Project Acronym: MEDIS

Project Title: A Methodology for the Formation of Highly Qualified Engineers at Masters Level in the Design and Development of Advanced Industrial Informatics Systems

Contract Number: 544490-TEMPUS-1-2013-1-ES-TEMPUS-JPCR

Starting date: 01/12/2013 **Ending date:** 30/11/2016

Deliverable Number: 2.4.1

Title of the Deliverable: AIISM teaching resources.

Task/WP related to the Deliverable: Development of the AIISM teaching resources - Industrial Networks and Fieldbuses - Introduction

Type (Internal or Restricted or Public): Public

Author(s): Luis Almeida, Mario Sousa

Partner(s) Contributing: UP- University of Porto

Contractual Date of Delivery to the CEC:

Actual Date of Delivery to the CEC:

Project Co-ordinator

Company name :	Universitat Politecnica de Valencia (UPV)
Name of representative :	Houcine Hassan
Address :	Camino de Vera, s/n. 46022-Valencia (Spain)
Phone number :	+34 96 387 7578
Fax number :	
E-mail :	husein@upv.es
Project WEB site address :	http://medis.upv.es/

Context

WP 2	Design of the AIISM-PBL methodology
WPLeader	Universitat Politècnica deValència (UPV)
Task 2.4	Development of the AIISM teaching resources – Industrial Networks and Fieldbuses - Introduction
Task Leader	UP
Dependencies	UPV, MDU, TUSofia, USTUTT, UP

Author(s)	Luis Almeida, Mario Sousa
Contributor(s)	
Reviewers	

History

Version	**Date**	**Author**	**Comments**
0.1	22/03/14	Luis Almeida	Initial draft
0.2	19/09/14	Luis Almeida, Mario Sousa	Final version

Table of Contents

1 Executive summary....4
2 Lecture....4
3 Lab – Week 1....18
4 Lab – Week 2....20
5 Seminar – Week 1....22
6 Seminar – Week 2....23
7 Mini-project – Week 1....23
8 Mini-project – Week 2....27

1 Executive summary

WP 2.4.1 details the learning materials of the Advanced Industrial Informatics Specialization Modules (AIISM) related to the Industrial Networks and Fieldbuses.

The contents of this package follows the guidelines presented in the Partner's documentation of the WP 1 (Industrial Networks and Fieldbuses)

- The PBL methodology was presented in WP 1.1
- The list of the module's chapters and the temporal scheduling in WP 1.2
- The required human and material resources in WP 1.3
- The evaluation in WP 1.4

During the development of this WP a separate document has been created for each of the chapters of the Industrial Networks and Fieldbuses Module (list of chapters in WP1.1).

In each of these documents, section 2 introduces the chapter; sections 3, 4, 5 and 6 details the Lecture, Laboratory, Seminar and Mini-project of the chapter; section 7 lists the bibliography and the references.

This deliverable, in particular, presents the Introduction chapter.

2 Lecture

Foundations of industrial networks: an historical perspective

Problem statement

This lecture addresses the problem of computer communication within industrial systems focusing on the respective communication requirements and on the network technologies that were developed along the years to cater for such requirements. This historical perspective also covers the evolution of the architecture of such systems to the Computer-Integrated Manufacturing (CIM) architecture, leading to the definition of multiple levels with different communication requirements, thus supported by different network technologies.

The lecture then addresses the problem of defining and characterizing the specific network technologies developed for the industrial context, namely the fieldbuses and, more recently, Real-Time Ethernet, and covers the specific methods and techniques used in the layers of their protocol stacks.

- *Goal*

Allow the students to understand why industrial system evolved into a CIM architecture, as well as the characteristics of such architecture and the information flows therein. From this point, the students will learn the networking technologies that were specifically developed for that architecture and their main features, particularly the most relevant techniques used in their protocol layers.

- *Contextualization*

The global context of the topic addressed in this lecture is the CIM architecture and how it appeared. From this context we will focus on the lower levels of that architecture, typically Process/Machine level and Cell level, within which we can find the most stringent time requirements. This sets the context to explain the network technologies that were specifically developed for the industrial domain and used at those levels.

- *Motivation*

In general, industry can be divided in two groups, process and manufacturing. The former is concerned with continuous, discontinuous or batch processes, which have very large material flows and often have stringent safety requirements, e.g. power generation, cement kilns, petrochemical production. The latter type, manufacturing industry, is concerned with the production of discrete objects. For these industries, achieving maximum quantity throughput of produced goods is, normally, very important.

Around the 1960s and 1970s, an increasing economic pressure arising from social and environmental legislation, forced both process and manufacturing industries to improve production monitoring and quality control keeping costs low. This pressure pushed the replacement of manual operation of processes and machines by automatic equipment resulting in highly automated complex plants to guarantee process and products quality whilst increasing production volumes and reducing production costs. For example, just-in-time production, part of the total quality concept that became very popular in the 1980s, demanded for a tight monitoring and control of the materials flows within the factory plant only possible with a distributed automated system.

The use of automatic equipment in industry has a long history. However, its generalized use started in the second half of the 20th century, only, with the large scale availability of low cost semiconductor devices which enabled programmability and communication capabilities to be integrated and applied to the automation of industrial processes (Jordan,1995).

This evolution in industrial systems has not been identical in process and in manufacturing industries. The former has always been larger in plant area and always used pieces of equipment spread over the plant, interconnected in some way, to control and monitor the respective process. In this sort of industry, signal communication was always an important issue. It is interesting to note that the first international standard in the process field published by the International Electrotechnical Commission (IEC) Technical Committee TC-65 in 1971, the IEC 381-1, concerned signal communication. More precisely, it established the well-known 4-20 mA analogue signaling over twisted pairs of copper wire, in use since the mid-1950s, as an international standard.

Up to the 1960s, the typical system architecture used in process plants was centralized with the respective components being interconnected in a star topology (Fig. 1). A mainframe computer, installed in a central control room, performed all the supervisory and control functions and received and sent information from and to the plant, where the sensors and actuators, also known as field devices, were located, using point-to-point interconnections.

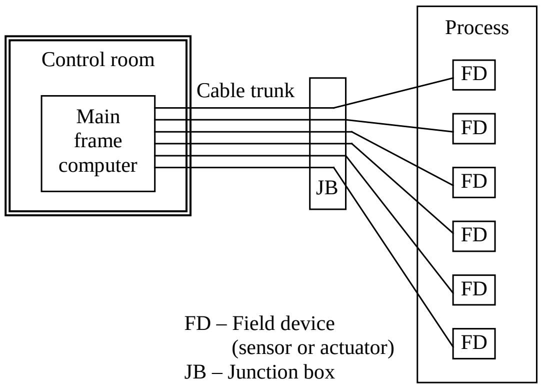


Figure 1. Typical star topology used in process control systems architecture until the 1960s.

Briefly, the disadvantages associated with such a centralized architecture, even if the mainframe computer is replaced by a set of workstations, are:

1) Complex and cumbersome wiring to interconnect field devices to the mainframe computer or workstations in the control room, being very large, difficult to install and reconfigure, and, particularly, very expensive;
2) The mainframe computers or workstations required to perform sophisticated control functions were also very expensive;
3) The mainframe computers or workstations constituted critical fault points where the existence of a fault could jeopardize the functioning of the whole system or in part;

One of the first steps that were adopted to reduce these disadvantages was to divide the supervisory and control functions among several controllers, each connected to a subset of the field devices and still using point-to-point interconnections. All the controllers were, in turn, connected to a central management information system. This was known as a hierarchical architecture (Fig. 2) and was introduced during the early 1970s. The processing power of the system would now be spread over several devices increasing tolerance to operating faults as well as maintainability.

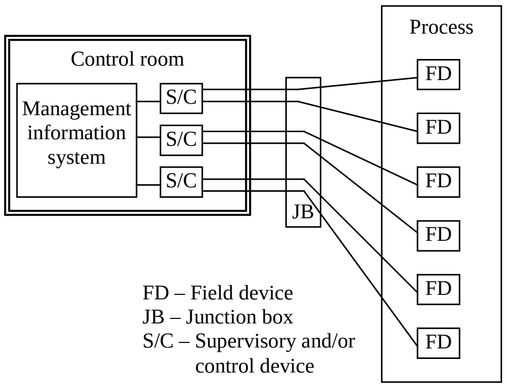


Figure 2. Hierarchical architecture typical in process control systems in the early 1970s.

Distributed architectures for process control appeared in the mid-1970s fostered by the advances in integrated circuits technology that improved controllers performance and functionality at lower costs. The controllers were interconnected to each other by means of a serial digital network and could be placed physically closer to the field devices reducing the length and complexity of the cabling system. However, each controller had its own sensors and actuators still connected to it through point-to-point interconnections (Fig. 3). The first distributed system of this kind for process control was introduced in 1975 by Honeywell, the TDC 2000 system.

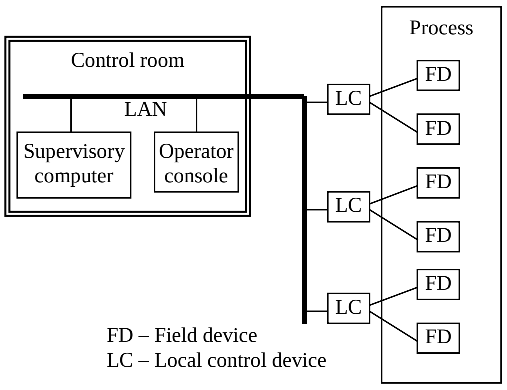


Figure 3. Distributed architecture of process control systems that appeared in the mid-1970s.

It did not take long until the field devices, spread over the plant, could also be connected by means of a serial digital network further simplifying the cabling system and improving overall system modularity and ease of maintenance (Fig. 4). In these systems, which appeared in the early 1980s, such interconnecting network became known as fieldbus as it provides communication among field devices and, in general, uses a bus as physical medium to carry the flow of information (Pimentel, 1990). This architecture was made possible by further advances in semiconductor technology that allowed incorporating more functionality into common field devices which became known as intelligent or smart sensors or actuators. Such added functionality included communications capability to allow direct connection to the fieldbus, automatic calibration and correction of offsets, drifts and other non-linear characteristics, processing capability to execute local control functions and condition monitoring for fault diagnosis.

Along this brief historical note, the emphasis put on the cabling system was deliberate because its simplification was of major importance in the evolution of process control systems. Note that such simplification brought along not only operational gains, coming from an increased system modularity and ease of maintenance, but also direct economic gains. According to Jordan (1995) several industrial studies showed savings greater than 25% of total system cost by adopting serial digital bus techniques instead of point-to-point interconnections.

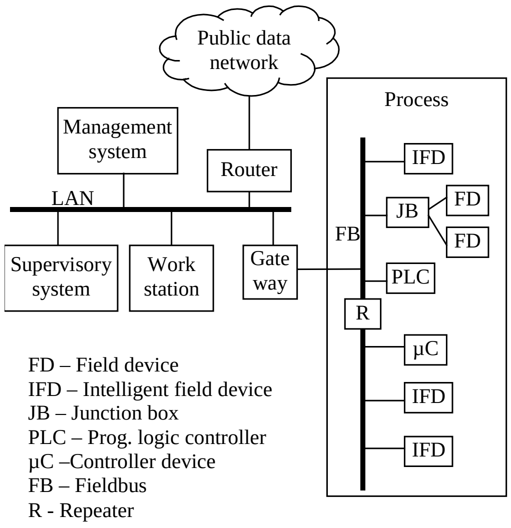


Figure 4. Fully distributed architecture of modern industrial automation systems.

But what happened with manufacturing industry? It was also subject to the same economic pressures towards lower production costs as process industry. However, it started from a different configuration. Manufacturing plants, or factories, were normally composed by several isolated production islands, also known as cells. One first evolution step was the use of numerically controlled machine tools to achieve maximum quantity throughput within each cell. However, grossly between the 1950s and 1970s, automation within factories was limited to the cell boundaries.

Only in the mid-1970s the need to interconnect cells came up as an important issue to improve factory wide production monitoring and control (Fig. 5). Hence, similarly to process industry, manufacturing automation started to move towards a distributed architecture where at least three different levels could be identified (Pleinevaux and Decotignie, 1988): level 2 or factory level, interconnecting the different areas within the factory, such as management, product development, maintenance, etc, level 1 or cell level, interconnecting automation equipment at the shop floor known as reflex automata, e.g., robot controllers, Programmable Logic Controllers (PLCs), Computerized Numerical Controls (CNCs), etc; and level 0 or sensor/actuator level linking sensors and actuators to the reflex automata. As in process industry, a cabling problem arized from the need to interconnect all this equipment at all layers. Solutions based on serial networks began to be introduced firstly in level 2 using general-purpose Local Area Networks (LANs). However, both levels 1 and 0 had communication requirements which could not be met by general-purpose LANs, particularly the need for frequent exchanges of small sized data. Thus, special-purpose networks, later known as fieldbuses, were developed firstly for level 1 and finally for level 0, too.

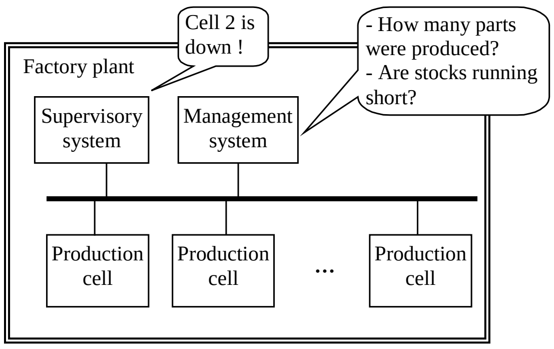


Figure 5. Distributed architectures appeared in manufacturing to interconnect plant cells.

The different levels of factory communication systems were hierarchically organized in what became known as the Computer Integrated Manufacturing (CIM) architecture (Pimentel, 1990), depicted in Fig. 6. This architecture was derived from the Manufacturing Automation Protocol (MAP) project, undertaken by a group of north-american companies lead by General Motors and started in 1980. The main purpose was to specify an open standard for the factory communication system that would allow interoperability between components of different vendors. The MAP protocol was thus based on the OSI reference model. Similar efforts to develop standard communication systems for factory automation followed in Europe, during the 1980s, with, for example, the creation of FIP in France, P-NET in Denmark, PROFIBUS in Germany (Thomesse, 1998).

Therefore, although process and manufacturing automation systems have traditionally developed separately, their differences in terms of the technical solutions nearly disappeared. In fact, both now use similar electronic systems for closed-loop control, for operator machine interfaces and for networking.

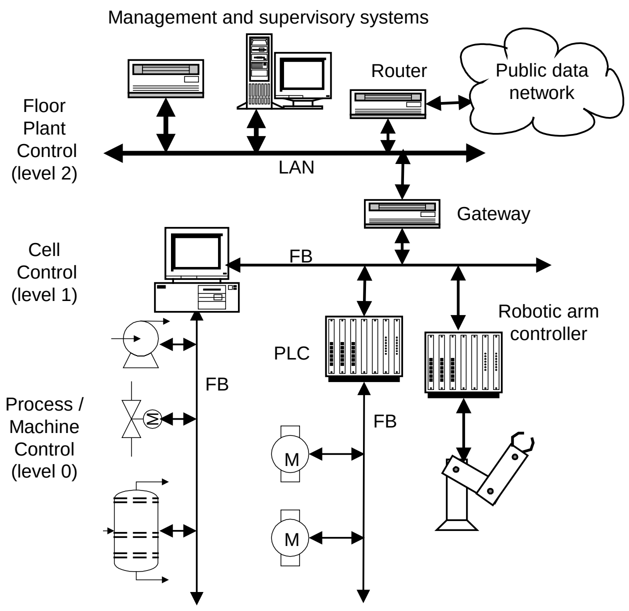

Management and supervisory systems
Router
Public data network
Floor Plant Control (level 2)
LAN
Gateway
Cell Control (level 1)
FB
Robotic arm controller
PLC
FB
Process / Machine Control (level 0)
FB
M
M

Figure 6. The Computer Integrated Manufacturing (CIM) architecture.

In particular, fieldbuses became the pivot element in industrial automation systems, interconnecting practically all instrumentation and control devices used in advanced industrial plants. Besides, fieldbuses have become pervasive in many areas of distributed real-time systems such as avionics, data acquisition and automotive systems. Among the potential benefits brought forward by fieldbuses are a system cost reduction due to cabling simplification and consequently, installation and maintenance, as well as improvements in system reliability, fault tolerance, modularity and reconfigurability and, last but not the least, in overall system performance.

More recently, the Ethernet technology made its way into the plant floor, establishing itself as an alternative to traditional fieldbuses given its high bandwidth and thus very small transmission latencies, openness, wide availability and low cost. Furthermore, it is a mature and well established technology with millions of nodes installed worldwide, being easy to deploy, manage and maintain.

However, Ethernet as it appears in general purpose LANs is not considered suitable to meet stringent real-time requirements. Thus, several modifications, extensions or limitations, were introduced to support time-constrained communication over Ethernet, giving rise to the so called Real-Time Ethernet (RTE) protocols, e.g., PROFINET, Ethernet POWERLINK, EtherCAT, TTEthernet (Decotignie, 2005). Nevertheless, from an architectural point of view, RTE protocols can be viewed as fieldbuses, with similar logical organization of the communication stack but with enhanced performance.

After the MAP project in the early 1980s it became commonly accepted that the implementation of the OSI seven layer capability at the device level would be too complex, too slow and too expensive. Therefore, fieldbuses were tailored according to a collapsed OSI model with three layers, in which a specialized Application Layer (AL) accesses directly the packet transfer services of the Data Link Layer (DLL), supported on a corresponding Physical Layer (PHY).

Starting from the PHY, the most important aspects that are addressed therein are:

- the interconnection topology;
- the physical medium used, and the possibility to use different media;
- the coding of the digital information into physical symbols (eventually including error detection and correction features);
- the transmission rate of physical symbols;
- the maximum bus length;
- the maximum number of nodes that can be connected to the bus;
- the possibility to feed power to the nodes through the bus;
- the immunity to EMI (Electro-Magnetic Interference);
- the possibility to limit the maximum level of energy supplied to any device in order to achieve intrinsic safety.

Concerning topology, as the name says, most fieldbuses provide a bus. Conversely, RTE protocols typically use switched stars or trees. With respect to the physical media, two are typically used, electric copper cabling and optical fibers. Nevertheless, there is a growing interest in using wireless networks in industrial systems, particularly for interconnecting mobile or rotating equipment, such as mobile robots, or to interconnect field devices spread over large distances or in difficult to access areas (see WirelessHART or ISA100).

Electric copper cabling is currently used mostly with voltage mode with a wide range of options for transmission rate. This choice typically determines the maximum bus length due to the propagation delay of physical symbols (Fig. 7). The higher the transmission rate, the lower the transmission delay should be and the shorter the maximum bus length. Thus, lower speeds are normally used in process control where time constants are large and bus lengths must be large, too. In manufacturing automation, higher speeds are normally required, for example, in closed loop control of the motion of robot manipulators. Another typical limitation is the number of nodes that can be connected to the fieldbus. In fact, each connected node adds an extra capacitance to the bus line which tends to increase the propagation delay of physical symbols.

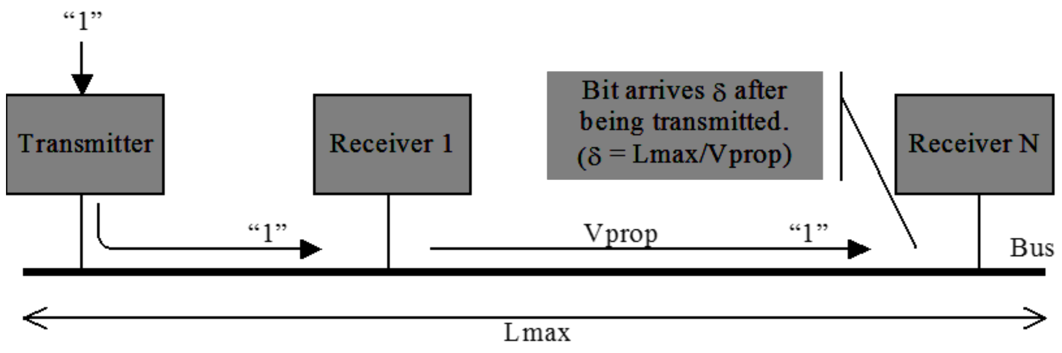


Figure 7. Propagation delay on a bus

For example, some fieldbuses use the RS485 standard to transmit signals over a twisted pair with differential voltage. This is the case of PROFIBUS which allows for transmission rates from 9.6 kbit/s to 1.5 Mbit/s depending on the required bus length which is bounded to 1200m or 4800m without or with repeaters, respectively. Transmissions rates of up to 12 Mbit/s are also supported with appropriate communication drivers. The maximum number of nodes in PROFIBUS/FMS is 127 of which 32 can be bus masters. Other fieldbuses, such as WorldFIP, use a twisted pair of wires to transmit synchronously with Manchester coding at one of three possible transmission rates, 31.25 kbit/s, 1 Mbit/s and 2.5 Mbit/s. WorldFIP allows for a maximum length of 2000m at 1 Mbit/s with 256 nodes. The Controller Area Network (CAN) also uses a two-wire differential bus with a common return line that supports up to 1 Mbit/s on a maximum bus length of 40m. The bus length can be increased to 1km reducing the transmission rate to 50kbit/s according to a nearly inverse function.

On the other hand, optical fibers have the major advantage of electromagnetic noise immunity with lower weight. Many of the existing fieldbuses allow the use of optical fibres. WorldFIP is one example of this which may operate over optical fibre with one of three possible transmission rates: 1, 2.5 and 5 Mbit/s.

In what concerns the Data link layer, the most relevant issues dealt there in are:

- Addressing the nodes involved in packet transfer;
- Medium Access Control (MAC);
- Logical Link Control (LLC).

Concerning addressing, there are, essentially, two different ways of identifying the destination of a packet exchange: directly identifying the destination node or identifying the packet that is being transmitted. The former is named direct addressing and is the most common type, used in PROFIBUS and P-Net. The destination node can be identified either by a physical address, known as the MAC address, or by a logical address, the network address. MAC addresses are more common in fieldbuses but cases exist of complex multi-segment topologies where network addresses are used. Indirect addressing is also rather common among fieldbuses and it inherently supports anonymous communication in which a sent packet is made available to all, or a group of, nodes that can individually decide on whether to receive the packet or not, without knowledge of the sender. There two types of indirect addressing. One is source addressing (Fig. 8), used by WorldFIP and CAN, in which what is identified is the source of the packet, namely the data entity being transmitted. The other type is time-based (Fig. 8), used by TTP, in which the packet is identified by the time at which it is transmitted.

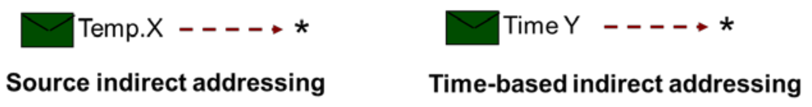


Figure 8. Source and time-based indirect addressing (forms of anonymous communication).

The MAC sub-layer controls the access to the bus, or generally a shared communication medium, to avoid collisions that would, otherwise, corrupt the information to be transmitted. MAC protocols have a direct influence in the timely behavior of a fieldbus. For proper real-time operation, it is mandatory that all nodes are given access to the bus within bounded time windows that allow meeting the communication time constraints.

MAC protocols can generally be divided in two main groups, controlled and uncontrolled access protocols (Thomesse, 1998). In the former group, the order by which nodes access the bus is determined by explicit, e.g. token, or implicit, e.g. time, external control (Fig. 9). Examples of protocols of this kind include Master-Slave, Token-Passing and Time-Division Multiple Access (TDMA).

In Master-Slave protocols, a special node called master cyclically polls and selects the remaining nodes called slaves (Fig.9). This approach is simple to implement and the resulting behavior is very predictable. Besides, in most cases nodes respond immediately to the master requests so that the timely behavior of the communication system depends mainly on the timeliness of the master polls. These systems are normally well suited to handle periodic traffic and require master redundancy to avoid the associated single point of failure. Well know examples include MIL-STD1553B, WorldFIP, AS-Interface, EtherCAT and even PROFIBUS between each master and the associated slaves.

Token-passing protocols use a special short packet, called token, to circulate an authorization to transmit among the nodes in the system (Fig. 9). Only the node currently holding the token is allowed to transmit. Broadcast buses using this access control protocol are commonly known as token-buses. Timely behavior is achieved limiting the time that a node can hold the token continuously before sending it to the next node, generically called timed-token protocol. If a node receives the token and has nothing to transmit forwards it immediately to the next node in the list. This allows bandwidth reclaiming, leading to an efficient bandwidth usage but the transmission instants granted to nodes are rather irregular due to the irregular token arrivals. Moreover, not that the token itself is a single point of failure that needs to be recovered in case of loss. PROFIBUS uses this technique to control the access of multiple master nodes to the bus.

TDMA is based on partitioning the global time in time windows, slots, during which only one node can access the bus (Fig. 9). It normally relies on a clock synchronization method which must be as precise as possible. Else large guarding windows are needed between slots that degrade the efficiency of bandwidth usage. Since slots arrive at precise periodic intervals, this method is very suited to support periodic communication. Moreover, since the slots are disjoint in time, the nodes transmissions are isolated from each other, which grants composability in the time domain. Protocols using this MAC method include the Time-Triggered Protocol (TTP), FlexRay, TTEthernet, PROFINET-IRT.

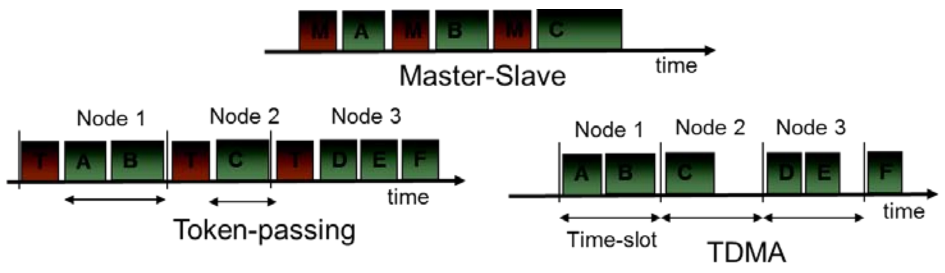


Figure 9. Common controlled access MAC protocols

In what concerns uncontrolled access protocols, there is no external control signal, either explicit, e.g. token, or implicit, e.g. global time, to tell a node when to transmit. The decision on whether to transmit or not is performed based on the medium status and on local information, only, thus fully decentralized. Currently, most protocols in this class belong to the persistent Carrier Sense Multiple Access (CSMA) family according to which a node wishing to transmit listens to the medium and starts transmission only upon silence detection (carrier sense feature). If the medium is busy, prospective senders wait until the medium becomes free (persistence feature) and then attempt transmission according to some rule.

In general, access collisions may still occur since it is possible that several nodes detect silence and start transmitting simultaneously. Several CSMA protocols exist that differ on what is done upon a collision detection or on what is done to prevent collisions at all.

The expression CSMA/CA, where CA stands for collision avoidance, is actually used for two different variants. One is fully asynchronous with respect to medium access and it just reduces collisions stochastically, thus not being completely collision-free. We refer to this variant as CSMA/CA-async (Fig. 10) and it is typically used in general purpose wireless LANs such as WiFi. The collision avoidance part is based on *p-persistence*, i.e., once a prospective sender detects the medium busy, it waits until the medium becomes free and transmits immediately with probability *p*. In practice this means that when the medium becomes free each prospective sender computes a random interval compatible with *p*, after which is transmits. It is this random interval that reduces collisions. Nevertheless, chained collisions can also occur and, whenever they do, *p* is halved until a certain limit.

The other variant presents a deterministic behavior and we refer to it as CSMA/CA-sync (Fig. 10). In this case the nodes are synchronized to a cyclic framework within which they are assigned a given time offset represented in mini-slots. If mini-slots are uniquely assigned to nodes, these simply need to count mini-slots and safely transmit when their mini-slot arrives. Note, however, that mini-slot counting is suspended whenever a transmission is detected. This mechanism deterministically prevents collisions and simultaneously assigns a medium access priority to each node, corresponding to the assigned mini-slot. The earlier the mini-slot, the higher the priority. A protocol that uses this, also called, mini-slotting MAC is FlexRay in its dynamic segment. P-Net also used a similar mechanism but with a different synchronization scheme.

Another CSMA protocol that is rather relevant is a *one-persistent* variant based on *dominance* and *bit-wise arbitration*, frequently referred to as CSMA/BA (Fig. 10). The dominance property means that when two or more senders collide on the medium while sending one digital symbol, the outcome of the collision is deterministic and always equal to the *dominant* symbol. The other symbol is called *recessive*. According to this method, all senders start their transmission with a special identifier, which is uniquely assigned to variables to be transmitted over the network and to the nodes that produce them. When several senders start transmitting at the same time, i.e., a collision situation, they will simultaneously transmit and sense the medium symbol by symbol (typically bit-wise). When and if a sender encounters a mismatch, meaning it was transmitting a recessive symbol while another sender transmitted a dominant one, it backs off, waits until the end of the current transmission and retries immediately after. As long as identifiers are assigned uniquely to nodes, this method sorts out collisions in a rather efficient way, without any extra waiting time, and in a priority order determined by the specific identifiers. If the dominant digital symbol is the 0, then the lower the identifier, the higher the priority. This is the MAC used by CAN.

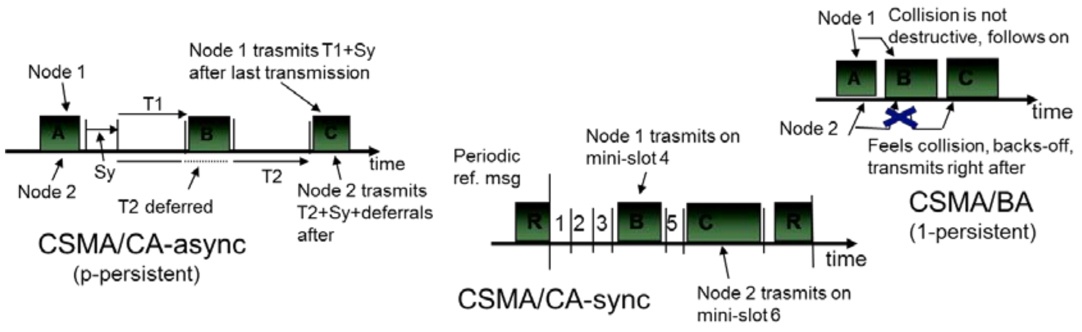

Node 1
Node 1 trasmits T1+Sy after last transmission
T1
A
B
C
time
Node 2
Sy
T2 deferred
T2
Node 2 trasmits T2+Sy+deferrals after
CSMA/CA-async
(p-persistent)
Periodic ref. msg
Node 1 trasmits on mini-slot 4
R
1
2
3
B
5
C
R
time
CSMA/CA-sync
Node 2 trasmits on mini-slot 6
Node 1
Collision is not destructive, follows on
A
B
C
time
Node 2
Feels collision, backs-off, transmits right after
CSMA/BA
(1-persistent)

Figure 10. Typical uncontrolled access MAC protocols.

In what concerns the LLC sub-layer, it basically supports reliable packet transfers (Fig. 11). Although in many fieldbuses the LLC is not clearly separated from the MAC sub-layer, most of its typical services can be identified. These services include the framing of the data in packets, error detection and correction mechanisms, establishment of data link connections between communicating nodes and communication acknowledgement.

Most fieldbuses use connectionless packet transfers with different acknowledge variants. Typically, periodic transfers are unacknowledged, which implies that they are not retransmitted in case of a packet loss. The rationale is that the missing value will be recovered in the following periodic refresh. However, aperiodic transfers normally refer to events and their occurrences are unknown. Thus, these communication services are typically protected by immediate acknowledgement, which triggers retransmissions in case of lost packets, increasing the reliability of each individual data transfer. However, since the packet contents frequently have tight temporal validity, only a few retransmission attempts are considered, e.g., 2 or 3 at most, after which the transmission is discarded if still unsuccessful. This is a consequence of the inherent conflict between timeliness and reliability based on temporal redundancy.

Did it get there?
Node A
Node B

Figure 11. The LLC sub-layer deals with reliable packet transfer.

Concerning packet framing, LLC sub-layers of fieldbuses are particularly efficient, adding just the minimum number of extra control information to assure proper packet transfer. This assures the lowest possible communication overhead and reasonable bandwidth efficiency even for rather short data. Typical SDUs of these protocols, at this DLL level, go from 8B in CAN, to 2B in MIL STD1553B to 128 or 256B in WorldFIP real-time and non-real-time services, or 246 in PROFIBUS. On a different stand are the RTE protocols, which use Ethernet frames that can go up to close to 1500B of data depending on the specific protocol.

Finally, on top of fieldbus protocol stacks is an Application Layer with specialized services for accessing real-time and non-real-time communication from within application programs. The AL determines the programming abstractions that are made available to applications, not only at the data representation level but also at the interaction level.

With respect to the first issue, the level of abstraction is relatively low. In the best case, several ALs provide a library of classes for object-oriented programming facilitating the creation of process objects, service objects, management objects, etc, e.g., CANopen and WorldFIP. These objects are typically represented by Electronic Data Sheets (EDS) in CANopen or Virtual Fieldbus Devices (VFD) in PROFIBUS that describe the functions and properties of field devices connected to the fieldbus and managed through an object directory.

On the interaction level we find two main models, Client-Server (CS) and Producer-Consumer (PC), both with several variants (Fig. 12). In the CS model, the relevant process information is held by servers and requested explicitly by clients. In this sense, sensors are typically servers and controllers are clients. However, every logical entity can be simultaneously client in one given relationship and server in another one. This model is inherently related with one-to-one (unicast) communication and implicitly imposes delays in the middle of clients' execution while waiting for servers answers. PROFIBUS is an example of a fieldbus that uses CS interactions. There are then variants that allow clients to do asynchronous server invocations, thus not waiting for the answers, as well as variants in which the servers reply in broadcast so that several clients can receive it the same time.

A near opposite model is the Producer-Consumer (PC). In this model, who has the relevant process data is the producer that broadcasts it autonomously. The consumers that are interested in the data retrieve it from the network. Thus, it is inherently an anonymous communication model in which consumers are automatically updated. This model is used by CANopen that also supports CS interactions in non-real-time objects. There are also several variants of the PC model that are rather relevant in the industrial context. For example, the Producer-Distributor-Consumer (PDC) model is used by WorldFIP and consists in a PC model in which the transmissions of multiple producers are coordinated by a central distributor. In fact, this is a PC model implemented on top of a master-slave MAC network.

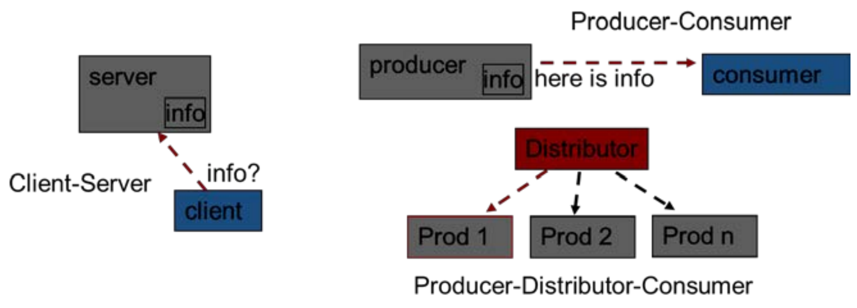


Figure 12. Fieldbus application layer interaction models.

One particular aspect that results from the interaction model and has a significant impact on the efficiency of bandwidth usage is the support for direct device-to-device communication, sometimes referred as D2D, understanding devices as field components of the same kind. While PC models naturally support D2D, e.g., CANopen and WorldFIP have always provided such support, CS models typically do not, since they need the information to be retrieved by a client that is frequently at a higher abstraction level, e.g., a controller, and then delivered to the other device. This was the case of PROFIBUS that requested slave-to-slave communication to be always relayed by one or more masters. This limitation was lifted in one of PROFIBUS latest releases, so that slaves could be polled by masters and then communicate directly with other slaves in a D2D fashion.

- - Bibliography

- ed. Richard Zurawski; The industrial communication technology handbook. ISBN: 0-8493-3077-7
- Thomesse J.P. (1998). The Fieldbuses. Annual Reviews in Control, 22: 35-45.
- J.-D. Decotignie (2005), "Ethernet-based real-time and industrial communications" Proc. of the IEEE, vol. 93, no. 6, pp. 1102–1117.
- Jordan, J. R. (1995). *Serial Networked Field Instrumentation.* John Wiley & Sons, England.

- Concepts
  - Historical evolution of industrial systems architecture
  - Real-time communication
  - Fieldbus
  - Medium Access Control (MAC) methods
  - Controlled access MAC protocols family
  - Uncontrolled access MAC protocols family
  - Logical Link Control (LLC) methods
  - Electronic Data Sheet
  - Client-Server interaction model

- Producer-Consumer interaction model
- Device-to-device communication

- *Examples*

- The CIM architecture
- Machine/process real-time communication requirements
- WorldFIP, PROFIBUS, P-Net, CANopen
- RTE protocols
- Master-slave, token-passing, TDMA MAC protocols
- CSMA/CA-async, CSMA/CA-sync and CSMA/BA MAC protocols
- Positive Acknowledge and Retry
- Automatic Repeat Request
- Unacknowledged communication
- CANopen process data objects (PDO) and service data objects (SDO)
- PROFIBUS-DP v2

- Control Questions

- What is the difference between process and manufacturing industries?
- Why did industrial architectures evolve to fully integrated systems?
- What is a fieldbus and which is its distinctive feature among data networks?
- Which topologies are more common in fieldbus networks?
- Give some examples of fieldbus protocols and characterize them according to their main features in each of the layers in the protocol stack?
- Is there a relationship between bus length and bit rate? Justify your answer.
- Why are RTE protocols becoming popular as replacement of traditional fieldbuses?
- Give examples of fieldbus MAC protocols and discuss their main properties.
- Compare master-slave and token-passing MAC methods in the context of fieldbuses.
- Why are retransmissions limited to a rather small number in fieldbuses, when used?
- Given examples of addressing schemes used in fieldbuses.
- What kind of interaction models are common in the application layers of fieldbuses?
- Discuss the suitability of the CS interaction model for dissemination of real-time data.

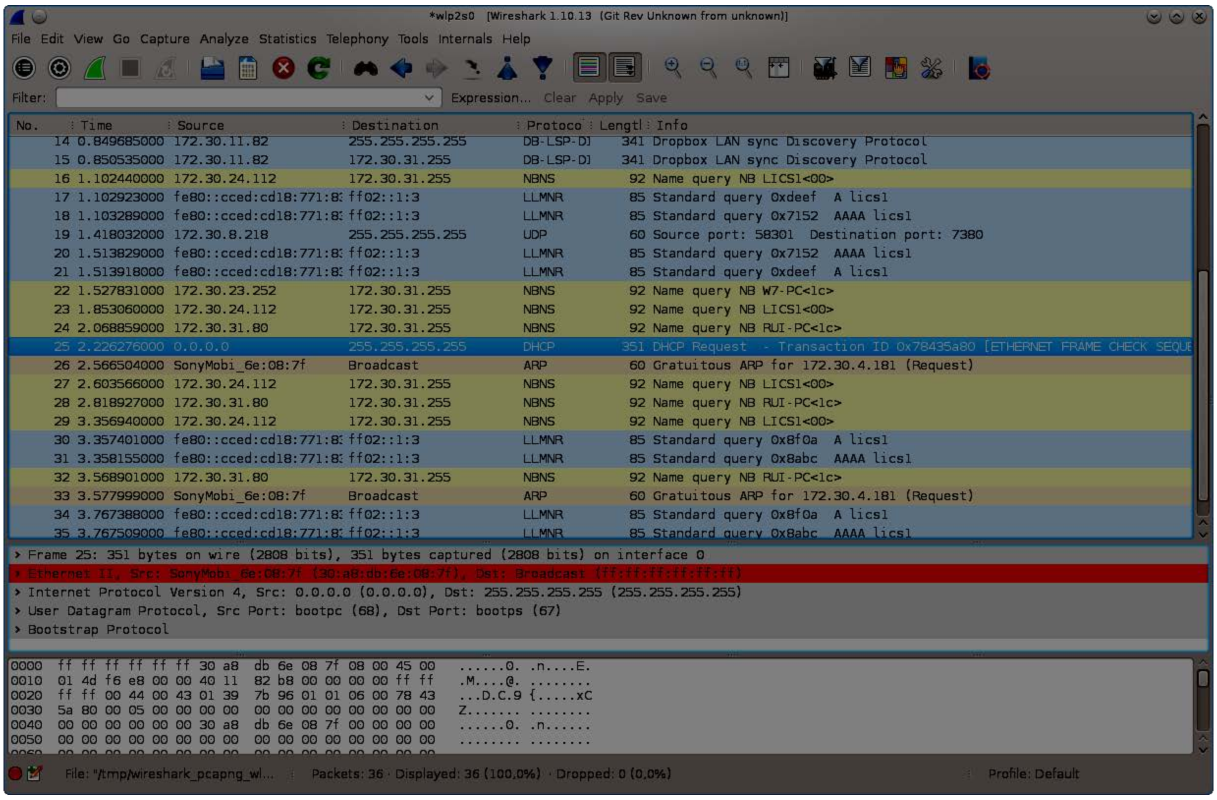
- Recommended Further Reading

- eds. Frithjof Klasen, Volker Oestreich, Michael Volz; Industrial communication. ISBN: 978-3-8007-3358-3
- Josef Weigmann and Gerhard Kilian; Decentralization with PROFIBUS-DP. ISBN: 3-89578-144-4
- Olaf Pfeiffer, Andrew Ayre, Christian Keydel; Embedded networking with CAN and CANopen. ISBN: 978-0-9765116-2-5
- Pimentel J. R. (1990). *Communication Networks for Manufacturing*. Prentice-Hall, Englewood-Cliffs, NJ.
- Pleinevaux, P. and J.D. Decotignie (1988). Time Critical Communication Networks: Field Buses. *IEEE Network*, **2(3)**: 55-63.

3 Lab – Week 1

Wireshark is a program that is capable of capturing every packet that arrives and leaves through a specific communication device, and analyse and identify the data contained in those packets. These packets are then presented in a list that can be sorted by time, sender, receiver, and many other properties related to the protocol itself.
Notice however that a lot of personal data may be transmitted over the network, so typically not all users of a specific computer should be given permission to snoop on all the data traffic of that computer. For this reason, the wireshark program will only be able to capture every data packet when it is run under administrator privileges.
Wireshark is an open-source program that will run under Linux or Windows. Download the program from www.wireshark.org, and install it on your computer.
Run it using administrator privileges.
Select the network card you want to snoop on, and collect some traffic for a few seconds. Notice how many packets are captured. Try to see if you can identify any of the other protocols that were captured.
Note that your computer receives many packets even when it is seemingly not doing any network activity. This is because many upper layer protocols typically require the services of lower layer protocols to work, and these lower layer protocols sometimes use broadcast based transmissions.
For example, the basic ARP protocol (Address Resolution Protocol) is used to map IP addresses onto Ethernet addresses. This is because the packets, when sent over an Ethernet network, have to use an Ethernet destination address, and not just the IP address of that same computer (each computer interface will have both a unique IP address, as well as a unique Ethernet address). The mapping between these two addresses is done by the ARP protocol. In this protocol, a computer wishing to send a packet to a remote host will basically broadcast a question to every computer on the local network specifying the desired IP address, and the mentioned computer will reply therefore giving his Ethernet address. Each computer will keep a local table with the most recently obtained mappings between the Ethernet and the IP address, so you will not see too many of these messages.
To see this table, open a command window in your computer and type the command “arp -a”. This will show you the current mappings between IP and Ethernet (so-called MAC) addresses.
In order to observe some related activity in Wireshark, issue a “ping” command to another computer in your network. Observe the ARP table again and see that the IP and MAC addresses of that computer now appear in there. Now explicitly delete this entry in the ARP table using the command “arp -d ip_address”. Start a Wireshark capture and issue another “ping” command

to the same computer. Stop the capture. You should now observe all the frame involved in the ARP protocol. Spend some time analysing their structure by inspecting the respective fields.


*wlp2s0 [Wireshark 1.10.13 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.   Time        Source                 Destination      Protoco  Lengtl Info
 14 0.849685000 172.30.11.82            255.255.255.255  DB-LSP-DI  341 Dropbox LAN sync Discovery Protocol
 15 0.850535000 172.30.11.82            172.30.31.255    DB-LSP-DI  341 Dropbox LAN sync Discovery Protocol
 16 1.102440000 172.30.24.112           172.30.31.255    NBNS        92 Name query NB LICS1<00>
 17 1.102923000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0xdeef  A licsl
 18 1.103289000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0x7152  AAAA licsl
 19 1.418032000 172.30.8.218            255.255.255.255  UDP         60 Source port: 58301  Destination port: 7380
 20 1.513829000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0x7152  AAAA licsl
 21 1.513918000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0xdeef  A licsl
 22 1.527831000 172.30.23.252           172.30.31.255    NBNS        92 Name query NB W7-PC<1c>
 23 1.853060000 172.30.24.112           172.30.31.255    NBNS        92 Name query NB LICS1<00>
 24 2.068859000 172.30.31.80            172.30.31.255    NBNS        92 Name query NB RUI-PC<1c>
 25 2.226276000 0.0.0.0                 255.255.255.255  DHCP       351 DHCP Request  - Transaction ID 0x78435a80 [ETHERNET FRAME CHECK SEQUE
 26 2.566504000 SonyMobi_6e:08:7f       Broadcast        ARP         60 Gratuitous ARP for 172.30.4.181 (Request)
 27 2.603566000 172.30.24.112           172.30.31.255    NBNS        92 Name query NB LICS1<00>
 28 2.818927000 172.30.31.80            172.30.31.255    NBNS        92 Name query NB RUI-PC<1c>
 29 3.356940000 172.30.24.112           172.30.31.255    NBNS        92 Name query NB LICS1<00>
 30 3.357401000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0x8f0a  A licsl
 31 3.358155000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0x8abc  AAAA licsl
 32 3.568901000 172.30.31.80            172.30.31.255    NBNS        92 Name query NB RUI-PC<1c>
 33 3.577999000 SonyMobi_6e:08:7f       Broadcast        ARP         60 Gratuitous ARP for 172.30.4.181 (Request)
 34 3.767388000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0x8f0a  A licsl
 35 3.767509000 fe80::cced:cd18:771:8: ff02::1:3        LLMNR       85 Standard query 0x8abc  AAAA licsl

> Frame 25: 351 bytes on wire (2808 bits), 351 bytes captured (2808 bits) on interface 0
> Ethernet II, Src: SonyMobi_6e:08:7f (30:a8:db:6e:08:7f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Internet Protocol Version 4, Src: 0.0.0.0 (0.0.0.0), Dst: 255.255.255.255 (255.255.255.255)
> User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
> Bootstrap Protocol

0000  ff ff ff ff ff ff 30 a8  db 6e 08 7f 08 00 45 00   ......0. .n....E.
0010  01 4d f6 e8 00 00 40 11  82 b8 00 00 00 00 ff ff   .M....@. ........
0020  ff ff 00 44 00 43 01 39  7b 96 01 01 06 00 78 43   ...D.C.9 {.....xC
0030  5a 80 00 05 00 00 00 00  00 00 00 00 00 00 00 00   Z....... ........
0040  00 00 00 00 00 00 30 a8  db 6e 08 7f 00 00 00 00   ......0. .n......
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........

File: "/tmp/wireshark_pcapng_wl...   Packets: 36 · Displayed: 36 (100,0%) · Dropped: 0 (0,0%)   Profile: Default
```

Then, start a wireshark capture and, during the capture, download a file using the ftp protocol (any URL starting with ftp://). The corresponding packets will be captured. When you list the packets notice that you can apply filters on the sender and receiver, as on many other parameters, to reduce the number of listed packets to only those you are really interested in.
Downloading a simple text file, you will be able to see the data contained in the file simply by analysing the data packets. This is because ftp does not encrypt the data that is transmitted. The same will happen if you use the smtp protocol for sending emails, or the pop3 protocol for reading emails. Emails are basically sent over the network as postcards that anybody can read, if they wish. No privacy exists when using these protocols! However, these protocols normally exchange one-to-one (unicast) messages that will be forwarded through the network links involved in the path between sender and receiver, only. Thus, most likely, you will not see any of such messages except those meant to, or generated by, your computer.
One important point to observe is the encapsulation of protocols associated to protocol stack. Pick any high level protocol frame, for example, one captured in the ftp transfer you did before. Observe that, at the highest level is it identified as an ftp frame (application layer). However, it is conveyed inside a TCP frame (transport layer), which in turn is encapsulated in an IP frame. This is then in the payload of an Ethernet frame (data linklayer) which is transmitted "on the wire" with an additional preamble and some other bits (physical layer).
Take some time to observe the structure of the frames in each protocol, particularly the respective header and its fields. Note that Wireshark already has the so-called "dissectors" for most of the common protocols, which allows the program to identify each field in the control information of each protocol.
Repeat this step for other protocol frames. In particular try identifying transactions based on TCP sessions and on UDP datagrams and observe carefully the control fields in the respective headers.

4 Lab – Week 2

At the lower layers of industrial systems we frequently find closed-loop control of processes which is carried out over the industrial networks. For such control to be effective, the respective communications must be timely. Thus, timing is very important.

In this session we observe the timings associated to communications and their impact on the control performance.

One basic way to observe communications timing is through Wireshark. This program timestamps all received packets at the lowest level of the protocol stack, thus very close to the actual reception instant. Open Wireshark, carry out a traffic capture for a few seconds and analyse the timestamps of the received frames.

A concept that is very relevant when it comes to observing network delays is that of Round-Trip Delay (RTD), which consists in sending a particular packet to a particular destination that automatically sends an answer back. The packet is timestamped right after transmissions and another timestamp is taken as soon as the answer arrives. The RTD is the difference between both timestamps. If we divide the RTD by two we have an estimate of the network delay between the two involved computers.

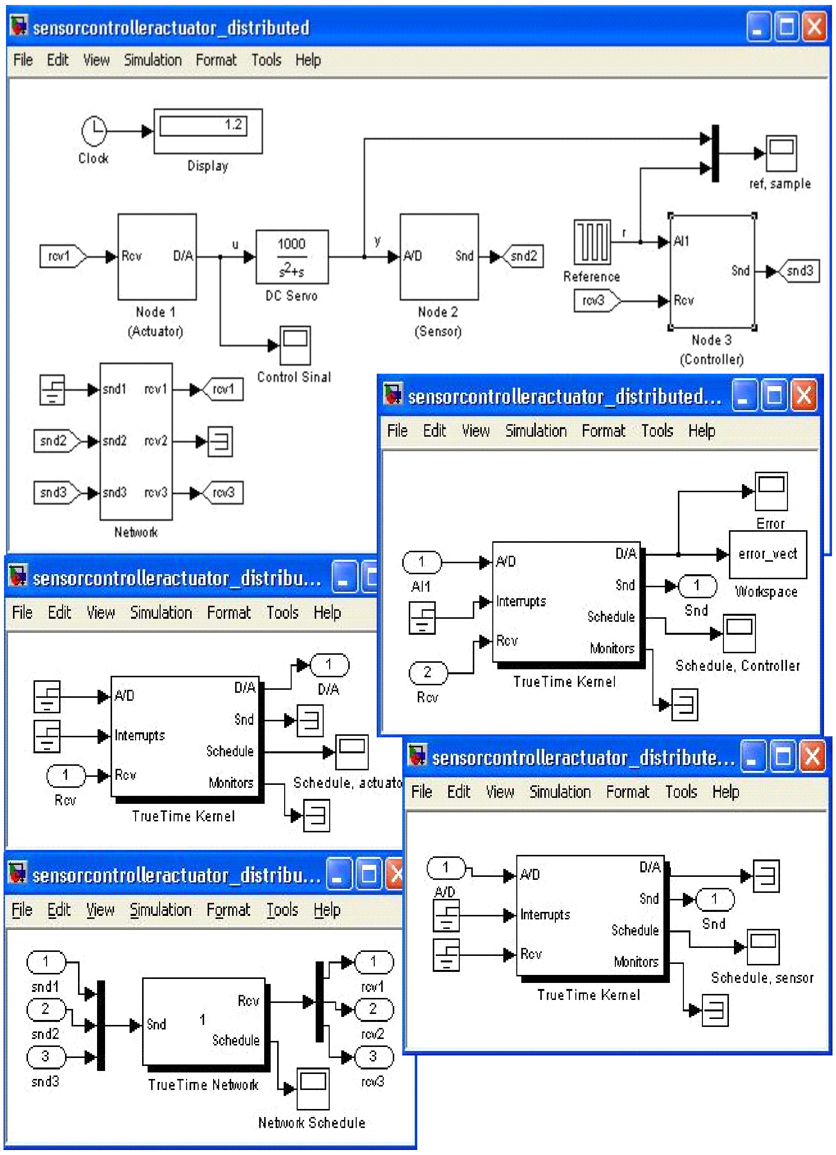
The IP protocol stack already provides direct support to measuring the RTD. It can be done automatically using ”ping” commands. For this purpose, open a command window in your computer and execute a series of “ping”commands addressed at different computers, one at a time. You can “ping” the local host (127.0.0.1), which just tests the proper functioning of the protocol stack and does not involved network communications, but then “ping” another computer in your network, another one outside your network but in your institution, and finally another “ping” to a computer that is far, for example, a portal from a well known Internet-based services provider (e.g., www.youtube.com). Observe the reported RTD and their variations.

From within Wireshark, start a capture. Then program the “ping” to generate at least 10 RTD measurements and issue that command to another computer in your network. Stop the capture and observe the timestamps of the involved ping frames. Compute their differences and observe the regularity of the “ping” generation as well as the accuracy of the respective RTD measurements.

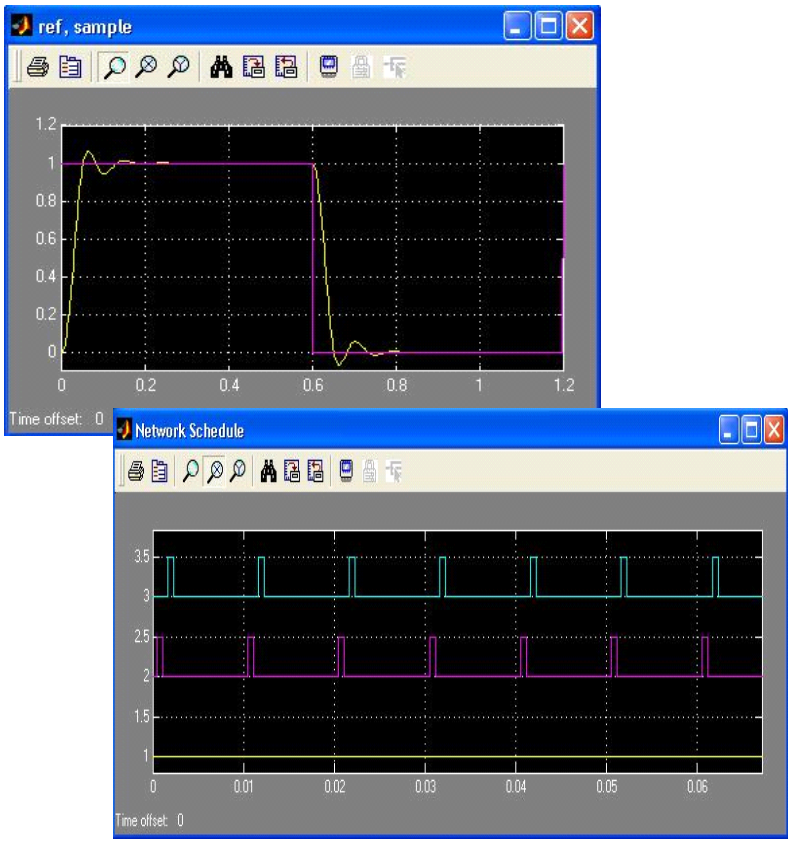
If you have Matlab/Simulink, you can do a further experiment to observe the impact of delays in the performance of closed-loop control, using the Truetime toolbox. You will need to install the toolbox. One of the standard demos that comes with Truetime is the model of a distributed control of a servo.

When opening up this model, note the plant as well as the sensor, actuator and controller nodes and an extra node that generates extra network traffic.

Open the



network box and you will be able to specify and configure the network protocol. Try different network configuration options. For each option, run the simulation and observe the plant output plots and well as the network traffic schedule.



In particular, use faster and slower network speeds, which reduces or increases, respectively, the duration of the network frames and thus, their associated delay. Observe the corresponding impact on the stability of the plant output.

5 Seminar – Week 1

Many protocols are commonly used over Internet. As a preparation for the first lab session, the students are asked to study some communication protocols used in a common office environment, especially when accessing services over the Internet. These protocols will later be identified during the lab session, so it is important to have an idea of what they do, and how they are structured.
For this 1st seminar, we ask each group to study one common protocol working over TCP or UDP, such as http (web), smtp (email), pop3 (email), ntp (time synchronization), ftp (file download), dns (name service), dhcp (ip addressing), etc. Additionally, one or two groups should focus on the layered structure of the TCP/IP protocols, and the nature of their addressing. These protocols will later be analysed during the first lab session.

6 Seminar – Week 2

This module will focus mainly on two fieldbuses very widely used in the industrial automation field – Modbus and CANopen. In this seminar we suggest that students make a brief overview of other protocols also used in this field. Example protocols include: Profibus-DP, PROFINET, ASI, EtherCAT, Ethernet/IP, SERCOS, FOUNDATION FIELDBUS, Interbus, LonWorks, etc...

7 Mini-project – Week 1

In this first week, no implementation work is expected to be done during the mini-project session. The project itself is presented, and the requirements are discussed in order to make sure these are completely understood. The students should be arranged in groups of 2 or at most 3. Each group is encouraged to discuss possible strategies of attacking the problem.
The mini-project itself will consist on automating a small manufacturing cell, consisting of two machines and several conveyors that transport the workpieces to/from these machines. This manufacturing cell is simulated using a java program which is supplied to the students. The plant simulator produces a graphical image representing its current state. Physical sensors and actuators are also simulated, and reading of sensors and writing to the actuators is achieved using the Modbus/TCP communication protocol. This communication protocol will the the subject of weeks 3 and 4. Until then work on the mini-project will focus on designing the algorithms to be implemented, the architecture of the controlling software, and analysing the system's functional and timing requirements.

As can be seen in the figure above, each conveyor (or machine/conveyor pair) has a unique name identified in the figure above in blue. The machines work together with the conveyors attached to the machines - we consider them as being a single unit, so the machines do not have an independent name.
There are two types of conveyors. T1, T2, T5, T8 and T9 are all linear conveyors. T3 and T7 are rotating conveyors. T4 and T5 are also linear conveyors, but since they are attached to a machine they will be explained later.
Linear conveyors can only move pieces in either the positive or negative direction. Notice that the x axis is inverted to the usual direction, so moving along the negative direction on the horizontal linear conveyors will move the work-pieces towards the right! Moving in the positive direction will move the pieces to the left.
Linear conveyors placed vertically work as is commonly expected: positive direction is up, and negative direction is down.
The movement in each direction is controlled by a single digital actuator, which means that linear conveyors have two actuators. If both of these actuators are activated simultaneously the conveyor will simply stop. This simulates the fact that an electrical based circuit implements an interlock that does not allow the conveyor to become damaged in these situations by simply deactivating both motors.
Linear conveyors also have a single sensor located at the center of the conveyor. This sensor will detect work-pieces when a work-piece reaches it.
The state of each sensor and actuator is represented by a very small 'light' in the top left corner of each conveyor. Light's with a grey background represent actuators, and lights with a black background represent the sensors.
Rotating conveyors also have a single sensor in the middle detecting a work-piece, and two actuators for displacing the work-piece. They are also able to rotate about their vertical axis in order to align in either the vertical or horizontal direction. This requires an extra two actuators (for rotating in the direct and inverse direction), as well as an extra two sensors to tell whether it has reached the horizontal or vertical position. Like the actuators for the displacement of the work-piece, if the actuators for rotating the conveyor are both set to 1 the conveyor simply stops rotating. However, if the rotation is not stopped once it reaches the horizontal or vertical position the conveyor becomes damaged and a red box is drawn around this conveyor. When

this occurs the conveyor will not be able to be repaired. You will need to re-start the simulator application.
The machine tools, besides the same sensor and actuators as linear conveyors, can also move their turret in the horizontal and vertical axis (4 actuators and 4 sensors), activate the tool on the work-piece on the conveyor, and change the tool. Each turret carries three tools, and a single sensor will tell you whether a tool is correctly aligned. No sensor will tell you which tool is currently aligned – you will need to count how many times the tool has changed. Every time the program starts, the first tool (ta) will be aligned and ready to use.
In summary, a machine/conveyor pair has 9 actuators, and 6 sensors.
All the actuators in the plant floor have a unique address, starting from 0. This address will be used by the Modbus protocol to identify which actuator to set or reset. Each sensor will also have a unique address, also starting off from 0. It can therefore be said that sensors and actuators use independent address spaces.
The address of the first sensor and first actuator of each conveyor (or machine/conveyor pair) can be determined by hovering your mouse over the respective conveyor. After a short time interval a message will appear. For example, for T5 you will see: "#5 T5 Conveyor (11:11, 17-18). This means that the sensors of this conveyor start at address 11, and finish at address 11 (only one sensor). The actuators start at address 17, and finish at address 18 (2 actuators).
Besides activating the actuators over the Modbus protocol, you can also set the actuators by using the menu that pops-up when you right-click over each conveyor. Using this menu you will also be able to place and remove work-pieces from the plant.
There are several type of work-pieces, each represented by a distinct color. Some work-pieces can be operated upon by some tools, during which case the work-piece will slowly start changing colour. The machine T4 has tools different to those on T6! If an incorrect tool is used to process a work-piece, this work-piece will instantly turn black.
Transferring a work-piece from one conveyor to the next will require that you activate both conveyors until the work-piece arrives at the second conveyor. Due to possible slippage during transfer of work-pieces between conveyors (an issue that only occurs in real conveyors, and is not simulated here), a restriction exists that does not allow the same conveyor to ever have more than one work-piece at a time. This means that while transferring a work-piece between two conveyors, both conveyors are considered occupied. The first conveyor only becomes free when the work-piece reaches the center of the second conveyor.
The process will consist in workers manually placing P1 type work-pieces on conveyor T1 and P2 work-pieces on T2. These will then have to move to T3 → T4 → T5 → T6 → T7. From here, the work-pieces that arrived on T1 will have to go to T8, whereas the pieces that arrived at T2 will have to go to T9. You will be asked to write a program to control this process. The program will run on a PC. Since access to the I/Os (actuators and sensors) is done over the Modbus/TCP protocol, the control program itself can either run on the same PC as the simulator, or on another networked PC.

At first you will be allowed to consider that only one work-piece is moving along the cell. Later on you will be asked to write a control program that is capable of handling multiple pieces in the cell at the same time, whatever order they may arrive in at the entry conveyors.
Towards the end of the module this problem will be augmented with a physical human-machine interface (HMI). This interface will consist of a few push buttons and two lights. The lights will be used to identify the mode the system is in (RUN, STOP, P1, P2,…). The buttons will be used to change between the modes of operation.

The HMI will be built using an Arduino based board, and it will communicate with the controlling program using a different version of the Modbus protocol. All the Arduino boards will also be networked together. This will allow to add multiple 'ALL STOP' and 'ALL START' buttons that should simultaneously start and stop all the cells.
In this first week, each group is required to discuss the above requirements with a view of identifying issues that are not completely specified. For example, what to do when both T1 and T2 have a work-piece present at the same time. You should also discuss the architecture of the software you will build.

8 Mini-project – Week 2

During this second week, each group is asked to identify the timing requirements imposed by the plant.
Each group should manually activate and deactivate the actuators and try to determine what would be considered an acceptable response time for your program. This is important as since your program will be controlling the system over a network, we will need to guarantee that the network is sufficiently fast to provide the required timing guarantees.
Special attention should be placed in the rotating conveyors and the machine tools that may become damaged if the actuators are not switched off as soon as the conveyor or machine turret reaches the end of its displacement.