEE Instrumentation and Measurement Technology Conference, Warsaw, Poland, May 1-3, 2007, pp 1–6

Coulouris, G., Dollimore, J. and Kindberg, T. (1994) *Distributed Systems, Concepts and Design*, Addison-Wesley, Reading, MA, 1994.

Elmasri, R., and Navathe, S., B. (1994) *Fundamentals of Database Systems*, Addison-Wesley, Reading, MA, 1994.

Elson, J. and Estrin, D. (2001), *Time synchronization for wireless sensor networks,* In Proc. of the 2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing. San Francisco, CA.

Elson, J., Girod, L. and Estrin, D. (2002), *Fine-Grained Time Synchronization using Reference Broadcasts*, Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.

Ferrari, P., Flammini, A., Marioli, D., Sisinni, E. and Taroni, A. (2003), *Sensor integration in Industrial Environment: From Field-bus to web-sensors*, Computer standards & Interfaces, 25, 2003.

Ganeriwal, S., Kumar, R. and Srivastava, M. (2003), *Timing Sync Protocol for Sensor Networks*, Proceedings of ACM SenSys, Los Angeles, November 2003.

Greunen, J. and V., Rabaey, J. (2003), *Lightweight Time Synchronization for Sensor Networks*, Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA), San Diego, CA, September 2003.

Grimaldi, D., Nigro, L. and Pupo, F. (1997), *Java based distributed measurement systems*, Proceedings of IEEE Instrumentation and Measurement Technology Conference, 19-21 May 1997, Ottawa, Canada, pp 686–689.

Grimaldi, D., Rapuano, S. and Laopoulos, T. (2005) *State of Art of the Distributed Measurement Systems for Industrial and Educational Purposes*, IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 5-7 September 2005, Sofia, Bulgaria pp 289–294.

Hamrita, T.K. Kaluskar, N.P. Wolfe, K.L. (2005). Advances in smart sensor technology. *Proc. Of Industry Applications Conference, 2005*. ISBN: 0-7803-9208-6. Volume: 3, pp.: 2059 – 2062.

Han, R., Perret, V. and Naghshineh M. (2000), *WebSplitter: A Unified XML Framework for Multi-device Collaborative Web Browsing*, Computer Supported Cooperative Work, pp 21–23.

Hrushal, V., Osolinskiyl, O., Daponte P. and Grimaldi D.(2005), *Distributed Web-based Measurement System*, Proceedings of IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 5-7 September 2005, Sofia, Bulgaria pp 355–358.

IEEE Standard for a Smart Transducer Interface for Sensors and Actuators, IEEE Std 1451.1-4, 1997, http://ieee1451.nist.gov/.

Knyziak, T. and Winiecki, W. (2003), *The New Prospects of Distributed Measurement Systems Using JavaTM 2 Micro Edition Mobile Phone*, Proceedings of IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing System: Technology and Applications, 8-10 September 2003, Lviv, Ukraine, pp 291–295.

Makarenko, A., Brooks, A., Williams, S., Durrant-Whyte, H. and Grocholsky B. (2004), *A decentralized architecture for Active Sensor Networks*, Proceedings of IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, April 26-May 1, 2004, pp 1097–1102.

Manuel, A., DelRio, J., Shariat, S., Piera, J. and Palomera, R. (2005), *Software Tools for a Distributed Temperature Measurement Systems*, Proceedings of Instrumentation and Measurement Technology Conference Ottawa, Ontario, Canada, May 17-19, 2005, pp 1566–1570.

Mar´oti, M., Kusy, B., Simon, G., and L´edeczi, A. (2004) *The flooding time synchronization protocol,* Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys ’04, ACM Press, 39–49.

Michal, K. and Wieslaw, W. (2001), *A New Java-Based Software Environment for Distributed Measurement Systems Designing*, Proceedings of IEEE Instrumentation and Measurement Technology Conference, 21-23 May 2001, Budapest, Hungary, pp 397–402.

Mielcarz, T. and Winiecki, W. (2005), *The Use of Web-services for Development of Distributed Measurement Systems*, Proceedings of IEEEWorkshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 5-7 September 2005 ,Sofia, Bulgaria, pp 320–324.

Morelli, S., Morelli, R., Ciancetta, F., Vasile, A., D'Intino, A., Di Donato, M., A., Di Gioacchino, M. and Boscolo P. (2004), *Monitoraggio dei campi elettromagnetici nelle aree urbane di Chieti e Pescara*, Proceedings of LXVII Congresso Nazionale S.I.M.L.I.I., Sorrento, Italy, 3-6 Novembre 2004, pp 301-302.

Ozsu, T. and Valduriez, P. (1991) *Principles of Distributed Database Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

Ripeanu, M. (2001), *Peer-to-Peer Architecture Case Study: Gnutella Network Analysis*, 1st International Conference in Peer-to-Peer Networks, Aug. 2001, Linkpings Universitet, Sweden.

Rusty, H., E. (2004) *XML 1.1 Bible*, Wiley, 2004. ISBN-10: 0-7645-4986-3.

Sichitiu, M.,L. and Veerarittiphan, C. (2003), *Simple, Accurate Time Synchronization for Wireless Sensor Networks*, Proceedings of IEEE Wireless Communications and Networking Conference, WCNC 2003.

Simic, S. and N., Sastry, S. (2003), *Distributed environmental monitoring using random sensor networks*, Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks, Palo Alto, California, April 22-23, 2003, pp 582–592.

Tanenbaum, A., S. (1992) *Modern Operating Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1992.

Tari, Z. and Bukhres, O. (2001) *Fundamentals of Distributed Object Systems: The CORBA Perspective*, Wiley, 2001.

Testa, A., Castaldo, D., Gallo, D. and Landi, C. (2004), *A Digital Instrument for non-Stationary Disturbance Analysis in Power Lines*, IEEE Transactions on Instrumentation and Measurement , 53, no. 5, August, 2004, pp 1353–1361.

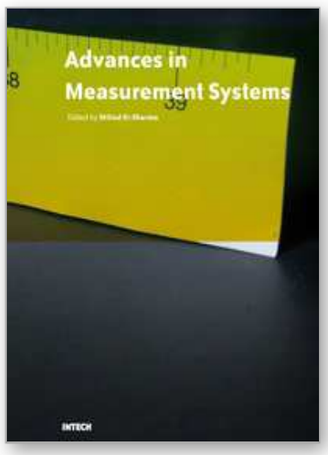
The Gnutella protocol specification v4.0. http://dss.clip2.com/GnutellaProtocol04.pdf., 2004.

Viegas V., Dias Pereira J. M., Silva Girão P. (2007). Framework and Web Services: A Profit Combination to Implement and Enhance the IEEE 1451.1 Standard. *IEEE Transactions on Instrumentation and Measurement, Volume 56*.NET, Issue 6, pp. 2739-2747, December 2007.

W3C, *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, http://www.w3.org/TR/2006/REC-xml-20060816/ 2006.

Yong Z.; Yikang G.; Vlatkovic, V.; Xiaojuan W. (2004). Progress of smart sensor and smart sensor networks. *Proc. of Intelligent Control and Automation*, 2004. WCICA 2004. Digital Object Identifier 10.1109/WCICA. 2004.1343265. Volume 4, pp.: 3600 – 3606.

IntechOpen



Advances in Measurement Systems

Edited by Milind Kr Sharma

ISBN 978-953-307-061-2
Hard cover, 592 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Giovanni Bucci, Fabrizio Ciancetta and Edoardo Fiorucci (2010). New Technologies For Measurement Systems Distributed On A Wide Area, Advances in Measurement Systems, Milind Kr Sharma (Ed.), ISBN: 978-953-307-061-2, InTech, Available from: http://www.intechopen.com/books/advances-in-measurement-systems/new-technologies-for-measurement-systems-distributed-on-a-wide-area

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen