INDUSTRIAL ETHERNET
## CONTENTS

### 1

**THIRTY YEARS OF ETHERNET** ......................................................... 6

1.1 Still going strong ........................................................................ 6
1.2 Ethernet for industrial use .......................................................... 6
1.3 Competition with industrial networks ......................................... 7
1.4 The competition ........................................................................... 9
1.5 One standard for real-time Ethernet? .......................................... 10

### 2

**ETHERNET BASICS** ....................................................................... 11

2.1 Inception ..................................................................................... 11
2.2 Cabling versions ......................................................................... 12
2.3 Speed .......................................................................................... 15
2.4 Network addresses ...................................................................... 15
2.5 Message structure ....................................................................... 19
2.6 Receiving network messages ....................................................... 22
2.7 Virtual networks ......................................................................... 23
2.8 Software ....................................................................................... 24

### 3

**ETHERNET AS AN INDUSTRIAL NETWORK** .................................... 25

3.1 Wiring ......................................................................................... 25
3.2 Power supply for I/O .................................................................. 26
3.3 Is Ethernet real-time or not? ....................................................... 31
3.4 Is Ethernet deterministic or not? ............................................... 32
3.5 The network connector ............................................................... 35
3.6 Using special protocols .............................................................. 36
3.7 Cost ............................................................................................. 38

### 4

**SPEED** .......................................................................................... 39

4.1 Overhead per network message ................................................ 39
4.2 Comparison of speed with CAN ................................................. 40
4.3 Software delays .......................................................................... 41
5  HUBS AND SWITCHES ......................................................... 45
   5.1 Required functionality .................................................. 45
   5.2 Terminology .................................................................. 46
   5.3 Hub operation .................................................................. 46
   5.4 Switch operation ............................................................. 47
   5.5 Advanced switch capabilities ............................................ 54
   5.6 Connecting equipment ..................................................... 60
   5.7 Industrial switches ........................................................ 61
   5.8 Spanning trees ............................................................... 63
   5.9 Always use switches? ...................................................... 65

6  PROTOCOLS ......................................................................... 67
   6.1 Modbus/TCP .................................................................. 67
   6.2 Profinet ........................................................................... 69
   6.3 Ethernet/IP .................................................................. 73
   6.4 Powerlink ..................................................................... 73
   6.5 IDA .............................................................................. 74
   6.6 Foundation Fieldbus HSE .................................................. 76
   6.7 EtherCat ....................................................................... 76
   6.8 Sercos-III .................................................................... 77
   6.9 Clock synchronisation protocols ........................................ 80

7  EQUIPMENT .......................................................................... 85
   7.1 Gateways ...................................................................... 85
   7.2 Serial Device Servers / RS232 Gateways ......................... 87
   7.3 Analysers, monitors, spies and sniffers ............................ 88

8  TIPS & TRICKS .................................................................... 91
   8.1 Adding devices .............................................................. 91
   8.2 Removing devices ........................................................ 91
   8.3 Replacement of defect devices ........................................ 91
   8.4 Assignment of TCP/IP addresses ..................................... 92
   8.5 Breakdown of a hub or switch ........................................... 93
   8.6 Replace a hub by a switch .............................................. 93
   8.7 SNMP network management ......................................... 94
PREFACE

The use of Ethernet in an office-environment or at home is common. Within the industrial automation, there is no such monopoly: more than 500 different industrial network systems are competing each other. This makes connecting equipment from different vendors difficult, and increases the cost of hardware and software.

It can therefore be applauded that Ethernet is more and more being used in industry. It is important to distinguish “using Ethernet in industrial applications” from “Ethernet as an industrial network”. Superficially this might sound identical, but in fact it isn’t.

The usage of Ethernet industrial applications is quite common; since 1983, when the first Ethernet interface modules came on the market. The high speed of 10 Mbit/s, in an era where PC’s ran at 4,77 MHz, made it very attractive then and the low prices and still higher speeds (up to 10 Gbit/s) make it attractive today. In many cases, this type of use of Ethernet is done with “standard” hardware and software. An enormous amount of literature is already available on this subject.

The use of Ethernet as an industrial network is a subject for which still very little documentation is available. This publication focuses entirely on this type of usage of Ethernet, and how this relates to the operation of Ethernet. A difference between an office-user of Ethernet and an industrial user of Ethernet is that the former is not interested in (seemingly trivial) details of internal Ethernet operation, while the latter must sometimes know all the details in order to assure that the network operates correctly under all circumstances. Despite many publications in the trade press about industrial Ethernet, it is difficult to find relevant technical information void of marketing hype and commercial interests. I therefore hope that this publication fills a need.

R.A. Hulsebos
Nuenen, 5672 PL 8 The Netherlands
January 2005.

r.hulsebos@onsnet.nu

About the author
Rob Hulsebos (1961) studied Computer Science with a specialisation in datacommunications. Since 1984, he has been working for R&D departments of various companies in the Netherlands. A spin-off of his experience with industrial networks, both in theory and in practice, has led to the publication of more than 130 articles in the trade press and the publication of three books about this subject.
1 THIRTY YEARS OF ETHERNET

1.1 Still going strong
In 2004, Ethernet celebrated its 30th birthday. Quite a respectable age for network technology developed in an era in which the average memory size of a microprocessor was measured in kilobytes, the processor speed in tens of kilohertz and software was coded in assembly language. It is remarkable that Ethernet has not been retired and replaced by more modern technology. On the contrary, the 10 Gbit/s Ethernet version was launched in 2003 and development of even higher speeds has already started.

Ethernet market share in the office-environment is estimated to be more than 95%. In contrast, the use of Ethernet in industrial applications has always been limited. The industrial network market does not have such a clear market leader: more than 500 different systems are competing with each other for market share.

1.2 Ethernet for industrial use
Before 1998, Ethernet was not seen as a viable network for industrial use. This started to change in the US, where the acceptance of industrial network technology was never as high as in Europe. Ethernet is a US technology, in contrast to most industrial networks that are mainly European (most of them originating in Germany). The use of Ethernet for industrial application became a major campaign, started by American companies. Many prospective fieldbus users became concerned that fieldbus technology was already becoming outdated, even before its use became fully accepted in the industrial market. In turn, European companies also started developments for industrial Ethernet. It is interesting to note that now, some 6 years later, the future of industrial Ethernet is mainly determined by European companies (most of them German-speaking) and user-groups; the US has become a follower again rather than a leader.

The development of industrial Ethernet takes much more time than was originally anticipated. A US market research company even predicted that in 5 years (in 2003) Ethernet would be the no. 1 fieldbus, having driven all other industrial networks from the market. This turned out to be optimistic: the R&D for a completely new network protocol takes 2...3 years, and then the incorporation into products and the battle for market share takes even more time. Market researchers also showed that Ethernet technology was not really understood by them; as this publication will show Ethernet is primarily a way of cabling a network. Existing fieldbus systems are developing modifications to allow the use of Ethernet as a cabling variant. Ethernet alone cannot drive other systems from the market; on the contrary, Ethernet technology and existing fieldbus technology will merge to become a new version of existing systems.

At the time of writing, industrial Ethernet is still being developed. It is 90% identical to Ethernet as we know it. The other 10% is completely different: wiring, hardware and software. The full development of industrial Ethernet, including protocol development, system design, hardware development, etc. will take some 10 years. Five of these years have now passed. In 2005 most new systems may emerge from the R&D departments of large companies, and soon thereafter
ready to enter the market. The Profibus User’s Group expects that in 10 years from now its own industrial Ethernet solution (ProfiNet) will have a larger market share than its current system (Profibus/DP). We are now approaching the launch of the first products and their use in non-trivial applications; 2005 will be an interesting year for the industrial Ethernet market! Initially, only the machine-building and high-speed motion markets will benefit from the new technology, as almost all new developments are targeted for these application areas. Application areas that do not require such a high performance as needed for motion do not need most of the new hardware and software extensions can use the standard Ethernet, and subsequently have been available for several years already. Examples of two such systems are Modbus/TCP and Foundation Fieldbus HSE (High Speed Ethernet), being the Ethernet-versions of Modbus/RTU and Foundation Fieldbus H1, respectively.

This does not mean that Ethernet is now not used in industry – on the contrary, as soon as Ethernet became available for general use (around 1983) it was used in industrial applications. Most suppliers of PLC’s and other industrial controllers support Ethernet: a PC is now delivered with an Ethernet interface as a standard feature. This makes the use of Ethernet quite interesting – one must have good reasons for buying a £200 industrial network controller, when an Ethernet interface is ‘free’ on a PC.

Despite the support for Ethernet on industrial controllers, there has never been a standard network protocol. This meant that many proprietary protocols were developed, none of them compatible with each other. Ethernet had a reputation for being a ‘closed’ system, as products from different vendors could not communicate with each other, despite everyone claiming support for Ethernet.

The new generation of industrial Ethernet protocols is much more ‘open’ – most specifications are written by user (= vendor) groups. These protocols are also much more modern, as they incorporate current IT trends: XML, component based development, object orientation, UML, etc. This makes a connection to “office” Ethernets and “industrial Ethernets” easier, as the software from both worlds is not completely different.

1.3 Competition with industrial networks
The existing industrial networks and Ethernet are not in direct competition. This is caused by the technical optimisations that are implemented for both. Ethernet was once developed as a LAN (Local Area Network), where large amounts of data need to be transferred quickly (but not in real-time), for use in systems with abundant hardware support (CPU cycles, memory, hard disk, power supply, etc.) and where the cost of a network interface is only a small percentage of the total system cost. In contrast, industrial networks are developed for efficient and real-time transport of small amounts of data (few bytes), on equipment with limited processor support, small space, stringent power supply requirements, etc. Thus, both Ethernet and the industrial networks had their own application area, and did not become competitors. And even if some suppliers had ideas about introducing Ethernet for industrial applications, the complete lack of standards and the subsequent proprietary implementations did not arouse much interest in Ethernet.

In a typical industrial installation a number of wiring structures (sometimes a network, sometimes not) are wired in parallel (figure 1-1). There can be a network connection to the ‘outside world’ (= factory or plant LAN), a network for communication between industrial controllers, a network for remote I/O handling, wiring for safety circuits, a real-time network for motion and/or vision components, and perhaps discrete I/O for the ultra-fast signals. Of course, it is not cheap to wire a system this way, but given all the different requirements, it was not possible to do this all via one network.
In an industrial application where both Ethernet and industrial networks were used, the controllers (PLC or DCS) were often the interface between both worlds. The increasing use of PCs as industrial computing platforms caused an increase in the use of Ethernet for industrial applications as well. But Ethernet has been in use in industrial applications almost from the beginning (1980s). However, it was never used as a control network, which is where the industrial networks excel. This meant that Ethernet had more the character of an office network being used in industry, rather than being an industrial network. Since 2000 this has started to change; Ethernet as a real industrial network (a fieldbus) is about to become reality.

Yogi Berra supposedly stated, “It’s hard to make predictions, especially about the future”, but by extrapolating from the developments of the last five years, a statement about the developments for the next five years can be made. Despite the past disadvantages of Ethernet as an industrial network, this will not remain so. Several innovative ideas have already been put forward, removing or circumventing the disadvantages. Whereas figure 1-1 shows a situation in which several independent wiring structures co-exist, figure 1-2 shows a situation in which all functions could be implemented via industrial Ethernet.

Figure 1-1: In an industrial application, multiple wiring structures can sometimes be present, sometimes in the form of a network, but discrete I/O signals are also possible (i.e., for safety purposes).

Figure 1-2: The increasing use of Ethernet as an industrial network makes it possible, together with dedicated network protocols, to use Ethernet at all levels.
When Ethernet is used at all levels, the next step will be to merge all these levels – Ethernet is perfectly capable of simultaneously supporting multiple protocols independent of each other (figure 1-3).

Figure 1-3: Because Ethernet can run multiple network protocols simultaneously, it is no longer necessary to wire separate networks. In order to ensure that real-time data is delivered in time, special switches (not drawn) may be needed to prioritise real-time traffic above less important traffic.

Once Ethernet is ready for discrete applications, it will expand into new application areas – process automation and safety. The first announcements have already been made for Ethernet/IP (= Allen Bradley) and Profinet (= Siemens); all are working on a safety version and Profinet has stated that work on a version for process automation will be started in 2005. In the field of process automation, protocols supporting Ethernet are already on the market; for example, the Foundation Fieldbus “High Speed Ethernet” variant was already launched in 2003 and the development of a safety version was started recently.

1.4 The competition

Ethernet faces fierce competition from existing fieldbus systems; despite its high bit rate of 10 or 100 Mbit/s a “win” for Ethernet is not automatically guaranteed. If we look at the cycle time, an important parameter for remote I/O systems, we see that the existing fieldbus systems can easily compete with a 10 Mbit/s Ethernet. At 100 Mbit/s Ethernet a different situation arises, but only if the Ethernet protocol is carefully designed to have as little overhead as possible, both in hardware and in software. If this is not done, even a 100 Mbit/s Ethernet will be substantially slower than existing fieldbus systems.

Additionally, Ethernet is not (yet) very cheap: cabling, hubs / switches, power-supply for I/O, electronics, required processor capacity, etc. all add up to give a much higher CAPEX cost per I/O point than existing fieldbus systems. At the higher levels of automation (PLC, SCADA, etc.) this is less important, as the cost of an Ethernet interface is only a minor percentage of the total CAPEX cost.

Industrial Ethernet may develop further to become a tough competitor for other fieldbus systems. Its disadvantages may be removed through the development of new innovations; in 2003 and 2004 several new innovations have been introduced. The fieldbus systems currently available, now approaching their 15th birthday, are more or less stable. Industrial Ethernet, now at age 5, still has a long way to go. Some of the new network protocols are not mature yet; this can be easily seen when new protocol versions are issued three years in a row.
Ethernet is also facing competition from other high-speed network technologies such as USB and Firewire. As these are available in every modern PC they are as cheap to use as Ethernet, since no extra network interface boards have to be bought. However USB has never been designed for industrial use, and is not seen as a serious contender. Of course, it is possible to control I/O with it (a bit is a bit). Firewire has a more industrial background, because it is already in use for high-speed motion and machine vision applications. But the maximum allowable distance is quite short (tens of meters), and the limited support by suppliers makes it a niche-player.

1.5 One standard for real-time Ethernet?
As of the end of 2004 there is no standard for real-time Ethernet protocols, and there probably never will be. In 2003, the IEC (International Electrotechnical Commission) has started work on the real-time Ethernet standard. The plan is to finish this in 2005, and the definitive release (after the member countries have voted) is expected in 2007. In anticipation of this release, parts of the new standard have already been implemented in various new network protocols (see chapter 6). It is expected that the ‘standard’ will simply be a collection of specifications for protocols that are all incompatible with each other. This is simply a repeat of the “8 headed monster”, IEC 61158, ‘standardising’ eight industrial networks in 2003. Some Ethernet-based protocols have already been incorporated in IEC 61158, so another standard is simply a duplication of effort. In another venue, the ‘standard’ for an industrial Ethernet connector lists 10 different versions, designed by 10 different companies. The lack of agreement within the industrial community will be a problem for users for years to come.
2

ETHERNET
BASICS

2.1 Inception
The first ideas on Ethernet originated at Xerox Research around 1970. This very first version of Ethernet was continuously developed further, until around 1980 a speed of 10 Mbit/s was reached. This now looks quite slow, but in an era where the average PC ran at a clock speed of 4.77 Mbit/s any network running at 10 Mbit/s looked infinitely fast. Xerox teamed up with Digital Equipment and Intel to develop Ethernet further. It was called “DIX Ethernet” after the first letters of the company names.

The IEEE 802.3 committee
In 1980 the IEEE (Institute of Electrical and Electro-technical Engineers) set up a committee to manage the continuing development of Ethernet (and all other local area networks as well). Because this committee was founded in 1980 and because it was the second new committee that year, it was given the label 802. Subcommittee 3 was assigned the task of developing Ethernet further. The full “name” of the committee is IEEE 802.3. In many books, leaflets, commercial folders etc. “Ethernet” and “802.3” are regarded as synonyms.

New extensions to Ethernet are regularly published and are indicated by two (formerly only one) letters following the committee number, i.e. “802.3ah”. After a few years the committee consolidates all extensions, and a new version of the IEEE 802.3 standard is published (this last occurred in 2002). The IEEE 802.3 standard is also published as IEC / ISO standard: 8802.3.

<table>
<thead>
<tr>
<th>Standard</th>
<th>Year</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>802.3a</td>
<td>1985</td>
<td>10BASE-2 (Thin Ethernet)</td>
</tr>
<tr>
<td>802.3c</td>
<td>1985</td>
<td>10 Mbit/s repeater specification</td>
</tr>
<tr>
<td>802.3d</td>
<td>1987</td>
<td>Fibre-optic repeaters</td>
</tr>
<tr>
<td>802.3i</td>
<td>1990</td>
<td>10BASE-T (Twisted-pair Ethernet)</td>
</tr>
<tr>
<td>802.3j</td>
<td>1993</td>
<td>10BASE-F (Fibre-optic Ethernet)</td>
</tr>
<tr>
<td>802.3u</td>
<td>1995</td>
<td>100BASE-T (Fast Ethernet)</td>
</tr>
<tr>
<td>802.3x</td>
<td>1997</td>
<td>Full-duplex version</td>
</tr>
<tr>
<td>802.3z</td>
<td>1998</td>
<td>1000BASE-X (Gigabit Ethernet)</td>
</tr>
<tr>
<td>802.3ab</td>
<td>1999</td>
<td>1000BASE-T (Gigabit Ethernet via twisted-pair)</td>
</tr>
<tr>
<td>802.3ac</td>
<td>1999</td>
<td>Frame size extension</td>
</tr>
<tr>
<td>802.3ad</td>
<td>2000</td>
<td>Link aggregation</td>
</tr>
</tbody>
</table>

Table 2-1: Several extensions to the original 802.3

Other Ethernets
Other networks that also have Ethernet in their names are: isoEthernet, Ethernet/IP and Safe Ethernet.

IsoEthernet (Isochronous Ethernet) is an adaptation of the original 10 Mbit/s Ethernet. By transmitting data on the cable using a different electrical method, the speed has been increased to 16 Mbit/s. The extra 6 Mbit/s are used to carry 97 ISDN telephone channels (“96B+D” in
ISDN-talk). This would make it possible to carry an Ethernet network and a telephone signal over the same networks. This is technically outdated; “VoiceOverIP” now carries telephone traffic via the TCP/IP protocol in software instead of in hardware.

Ethernet/IP is a protocol of Allen-Bradley, discussed in more detail in chapter 6.

Safe Ethernet is a protocol of the HIMA company, specially designed to allow safety applications to use Ethernet. By keeping the network load low (< 10%) the network is guaranteed to react quickly enough, an essential characteristic of a safety network.

2.2 Cabling versions

Ethernet supports a large number of wiring variants. The original Ethernet called “10Base5” used a thick (1 cm diameter) and thus sturdy coax cable, allowing a network of maximum 500 meters in length. At distances of 2.5 metres and multiples of this, a maximum of 100 devices could be connected via drop cables of maximum 50 metres in length. This Ethernet version has been given the nickname “Thick Ethernet”.

The Ethernet versions are named according a standard system (see also table 2-1 at the right-hand side). First, a number is used to indicate the speed in Mbit/s, or for the most recent version in Gbit/s (10G). Next, the transmission technology is given: BASE for baseband networks, BROAD for broadband networks (now no longer used). Last, in older versions the maximum size of the network is given in units of 100 metres, but as of 1990 this was replaced by a letter combination indicating the wiring version. As an example: “100BaseF” is a network running at 100 Mbit/s, using baseband transmission, and using fibre optic. Another example: “10Broad36” runs at 10 Mbit/s, uses broadband technology, at a maximum distance of 3.6 kilometres. In modern Ethernets the maximum size of the network is no longer determined by the wiring technology, since switches and routers can be used to extend the network indefinitely.

CheaperNet

The use of the thick coax cable resulted in many complaints. An Ethernet version was developed (by 3Com) which allowed the use of thinner, more flexible coax cable. This is called the “10Base2” version. It supports 30 devices at a maximum network length of 185 metres. Devices are coupled to the network via T-connectors, which can be done in a few seconds. Because the cabling was so much cheaper, 10Base2 is often nicknamed “Thin Ethernet” or “CheaperNet”.

Despite its advantages CheaperNet was not really maintenance friendly. Theoretically, the coax cable is a “bus”, one long electrical wire to which all devices are connected. Any problems with the network: length too long, too many devices, incorrect termination, cable interruptions, shorts, earthing and shielding problems, electro-magnetic interference etc. are ‘seen’ by all devices, and may thus impair everybody’s correct operation. This is one of the major drawbacks of any bus-wired network, not only on Ethernet but also on existing industrial networks that are wired this way. It is difficult to find the exact physical location of a problem, other than by physically checking all devices and the whole cable run.

From a maintenance point of view a coax-based Ethernet was not very practical, because wiring problems cannot always be found and solved quickly. The network may be out of order or unable to function at its rated capacity. A solution to this was found several years later (see next section). This also marks the end of coax-based Ethernet versions, which are only available at 10 Mbit/s but not at higher speeds.

Twisted-pair

Ethernet became popular when the twisted-pair version was developed, as it took away most of the disadvantages of the coax-based versions, and was cheaper. The version was called “10BaseT”; the maximum distance was no longer mentioned in the name. A major difference between the older versions and 10BaseT was that a new device was necessary: a “hub”. Each
device on the network is connected via its own cable (maximum length 100 metres) to the hub. It functions like a telephone exchange: just as conversations between subscribers are handled by the exchange, a hub receives messages from the devices connected to it, and it forwards these messages to their destination.

For network managers, 10BaseT was a huge improvement compared to coax-based Ethernets. Because each device has its own cable, electrical disturbances can only influence one device, not all devices. The total capacity of the network is determined by the number of connection ports on the hub, and not by the electrical characteristics of the cable. Adding a new device has no influence on the wiring of the existing devices. Finally, a hub can be given intelligence (it is then called a "manageable hub"); it can monitor traffic on all its ports, count errors, and can be given a command to shut down a connection port when a device generates too many errors, too much traffic, or is seen as a security risk. The SNMP (Simple Network Management Protocol) is a very popular protocol for managing hubs from a distance. Taken together the introduction of 10BaseT did much to improve the availability of Ethernets, reduce the down time, and improve the manageability and extensibility, all of which helped to make Ethernet the most popular LAN.

After 10BaseT came 100BaseTx. At first sight the cabling and the connectors are identical, but this is not so – 100BaseTx requires a better quality cable that supports the higher frequencies needed for such a high speed. This cable is called “CAT5”; for 10 Mbit/s “CAT3” is sufficient. However it is recommended that when a 10BaseT network is built nowadays, CAT5 cabling is used – it allows the network to be upgraded to 100 Mbit/s should the need arise, without having to replace all cables. CAT5 is not the best quality cable available for Ethernet – CAT5e, CAT6 and CAT7 also exist but these are more expensive. Sometimes it is not good to have a better quality cable than you actually need – such cables are designed to transmit higher frequencies, but they also pick up higher frequency interference, so the result in the end might not be what was expected of such expensive cable.

**Optical fibre**

Optical fibre is another wiring version of Ethernet, and available for all speeds (10BaseFL, 100BaseFX, etc.). The distances that can be bridged depend on the type of fibre and the wavelength of the lasers used; distances from several hundred metres up to tens of kilometres are possible. Some vendors (i.e., Hirschmann) have their own proprietary fibre optic technology to allow even longer distances, at speeds of Gbit/s.

**Wireless**

There are wireless versions of Ethernet, known as IEEE 802.11. The 802.11 committee is responsible for the development of wireless network protocols, and decided to make several 802.11 versions (a, b and g) compatible with Ethernet (not all 802.11 network protocols are Ethernet compatible, i.e. Bluetooth and Zigbee certainly are not). The usage of wireless Ethernet in the consumer market has made wireless Ethernet equipment very cheap.

Wireless Ethernet versions existed before IEEE 802.11 came into being. Many companies have developed their own wireless Ethernet products. These are not based on the IEEE standard, but use their own radio implementations at frequencies that sometimes do not allow high speeds but do allow much longer distances than is possible with IEEE 802.11.
A disadvantage of wireless networks is security, as anyone can tune in to the radio frequency in use, and listen to all network traffic. Despite encryption techniques, it appeared to be very simple to break the encryption; any hacker with a laptop can break the encryption of a busy network in a few hours and sometimes within 15 minutes. In 2004, a new encryption technique called WPA2 was standardised, but not after earlier proposals for a stronger encryption proved to be even weaker than the original encryption technique (called WEP). Hackers make it their hobby to drive through a city in a car with a laptop capturing the signals of wireless networks, determining whether the security is enabled or disabled. Statistics show that in 60% of networks, security measures are not even turned on!
from other network protocols like Bluetooth and Zigbee that are especially designed for low power requirements and long battery life (several years).

2.3 Speed

Ethernet originally started at a speed of 10 Mbit/s, which was later increased to 100 Mbit/s and 1 Gbit/s, and in 2003 the 10 Gbit/s version was released (10GBaseT). This is not the end; the ever-increasing demand for bandwidth will undoubtedly lead to a 100 Gbit/s version. For industrial use, especially at the lower levels in the automation hierarchy, 100 Mbit/s is fast enough at present even for the most demanding application (= high-speed motion). The 10 Mbit/s version can be used for industrial automation, but existing industrial networks can easily compete with regard to speed of I/O handling, and are also much cheaper in their wiring. Most new industrial Ethernet products nowadays support 100 Mbit/s; a few years back this was still the exception.

A 1 Gbit/s Ethernet is not efficient for many industrial applications. Ethernet was originally developed to transport large quantities of data, and is then very efficient (> 90%). The drawback is that for small quantities of data, which is the case when processing I/O, Ethernet has a large amount of overhead (> 95%). The 1 Gbit/s version adds another factor 8 of overhead to this, caused by its being backwards compatible with the 10 and 100 Mbit/s versions of Ethernet. All current developments of industrial network protocols (see chapter 6) assume 100 Mbit/s to be currently sufficient. Using a gigabit Ethernet is thus mainly useful at the higher levels of industrial automation applications, where larger amounts of data are transmitted. Another application area for gigabit Ethernet is in transporting audio and video signals, for example pictures from security cameras on an industrial site. Although gigabit Ethernet was quite expensive, this will change quickly, as more and more PC vendors integrate a gigabit network interface on their PC motherboards.

The use of multiple bit rates on Ethernet does not cause as many problems as with other networks, because a hub or switch is always needed. The more intelligent hubs / switches support the “auto-negotiation” feature, whereby the hub / switch negotiates with each connected device, and chooses (per device) the highest common bit rate. The hub / switch handles all conversions between devices automatically. This makes it very easy to mix 10 and 100 Mbit/s equipment on a network, and allows for easy changes when older equipment is removed and replaced by faster equipment.

2.4 Network addresses

Like most other networks, each device on an Ethernet must have its own, unique, network address. It is officially called the “MAC-address” (Medium Access Control).

Ethernet has a very wide range of addresses: up to 281 billion addresses can be formed with a 48-bit number, enough to support worldwide unique addresses. The IEEE has devised a scheme to make this possible; the 48 bits are divided into two fields of 24 bits each and used as follows:

- The highest 24 bits indicate who the supplier of a device is. This field is called the “Organisationally Unique Identifier” (OUI). The IEEE assigns this value on request of a supplier.
- The supplier determines the value of the lowest 24 bits. No technical or commercial significance should be attached to the value by the user.

1 The Ethernet MAC-address is only used for Ethernet itself. For higher protocol layers it may be necessary to have a form of network addresses of its own. For example, TCP/IP network addresses are 32 bits in size. The usage of IP-addresses is completely independent of Ethernet MAC-addresses. Of course, a translation from one type of address to the other has to be done in a functioning network, and TCP/IP has support for this in the form of “ARP” (Address Resolution Protocol).
A supplier is free to determine the values of the lowest 24 bits (usually this is serial number) as long as each value is used only once. There are still 16 million combinations possible with 24 bits, large enough for most companies. When 90% of all possible values are used, a new OUI can be bought from the IEEE (see http://standards.ieee.org/reqauth/oui/forms/).

When a supplier builds an Ethernet interface board, the MAC-address must be programmed. Usually this is done via flash-memory or a small (E)PROM or something similar. Which MAC-address has been used can normally be found externally: in the documentation or printed on a label glued to the device. The MAC-address is given in hexadecimal format, 6 numbers separated by a dash or colon, for example “03-80-86-12-20-04”. The first three numbers (03-80-86) are always the OUI, and the remaining three numbers “12-20-04” are the supplier’s own number.

There are a number of special values for the MAC-address, to distinguish “normal” Ethernet messages from so-called “broadcast” or “multicast” messages.

The “Organisationally Unique Identifier”
The first three bytes (24 bits) of the MAC-address form the so-called “Organisational Unique Identifier” (OUI), sometimes also called “Company ID”. The IEEE determines the value, after payment of an administrative fee. There are 8 million possible values for the OUI, only a very small number of which are now in use. Some examples:

<table>
<thead>
<tr>
<th>OUI</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-A0-45</td>
<td>Phoenix Contact</td>
</tr>
<tr>
<td>00-09-5C</td>
<td>Philips Medical Systems</td>
</tr>
<tr>
<td>00-05-1A</td>
<td>3Com</td>
</tr>
</tbody>
</table>

Some companies have been assigned multiple OUI, not necessarily because they needed them, but because of mergers or acquisitions of other companies. For example Advantech China has OUI 00-0B-AB, Advantech Taiwan has OUI 00-D0-C9 and Advantech Canada has OUI 00-E0-02. On the other hand, some OUI values are no longer used because the requesting company no longer exists.

The first byte of the OUI must always be an even value. This is because the least significant bit must always have value ‘0’. This bit is not used for creating individual MAC addresses, but for identifying what are known as “multicast groups” (see below).

Users of the PPP-protocol have been given a special OUI value: CF-xx-yy. See RFC-2153 (http://www.faqs.org/rfcs/rfc2153.html) for further details.

If you would like to determine your own value for a MAC address (some network interface modules allow this), you must officially ask the IEEE for an OUI value. In practice, this is not always done, especially for standalone or temporary networks. As there are no checks done in Ethernet on where the value of an OUI comes from, any value will do. Of course, when two or more network devices erroneously get the same MAC address, the communication with these devices will not function properly, and it may take a while before the root cause is found.

Individual Address Blocks (IAB)
Companies that do not need 16 million MAC addresses can ask the IEEE for a smaller block of 4096 MAC addresses to be assigned to them. This is called an “Individual Address Block” (IAB). Of the 48 bits Ethernet address a company can then assign the lowest 12 bits only. IAB MAC addresses always begin with 00-50-C2. Some examples:

<table>
<thead>
<tr>
<th>OUI</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-50-C2-00-30-00 through 00-50-C2-00-3F-FF</td>
<td>Microsoft (Redmond, US)</td>
</tr>
<tr>
<td>00-50-C2-01-10-00 through 00-50-C2-01-1F-FF</td>
<td>Bihl &amp; Wiedemann GmbH (Mannheim, D)</td>
</tr>
</tbody>
</table>
It makes no difference to the operation of Ethernet whether a ‘normal’ OUI block of MAC addresses is used or an IAB block of MAC addresses; in both cases Ethernet just sees it as a group of 48 bits. The usage of IAB MAC addresses is purely an IEEE administrative measure.

**Broadcast addresses**

When the destination MAC-address is set to the hexadecimal value FF-FF-FF-FF-FF-FF by the sending device, it is a so-called “broadcast” message that is accepted by everyone on the network and passed on to higher network protocol layers for further processing. Every modern Ethernet network controller automatically does this. A finer distinction on the contents of the message is indicated in the ‘type’ field of the Ethernet message. This field is used by the Ethernet driver software to pass the message on to the software protocol driver for that particular protocol. A few examples of possible values for the ‘type’ field are:

- **0600** XNS protocol (Xerox Network Services)
- **0800** IP (e.g. RWHOD via UDP) as needed
- **0804** CHAOS protocol
- **0806** ARP (for IP and CHAOS) as needed
- **0BAD** Banyan protocol
- **1600** VALID protocol
- **8035** Reverse ARP (RARP) protocol
- **807C** Merit Internodal (INP) protocol
- **809B** EtherTalk protocol

A device will only process those broadcast messages with a certain type for which there is a protocol driver available; if this is not available, the message will be ignored. This filtering is done by software. The processing delay incurred on all devices on the network is a penalty for the application programs running on those devices. The use of broadcasts should therefore be kept to a minimum as much as possible. Some protocols are known for their bad behaviour in this respect; for example the XNS protocol transmits 24 broadcasts per device per second. In larger networks (not anticipated when XNS was developed!) this may become a burden on the network and its devices.

The transmitter of a message normally places its own MAC-address value in the ‘sender’ field of an Ethernet message, so the receiving devices know to whom to send their replies. If the broadcast address FF-FF-FF-FF-FF-FF is written in the ‘sender’ field (abuse!), the answer is broadcast too. This may cause what is known as a ‘broadcast storm’.

**Multicast addresses**

Sometimes it is desirable to have functionality like a broadcast, but without broadcasting to every device on the network. Such functionality is called “multicast”, where a network message is transmitted to a certain group of devices on the network. Multicast MAC-addresses are reserved for this purpose.

A multicast MAC-address can be recognised by having bit 0 (rightmost) of the first byte in the MAC-address set to ‘1’. The location of this bit may seem strange, as usage of bit 7 (leftmost) would seem more logical. The reason for the usage of bit 0 becomes visible during transmission: it is transmitted as the first bit of the actual Ethernet message; Ethernet transmits byte from bit 0 to bit 7, and not the reverse as is usual in many other networks. The receiver of a message thus knows immediately after having received the first bit, whether this message is being multicast or not.

The further handling of the multicast message is done via software. This is necessary, as an Ethernet controller board cannot ‘know’ which multicast groups this device is a part of. A proto-
The col that has functionality for handling multicasts is IP (from TCP/IP). IP addresses are 32 bits in size, and a small part of the 4 billion possible addresses is reserved for IP multicasts. These are known as “GDA’s” (Group Destination Network addresses) and are found in the “Class D” address range of IP, with values of 224.0.0.0 to 239.255.255.255. Written in binary, the leftmost 4 bits are always ‘1’, and the other 28 bits indicate the multicast group for which a message is destined. The GDA’s on their own have nothing to do with Ethernet; it is possible to use TCP/IP with other networks than Ethernet. On Ethernet the conversion from a GDA-address to a matching MAC-address is done as follows:

- The first 3 bytes get the hexadecimal value 01-00-5E. By having this fixed value (in the OUI field) Ethernet switches and routers and software drivers can immediately recognize a multicast message. This is especially of interest for the “IGMP Snooping” feature in switches and routers.
- The remaining 3 bytes are filled with the 23 lowest (rightmost) bits from the GDA. Although the GDA itself is 28 bits, there is simply not enough space in the MAC-address field.

For example, IP-address 224.0.0.1 is converted to MAC-address 01-00-5E-00-00-01, and IP-address 225.10.20.30 to MAC-address 01-00-5E-0A-14-2D. Please note that a full conversion from a GDA to a MAC-address is not possible: 32 different GDA’s give the same MAC-address. It is thus always necessary to have some level of filtering in software on all multicast messages received.

There is one very special multicast MAC-address: 01-80-C2-00-00-01. Actually, it has nothing to do with multicast transmissions at all, because it is only used on point-to-point connections between two devices. It is used for flow control purposes; chapter 5 gives a more detailed explanation.

**Determining the MAC-address**

In order to send a message to a device on the network, its MAC-address must first be known, just as we have to know someone’s telephone number before we can call that person. The supplier of the device has determined the value for the MAC-address, and has the responsibility of informing us of the value. As it is usually impossible to open a device and read the MAC-address from flash memory, many suppliers glue a label to their device on which the MAC-address is printed. I recommend writing this MAC-address on the device’s manual or somewhere else for backup purposes; if the MAC-address value of a device is lost it could mean that no further communication is possible. Sometimes it is possible to read the MAC-address from a device via software, but this should not be relied upon, especially in the more exotic software operating systems in use in the industrial automation community.

![Figure 2-3: An example of a MAC-address label ("Ethernet Address") of a PC network controller board.](image)

Figures 2-3 and 2-4 show two examples of the retrieval of MAC-addresses. On a PC (running Windows) it is quite easy to retrieve the local MAC-address: via Start -> Control Panel / Configu-
ration -> Settings, and then select the network driver. Alternatively, the DOS command “ipconfig /all” can be used. The MAC-addresses of all communication partners of your PC can be retrieved via the DOS command “arp –a”, which is also available on Unix and Linux systems.

Note that all sorts of different words are used for the MAC-address (“MAC-ID”, “physical address”, “Ethernet address”, “EA”, “Adapter address”, “INC”, etc.). The fact that they are always six bytes in size makes them quickly recognisable as being a MAC-address. The official notation uses dashes between the 6 numbers, but colons are often seen as well.

![Figure 2-4: An example of a software package for the “BOOTP” protocol, which only recognises those devices for which a MAC-address has been configured in advance.](image)

2.5 **Message structure**

Each Ethernet network message has an identical structure, consisting of six fields of various sizes (figure 2-5).
Each message always begins with a preamble of 64 bits, consisting of alternating 0 and 1 bits (except the last bit, which is also a 1). The purpose of this field is to re-align the electronic clock circuits of the receiver with that of the transmitter. Even when both are set at the same speed (10 or 100 Mbit/s) there are small deviations at sub-second level caused by temperature differences, the variability of electronic parts, etc. By performing a re-alignment on every network message, speed variations are minimised. Every network, thus also Ethernet, does this.

The next field is the 48 bits MAC-address for which a message is destined. The sender of the message determines its value. Next comes its own MAC-address (also 48 bits).

The fourth field is 16 bits in size, and indicates how many bytes of application data follow. The maximum value is 1500, as this is the size of the data field (coming next). An exception to this are the so-called “Jumbo frames” that allow up to 9000 bytes (see below).

The fifth field has a variable size; it contains the application data. What its exact contents are is not of importance for Ethernet, which just sees bytes without having to know what they mean. The data field must always be at least 46 bytes in size; the maximum size is 1500 bytes. If the application has less than 46 bytes to transmit, fillers must be added. Why 46 bytes? This is caused by the minimum size of a total Ethernet message, which is 64 bytes. Deduct from this the two MAC-address fields (2*6 bytes), the size field (2 bytes) and the CRC field (4 bytes), and you are left with 46 bytes. Note that the size field must always contain the real length of the application data, which is thus different from the ‘transmission length’.

The sixth and last field is the “Cyclic Redundancy Check” (CRC), which is used to detect transmission errors. The sender of a message calculates the CRC from the value of the application data. The receiver performs the same calculation; if the locally calculated CRC is identical to the received CRC the message has been received without errors. If there is a difference, one or
more bits in the network message have been corrupted (0->1 or 1->0). Since the corrupted bit(s) is/are not known, it/they cannot be repaired, and the whole message is discarded. It is up to higher-level protocols (for example, TCP) to repair the loss of a message.

Finally, a transmission silence of 96 bits “Interpacket Spacing” (IPS) follows, to give all receivers time to process the message internally. This field is not part of the actual network message itself. However, when calculating the performance of an Ethernet the relatively large size of the IPS must be taken into account. It is the experience of the author that this is often forgotten, and the real performance of the network can be 15% off the calculated value, not something that can always be silently ignored.

**Efficiency**

When adding the sizes of all fields together, any network message always needs a minimum of 64+48+48+16+46*8+32+96 = 672 bits transmission time (= 67.2 µsec at 10 Mbit/s, 6.72 µsec at 100 Mbit/s), even when sending only 1 byte of application data. This gives an efficiency of 8 bits / 672 bits = 1.2%. This shows that Ethernet is not particularly suitable for transmitting small quantities of data, which is uncommon in an office environment, but very common in industrial applications. One of the reasons for the existence of industrial networks is that they are heavily optimised for small quantities of data. The EtherCat and Sercos-III protocols make Ethernet more suitable for small quantities of data (see chapter 6).

When the “Virtual LAN” (VLAN) extension of Ethernet is used (IEEE 802.1p, see below) one extra 4-byte field is added to each network message. This is not of importance when calculating the efficiency of Ethernet, because the minimum length of the data field is decreased by 4 bytes (from 46 to 42 bytes). So with or without VLAN’s an Ethernet message always requires a minimum of 672 bits transmission time.

**Jumbo frames**

In 1998 the company Alteon (now part of Nortel) made a proprietary extension to Ethernet for their own products to support the so-called “Jumbo frames”. These are normal Ethernet messages, but allow up to 9000 bytes of data instead of only 1500. The purpose of jumbo frames is to decrease the processing load (in software) on devices connected to 1 or 10 Gbit/s networks. At these high speeds so many messages can be received in one second (19,000,000 small messages, or 800,000 maximum size messages with 1500 bytes data) that no existing processor is able to process them all. By allowing larger amounts of data to be sent in a single message, up to 5/6 fewer network messages are necessary, decreasing the processor load considerably. Jumbo frames are not officially supported in Ethernet; however many suppliers recognise them (but many do not). Via the “auto negotiate” feature of Ethernet two devices can determine at run-time whether both support jumbo frames, and if so, use them.

Note that a jumbo frame is split into several smaller messages by a router when it cannot be transferred further on a network section that does not support jumbo frames. Once split up, these smaller messages (actually normal-size Ethernet messages) are not recombined into the original jumbo frame when another router detects that another section of the network allows jumbo frames again.

**Maximum Transmission Unit (MTU)**

If the TCP/IP protocol is used, a configuration parameter for TCP/IP is the maximum amount of data (bytes) that can be sent in a TCP/IP message. This parameter is called the “Maximum Transmission Unit” (MTU, MaxMTU). Because an Ethernet message has room for 1500 bytes of data, MTU can be set to 1500 bytes. If an application wants to send larger amounts of data, IP will split this data up into fragments of MTU bytes each, send them individually on the network, and reassemble the fragments at the receiving device.
For reasons of performance, it is sometimes advisable to set the MTU to a value smaller than 1500. This makes a great deal of sense when a slow connection is used (i.e., telephone line, wireless network), or a connection with a high error rate in the network path between sender and receiver. When a network message is corrupted due to electromagnetic interference, the higher-level protocols (such as TCP) will attempt to repair the error by resending the message, in the hope that it will not be corrupted again: the shorter the message, the higher the probability of success. Setting the MTU to a smaller message can thus sometimes considerably increase the communication speed between devices, simply because less time is lost in retries. This trick is often used by PC-based "internet speed boost" programs. Although these may work well on telephone lines, they might interfere with network traffic that does not leave the Ethernet at all. Usually the MTU value is valid for all network messages; using a smaller MTU then decreases communication speed because each network message has its own overhead and processing delays.

The Maximum Transmission Unit is also the Maximum Receive Unit (MRU). Theoretically, a network can have a MRU with a different value than the MTU. Some Ethernet protocol drivers ignore messages coming in whose length is larger than the MTU; others don’t.

2.6 Receiving network messages
An Ethernet controller will hear all sorts of network messages; exactly which ones depend on the wiring of the network (coax, UTP, use of hub or switch), and the application(s) using the network. The Ethernet controller has to filter out those network messages addressed to it through all it hears, and ignore all other messages. Because an Ethernet controller knows what its own MAC-address is (as programmed by the vendor), the filter is programmed to pass the following through for further processing:

- Those network messages for which the "destination MAC-address" field is identical to the own MAC-address; and
- Those network messages for which the "destination MAC-address" field is identical to the special broadcast MAC-address (48 times '1').

All other network messages are silently ignored. Because the filter is executed in hardware, no CPU processing power is necessary.

Promiscuous mode
Some network interface controllers allow themselves to be set to "promiscuous mode". This is a very special mode of operation in which the controller receives all messages sent by all other devices on the network, a sophisticated form of wire-tapping in fact. This is very useful for analysing network traffic (also see chapter 7).

Setting a network interface controller to promiscuous mode is a security leak, because all network messages can then be analysed, file I/O read, passwords seen, etc. Some network administrators therefore disallow usage of promiscuous mode. Although difficult to detect from the outside, programs can be found that on Internet that reportedly can remotely detect whether a device is in promiscuous mode.

Differences between coax and twisted-pair cabled networks
On a coax-wired Ethernet each device will automatically receive all transmitted network messages; the coax is a shared medium where everyone always listens to everyone else. When using a network analyser this is very handy. When using a twisted-pair network with hubs, the same holds – every device receives everything. In this regard the use of a network analyser is as simple as with coax. In each device the Ethernet controller board will filter all incoming messages for the correct MAC-address, and only the matching values will be passed on for further processing to the software (the network protocols).
On a twisted-pair network with switches the MAC-filtering function is largely done by the switch; a device only receives those messages meant for it. Even though the MAC-filter is present, it actually does not have to do much. It is not disadvantageous to the speed of the network. When using a network analyser it is disadvantageous to have MAC-filtering in the switch; Setting the analyser’s network interface to promiscuous mode does not help at all, since the switch blocks all messages not meant for the network analyser; only broadcast messages are passed on.

2.7 Virtual networks

VLAN (Virtual LAN) is an extension to Ethernet originally developed for managing office networks where different groups of devices may not ‘see’ each other’s traffic on the network. This is a functional compartmentalisation, which looks strange – devices are connected to a network in order to communicate with each other, not to prevent them from communicating. But in a large company different groups of users have different requirements – confidential information that may not leave a department, prevention of the spreading of worms and viruses, a sub network for testing new software before deployment, a sub network for connecting laptops with unverified software packages, a sub network for connecting visitors’ equipment, etc. Of course it is possible to wire completely independent networks, but then communication with file servers, databases, printers, etc. is only possible when they are all connected to the same network. A relocation of offices or introduction of new staff may require extensive rewiring. It is actually much easier when there is one network “in hardware”, but when “software” can make this network appear like several separate networks, we have the advantages but not the disadvantages. This is where VLANs come in.

How does this work? The Ethernet extension IEEE 802.1q gives every device a “tag”, in which the “virtual” networks are listed to which that device belongs. The device can only communicate with other devices that are members of the same virtual network(s). The tag has a 4-byte value, which is sent in every network message. It can be compared to having multiple ‘channels’ on the radio; if a receiver is tuned to the right frequency (= the same virtual LAN) it can send and receive on that channel (VLAN).

Technically, working with VLANs is quite similar to normal Ethernet operation – those devices for which a network message is intended receive it; all others do not see it. VLANs operate differently from normal Ethernet where broadcasts are concerned – these are not sent to all devices, but only to those devices operating on the same VLAN. Technically speaking such a broadcast is not really a broadcast anymore. Broadcasts are often used for network management and application services – for example all printer servers can regularly broadcast that they are up-and-running, and any PC which needs to find a printer can just listen a while for incoming broadcasts from printer servers, and after, say, 10 seconds the user can pick his printer from a menu. This is also a security risk – just by listening for broadcasts, the complete infrastructure of all devices connected to a network and the services they offer becomes visible. When using VLANs, a device only sees the broadcasts of the devices it is allowed to see.

A switch (or router) plays a very important role in the management of VLANs, because these devices ‘know’ which devices are connected to the network and to which VLANs they belong. A VLAN can thus only be built when the switches and routers also support VLANs. As VLAN support is not a standard function of switches / routers, care must be taken to order to correct equipment when VLAN support is needed.

It might look as if the use of VLAN in industrial applications is superfluous, as industrial devices are not security risks to the same extent as PC’s are in office automation. But still, VLANs can be successfully employed in industrial networks, for example to separate real-time traffic from a controller to his I/O devices from the traffic to / from HMI (Human/Machine Interface) devices, or traffic to / from databases. The separation of I/O traffic from the rest is especially necessary in many of the new protocols (like ProfiNet, IDA, Powerlink etc.) as these make heavy use of broadcasts. This would cause a huge processing burden for all non-I/O devices because they
receive these broadcasts that are not meant for them, but each message must be (partially) processed before a decision can be made to ignore it. By using VLANs the I/O devices can operate in their own virtual network separate from the other devices, which no longer receive the broadcasts.

2.8 Software
One is tempted to conclude that the large market share of Ethernet in the LAN-market also means that it is also setting the standard in the area of network protocols. On one hand, it is a correct conclusion, but on the other hand it is a wrong conclusion. In order to get a working Ethernet, network protocols are necessary, for example from the TCP/IP suite. But TCP/IP is not Ethernet and vice-versa, both were developed independently. However, in practice we see that many people regard Ethernet and TCP/IP as synonyms – where there is Ethernet, there is also TCP/IP. The market share of Ethernet undoubtedly has profited from the popularity of TCP/IP; its abundant support in all automation platforms (office and industrial) makes it a natural first choice for anyone building an Ethernet-based network. But it is important to keep the distinction in mind – Ethernet is more the “hardware” side of the network, whereas TCP/IP is the “software” side.

TCP/IP is also widely used in industrial automation, but not to the same extent as in office automation. This is caused by a lack of support for industrial applications in TCP/IP and its bad performance in real-time applications. Some vendors have made their own (proprietary) extensions to TCP/IP, allowing their own products to communicate with each other, but not to products of other suppliers. Even though everybody supports Ethernet and TCP/IP, communication between equipment from different suppliers was impossible. This has had its effect on the popularity of Ethernet in industrial automation.

Since 1999 many new developments have been started to make Ethernet suitable for real-time applications, made possible by the introduction of specially developed protocols, such as Profinet, Sercos-III, Ethernet/IP, IDA, and others (see chapter 6). But even with the technical issues now solved, there is still no single protocol – equipment with Profinet cannot communicate with IDA devices, IDA cannot communicate with Sercos-III, even though all are using Ethernet. Time will tell whether one of the new protocols gets a majority market share, or whether all get a small piece of the cake, as happened ten years ago with the industrial networks Profibus, CAN, Fieldbus Foundation, etc.

Ethernet has one advantage above the existing industrial networks: it is possible to run multiple protocols simultaneously over the same network. Most industrial networks do not support this: if a decision has been taken to use Profibus and later this turns out to be a bad choice and another network is chosen, all network cabling has to be redone. With Ethernet, the existing cabling can be re-used for the other protocol: sometimes even several protocols can run concurrently, allowing for a gradual migration.
Not everyone takes it for granted that Ethernet is capable of being used as an industrial network (fieldbus). In this chapter we discuss the various developments in ‘hardware’ (cabling, connector, ...) and ‘software’ (protocols). At the time of writing, many developments have not yet been finalized; some parts of Ethernet, which are now seen as disadvantageous to it being used as a fieldbus, may be circumvented by new innovations, some of which have already been announced. I expect many more innovations in the coming years, removing the disadvantages of ‘standard’ Ethernet one by one and making it fully useable in industrial environments.

3.1 Wiring
An office-based Ethernet nowadays always uses UTP (Unshielded Twisted Pair) cabling, but for industrial use UTP is frowned upon. Just as most industrial networks recommend the use of shielded cable to minimise the number of transmission errors due to data corruption caused by EMI (electromagnetic interference), an industrial Ethernet can use shielded cable (STP) too.

Shielded cables can, in any network, sometimes cause more problems than they solve. The shield can function as an antenna, and pick up EMI even when it is properly grounded. The currents induced in the cable create identical but inverted currents in the cable’s signal wires. As long as these two currents are exactly each others inverse they cancel each other out, but every discontinuity in the shield or the signal wires causes a small asymmetry, which in turn results in a small negative influence on the signal quality.

Another disadvantage of STP is the attenuation of signals at higher frequencies. The possible presence of shielded cable must have been taken into account during the development of electronic circuits. The effect of a shield itself depends on the material used, thickness, type of EMI, frequency of the EMI, the distance between the cable and the EMI source, discontinuities or discontinuities in the shields, and the grounding method. In all cases the bending radius of a cable must be taken into account, to prevent cracks (discontinuities) in the shield.

There are several types of shielded cables for Ethernet. The cheapest cables only have a braided metal shield and are officially called “STP” (Shielded Twisted Pair). More expensive cables have a metal foil around the signal wires, and are called “FTP” (Foiled Twisted Pair). The most expensive cables have a combination of the two methods “SFTP” (Shielded Foiled Twisted Pair), and “PIMF” (Pair in Metal Foil) which has separate shielding around each signal pair.

Not only are the cables important for a network using shielded cables, but other components are as well, to assure that the shielding is really effective. The connectors, enclosures, housing
and equipment also need to be properly designed and connected. High-frequency signals require the shield to be earthed at both ends. However this may cause potential equalisation currents to flow through the shield, which may in turn degrade the signal quality, or (worse) damage the network or the connected equipment. The physical location of the earth terminals is also important: if they are too far away it will not function as expected. In principle each network is unique; although it is possible to give general rules for the design of networks one must always be prepared for surprises.

In the area of EMI-emissions there are large differences between STP, FTP, SFTP and UTP cables. Contrary to first impressions, UTP cabled networks sometimes comply better with European regulations than STP cabled networks (based on information from www.evolution.nl).

![Industrial Ethernet cable](source: Phoenix Contact).

Standard UTP cables are not resistant against temperatures that are too low; the insulation crumbles, which in due course may cause short-circuits. Several suppliers therefore offer industrial-quality network cables either in copper or optical fibre (i.e., Siemens, Hirschmann, Phoenix Contact), sometimes also with proprietary additions that allow for longer distances to be covered than specified by Ethernet.

### 3.2 Power supply for I/O

In an office environment it is not very difficult to find a 230V socket outlet. In an industrial environment this is not always easy; in many cases different supply voltages are used for equipment (i.e., 24V). Many existing industrial networks allow for the devices to be supplied via the network itself, for example:

- DeviceNet uses 2 extra wires in the network cable (24V/8A).
- AS-Interface modulates the network signal on top of a 30V power supply; so only two wires in a cable are needed.
- Foundation Fieldbus and Profibus/PA modulate the network signal on top of a 15V power supply (to comply with intrinsic safety regulations).

With Ethernet, it is usually impossible to supply power to devices via the network itself. This is a major disadvantage for Ethernet in comparison with existing industrial networks, as many users have just got accustomed to the easy of use of supplying power + network via the same cable.
Some suppliers (i.e., Cisco) have developed their own (proprietary) solutions to supply power via Ethernet, mainly to support IP-telephony applications. These various developments are now integrated in the IEEE 802.3af committee and from 2004 on the “Power over Ethernet” (PoE) functionality is officially possible. The company PowerDsine (www.powerdsine.com) is one of the first companies to have launched products supporting PoE.

For PoE special “power hubs” are needed. These are placed between a standard hub / switch, and the device to be connected. The power hub (figure 3-2) is transparent for the network signal pairs, but can supply 48V / 13W to any device. Why 48V? This is related to the initial application for which PoE was developed: IP-telephony. A ‘standard’ telephone also receives 48V from the telephone exchange, so this voltage was chosen for compatibility reasons for IP-telephony too. If 13W is not sufficient, a device must have a local power supply. PowerDsine has proprietary products that are capable of providing more power; perhaps this will become a future extension to IEEE 802.3af.

Figure 3-2: Electrical schematics of a PoE “power hub”. The version of PoE is shown in which the power supply is offered via the normally unused pins 4/5 and 7/8.

The 48V is offered on the 4 unused pins of the RJ45 connector. In order to prevent equipment damage to equipment that does not expect 48V, the device must, after connection, consume current according to a certain profile. If this does not happen, the power hub will disconnect the power supply to that port, preventing further damage.

A second version of PoE offers power supply on the same wires as those already in use for network signals. This is necessary for using PoE in combination with gigabit Ethernet, as it uses 8 wires and all the pins on the RJ45 connector.

The advantage of a power hub is the saving on wiring for a power supply running in parallel to the network. This is irrelevant for many applications, but for applications with devices dispersed over large areas without 230V in every corner, PoE can be very interesting. Additionally, using a
power hubs means that there is only one power supply for many devices; using a back-up power supply (UPS) for all devices simultaneously is now also very easy.

Power hubs are of course only possible when using copper cabling, but fibre optic users don’t have to despair. The US company Photonic has developed the technology to generate 6V / 10 mA from a single fibre optic connection. Although this isn’t in any way related to Ethernet, it indicates what is about to become a reality in a few years from now.

### 3.3 Wiring topologies

Ethernet allows a network to be cabled according to several topologies: the so-called “bus”, “star”, “line” or “ring”. Each has its own advantages and disadvantages.

**Bus topology**

Many industrial networks use the “bus” topology is used. In principle a bus is a single, long “trunk” cable, on which all devices are connected, either directly, or via a short “stub” or “spur” cable (figure 3-3). Only one device may be connected per stub; multiple devices are not permitted on a single stub. The bus structure is a simple, flexible and cheap way of wiring a network, and is therefore quite popular.

*Figure 3-3: A “bus” consists of a main “trunk” cable, to which all devices are connected via short “stubs”.*

Usually a “termination resistor” has to be connected to both ends of the trunk. Failing to do this correctly is one of the largest sources of errors in bus wired networks. The maximum length of the network is determined by the bit rate: the higher the bit rate, the shorter the trunk and stubs must be.

It is ironic that Ethernet supports the bus structure, but it only does so in the 10 Mbit/s coax versions (10Base2 and 10Base5), which are seldom used anymore. Despite the easy wiring I do not recommend the use of coax; it is not possible to upgrade to 100 Mbit/s, and it is difficult to get equipment that still has a coax interface.

**Star topology**

The twisted pair versions of Ethernet (10BaseT and 100BaseTx) do not allow a bus topology to be used; it would not even work because of the electrical interface with its separate receive and transmit channels. A “hub” or “switch” is needed; each device is connected (with its own cable) to the hub / switch. The maximum number of devices is limited by the number of ports on the hub / switch, but if there are too few ports hub and switches may be connected together, allowing for larger networks to be built.
A star topology is not very practical with long linear systems, as a lot of wiring is needed (figure 3-5). Additionally it is difficult to build modular systems, because new wires have to be added for each new device. The only solution is to use not one large hub / switch, but many smaller ones (with 4 or 8 ports each), but this is quite expensive. Support for a bus structure is much in demand; innovations by Profinet, EtherCat and Sercos (see chapter 6) will allow this.

**Figure 3-5:** A star topology is not practical in linear systems. Of course the Ethernet coax-versions could be used, but as these only support 10 Mbit/s this is not a future-proof solution.
Combination of bus and star topology
Because star topology is not always an efficient way of wiring, an alternative solution is sometimes implemented. This is two-level network architecture: a star-wired Ethernet at the highest level, and another industrial network wired according the bus topology at the lowest level. This gives a system in which the two types of networks each have their own most optimal wiring structure. To connect the two types of networks a “coupler”, “proxy”, “linking device” or “gateway” is used. It handles the electrical conversion between the two types of networks, the speed conversion, and the protocol conversion (figure 3-6).

![Figure 3-6: Two-level network architecture consisting of Ethernet at the highest level, an industrial network at the lowest level, and a coupler / proxy / linking device / gateway to connect them together.](image)

Line topology
Because of the disadvantages of star topology, Profinet has chosen a different solution: each device on the network will have its own (internal or external) switch (figure 3-7).

![Figure 3-7: As of Profinet V3 each device will have its own internal switch. This allows a network wired as a line instead of a star.](image)
The switch in each device can have either 2 or 4 ports. This allows a line-wired network, simply by connecting devices to each other. Devices with the 4-port switch have the capability to make branches. Any possible wiring structure may be made as long as any network message does not have to traverse more than 20 switches (due to the internal processing delays of each switch). Thanks to the internal switches, a Profinet network does not need external switches; this gives it a financial advantage over other industrial Ethernet systems. Another advantage of internal switches is that they are specially made for Profinet, and can thus also recognize Profinet-traffic and favour it above all other Ethernet protocols running concurrently. This improves the real-time characteristics of Profinet.

A disadvantage of internal switches is that removal of a device causes the network to be interrupted; this also happens when a device is switched off or its power-supply no longer functions. If this is not wanted, a star-wired Profinet may be a better solution.

The Profinet internal switch is a way of using industrial Ethernet that will undoubtedly be copied by other industrial Ethernet protocols. It takes away the disadvantages of the star topology, and makes it more competitive against the existing remote I/O networks. Two other systems that have already taken identical steps are EtherCat and Sercos-III.

### Ring topology

Ring-wired networks are possible for industrial Ethernet. Only switches can be used, but network messages traversing ‘endless loops’ must be avoided. Special network protocols, i.e. STP (Spanning Tree Protocol) have been developed to detect such loops and break them “in software”. The advantage of a ring is the redundancy it offers. STP can automatically reconfigure the network cabling problems, network interruptions, switch failure etc. that occur, in such a way that all devices remain accessible. More information about STP is given in chapter 5.

#### 3.3 Is Ethernet real-time or not?

The question of the real-time characteristics of Ethernet has been the subject of fierce discussions for many years (and will be for many years to come), because the answer to the question can be either “yes” or “no”. Additional confusion is created because some confuse ‘speed’, ‘real time’ and ‘determinism’ with each other, three totally different things.

Whether Ethernet is ‘fast’ or ‘slow’, is of course dependent on the requirements of the application. Some experts go as far as stating that “Ethernet is not fit for real-time systems because it can not be used in all real-time applications”, very often misquoted as “Ethernet is not fit for real-time systems”. Personally I consider this to be a harsh judgement; like any other network, it is always possible to find applications for which Ethernet is fast enough, but there are also applications for which Ethernet is much too slow, but this is not reason enough to state that it isn’t good for anything.

In order to prevent continuous discussions about “speed”, the IAONA committee for real-time systems has created four “real-time classes”, each with their own characteristics (table 3-1).

<table>
<thead>
<tr>
<th>Class</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. jitter</td>
<td>&gt; 1 ms</td>
<td>0.1 .. 3 ms</td>
<td>10 .. 400 µs</td>
<td>0.5 .. 15 µs</td>
</tr>
<tr>
<td>Hardware</td>
<td>Standard</td>
<td>Standard</td>
<td>Standard</td>
<td>Special</td>
</tr>
<tr>
<td>SW Stack</td>
<td>Standard</td>
<td>Modified</td>
<td>Special</td>
<td>Special</td>
</tr>
<tr>
<td>Protocol</td>
<td>Standard</td>
<td>Standard</td>
<td>Special</td>
<td>Special</td>
</tr>
</tbody>
</table>

Table 3-1: The IAONA real-time classes

Class 1 systems can be built with standard Ethernet hardware, standard software, and with existing protocols (such as TCP/IP). Class 2 systems require special software support, in the form of a real-time kernel or a real-time extension to Windows. Class 3 systems still use standard hardware, but require specially adapted network protocols and supporting software. Class 4 systems require all of this too, but also special hardware in the form of Ethernet interface cards.
(like the built-in switches for Profinet V3), special switches (to support IEEE 1588), or software support in switches (IEEE 802.1p).

**10 or 100 Mbit/s?**
The speed of Ethernet is quite high; even at its slowest speed of 10 Mbit/s it looks fast enough when compared with most current industrial networks, for example Foundation Fieldbus H1 at 31.25 Kbit/s, Interbus at 500 Kbit/s and CAN at 1 Mbit/s. An Ethernet at 100 Mbit/s then seems fast enough for the most extreme application. However, this is not necessarily so. As on any network, the bit rate is only a rough indicator of network speed, because the overhead of the network protocol and the software is not taken into account. When we calculate the real speed of a 10 Mbit/s Ethernet, taking into account all sorts of overhead and a realistic amount of software delays, we can easily conclude that the modern industrial networks like CAN and Profinbus/DP can compete very well. This is discussed in more detail in chapter 4.

When using a master/slave protocol on Ethernet, the effective speed can get even lower when using a switch – it must have received a network message in full before it is sent on to its destination. The total transmission time needed for a network message is thus double the amount expected.

The effective Ethernet speed is also heavily influenced by software. The performance calculations shown in chapter 4 assume 0% software overhead. Existing industrial networks (like Interbus, CAN, and Profinbus/DP) can have 0% overhead because the network protocol can be completely executed in hardware. With Ethernet this is not yet possible; a processor running the protocols for the higher-level OSI-layers (3...7) is always needed. Depending on the speed of the processor (not always a 4 GHz Pentium) it possible that the software overhead is the major component determining a network’s performance. This often comes as a surprise to users upgrading their network from 10 to 100 Mbit/s; there may not be any speed increase at all.

**3.4 Is Ethernet deterministic or not?**
The meaning of the word “determinism” is, just like “real time”, understood by very few. Many (non-Ethernet) suppliers simply state “Ethernet is not deterministic”, and without further knowledge about what determinism is, it looks as though Ethernet is missing something very fundamental, and that it is “thus” not suited for industrial applications that require a network with real-time capabilities. I have heard many users dismiss Ethernet for industrial use for the sole reason of its (alleged) non-determinism.

When a network is deterministic, it is simply nothing more than being capable of guaranteeing in advance that the time needed to deliver a message at its destination never exceeds (guaranteed!) a certain maximum time. As on any network, this not only depends on the bit rate and the use of hubs or switches, but also on the network load generated by the other devices at the moment a message must be sent. If this network load is predictable, the maximum time needed for delivery of a message can also be predicted.

If a hub is used, there is always a possibility that two (or more) devices start the transmission of a network message at the same time; Ethernet has no central coordination algorithm. Of course it is not possible to have two transmissions in parallel; a so-called “collision” occurs which is easily detected because of the unusual voltage level on the cable. All devices must now stop their transmission, wait a random length of time, and then start again. Should a new collision occur, this algorithm simply repeats itself. In the rare event that 16 collisions occur in a row, the Ethernet interface will stop sending the network message, and an error will be reported to higher layer network protocols. It is then up to them to handle the error of the non-send message (this is the duty of TCP, if TCP/IP is used).

The random waiting time is chosen from an interval that doubles in size after each collision (1, 3, 7, 15, etc. but with a maximum of 1023) multiplied by the slot time of 51.2 microseconds (10
Mbit/s) or 5.1 microseconds (100 Mbits/s). The possibility that two devices chose exactly the same random waiting time thus decreases very quickly. After the first collision (interval 0..1) there is a 50% chance that two devices chose the same random waiting time, and then a new collision occurs. The interval then doubles to 0..3, when there is a 25% chance that both chose the same random waiting time, and a 75% chance that a different waiting time is chosen, increasing to 87.5% after the third collision, etc. The worst-case situation is that the maximum waiting time is chosen for every collision. Although unlikely, such worst-case scenarios must be used for determining the deterministic and real-time qualities of Ethernet.

<table>
<thead>
<tr>
<th>Collision</th>
<th>Interval-boundaries</th>
<th>Max. waiting time at 10 Mbit/s (msec)</th>
<th>Max. cumulative waiting time at 10 Mbit/s (msec)</th>
<th>Max. cumulative waiting time at 100 Mbit/s (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0..1</td>
<td>0.0512</td>
<td>0.0512</td>
<td>0.0051</td>
</tr>
<tr>
<td>2</td>
<td>0..3</td>
<td>0.1536</td>
<td>0.2048</td>
<td>0.0204</td>
</tr>
<tr>
<td>3</td>
<td>0..7</td>
<td>0.3584</td>
<td>0.5632</td>
<td>0.056</td>
</tr>
<tr>
<td>4</td>
<td>0..15</td>
<td>0.7680</td>
<td>1.331</td>
<td>0.133</td>
</tr>
<tr>
<td>5</td>
<td>0..31</td>
<td>1.587</td>
<td>2.918</td>
<td>0.291</td>
</tr>
<tr>
<td>6</td>
<td>0..63</td>
<td>3.226</td>
<td>6.144</td>
<td>0.614</td>
</tr>
<tr>
<td>7</td>
<td>0..127</td>
<td>6.502</td>
<td>12.64</td>
<td>1.264</td>
</tr>
<tr>
<td>8</td>
<td>0..255</td>
<td>13.05</td>
<td>25.70</td>
<td>2.570</td>
</tr>
<tr>
<td>9</td>
<td>0..511</td>
<td>26.16</td>
<td>51.86</td>
<td>5.186</td>
</tr>
<tr>
<td>10</td>
<td>0..1023</td>
<td>52.38</td>
<td>104.2</td>
<td>10.42</td>
</tr>
<tr>
<td>11</td>
<td>0..1023</td>
<td>52.38</td>
<td>156.6</td>
<td>15.66</td>
</tr>
<tr>
<td>12</td>
<td>0..1023</td>
<td>52.38</td>
<td>209.0</td>
<td>20.90</td>
</tr>
<tr>
<td>13</td>
<td>0..1023</td>
<td>52.38</td>
<td>261.4</td>
<td>26.14</td>
</tr>
<tr>
<td>14</td>
<td>0..1023</td>
<td>52.38</td>
<td>313.8</td>
<td>31.38</td>
</tr>
<tr>
<td>15</td>
<td>0..1023</td>
<td>52.38</td>
<td>366.1</td>
<td>36.1</td>
</tr>
<tr>
<td>16</td>
<td>0..1023</td>
<td>52.38</td>
<td>418.5</td>
<td>41.85</td>
</tr>
</tbody>
</table>

Table 3-2: Maximum waiting time per collision, and the cumulative (worst-case) waiting time after ‘n’ collisions.

Table 3-2 shows the interval and the maximum waiting time for a consecutive series of collisions. As can be easily seen, there can be quite long waiting times on a 10 Mbit/s network; for example after 10 collisions have occurred, there is a 1/1023 chance that the wait is 52.38 milliseconds. But before having had 10 collisions, the waiting times for the first nine collisions must also be taken into account. Suppose that the worst-case waiting time has been chosen every time, 104.2 milliseconds have already passed. The likelihood that this occurs is of course very small, but on average the wait will be half the amount of the worst-case time. But even then a considerable waiting time can occur. When using a 100 Mbit/s the collisions will be resolved 10 times as fast, but for modern applications even 10 milliseconds is "eternity".

Usually the unpredictability of the waiting time is the main reason for declaring that Ethernet is not deterministic and thus unfit for industrial applications. The question is: how often does Ethernet violate the timing requirements for an application? It is just like driving a car: there is a possibility that one gets involved in a traffic accident (a real “collision”) with fatal consequences. Usually this does not deter us from driving. Why? Because the likelihood that this happens is very small, and so we take the risk. With Ethernet, a similar approach can be taken – OK, it may not be deterministic, but is it deterministic enough for my application? What are the consequences if the delivery of a message is delayed too much once per year? Or 10 years? Or 1000 years?

<table>
<thead>
<tr>
<th>Network speed (Mbit/s)</th>
<th>Message size (bytes)</th>
<th>Message rate (1/sec)</th>
<th>Deadline (max ms)</th>
<th>Lifetime (year)</th>
<th>Network load</th>
<th>Chance of success</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>128</td>
<td>1000</td>
<td>3</td>
<td>5</td>
<td>1%</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>128</td>
<td>1000</td>
<td>2</td>
<td>5</td>
<td>1%</td>
<td>0.999999983</td>
</tr>
</tbody>
</table>
Table 3-3: The RTI spreadsheet determines what the likelihood is for a number of scenarios that the given deadline for a message delivery is exceeded, given the network load, the application deadline, and the number of years that this deadline may not be exceeded.

The American company Real Time Innovations (www.rti.com) has developed a spreadsheet (table 3-3) that determines the chance that a certain deadline for a message delivery will be exceeded, given the network load, the application deadline, and the number of years that this deadline may not be exceeded. An example: given a 10 Mbit/s Ethernet, on which 1 Kbyte of data is sent 500 times per second, with a deadline of 8 milliseconds not to be exceeded for 5 years; what is the chance that this deadline is never exceeded? Answer: only 35%. This is probably not acceptable. Optimising the application program so that only 250 messages of 1 Kbyte data are sent per second increases the chance of success: 92.6%. With the RTI spreadsheet it is easy to calculate a number of scenarios, and assess the risks. Frank Fuller takes another approach in his book (see chapter 9); he simulates a network via a special software package.

Collision prevention
All calculations done in the previous section assume a way of using the network in which there is no coordination between devices; any device is allowed to send a network message at any moment. This is typical for office networks, where PCs are used as individual devices; they do not cooperate with each other. A network load that is too high generates so many collisions making an Ethernet unusable, as little productive works gets done. Which network load brings an Ethernet to its saturation point is not clear, and there are many myths around: according to one expert any load above 8% is dangerous, the next expert allows a load of 50%, and there are also experts who claim a 90% load should be no problem at all (read chapter 19 of Spurgeon’s book, see chapter 9).

Regardless of which expert is right, there are other methods for creating a workable Ethernet: get rid of the collisions. Both hardware and software (= network protocol) solutions exist:

- Don’t use hubs; use switches instead. Each device has its own transmission channel to the switch. Because there is always only one device per transmission channel, collisions cannot occur.
- Keep the network load as low as reasonably possible; should a collision occur (with a very small chance), it can be resolved very quickly. This limits the total delay of the collision handling. Hima’s “Safe Ethernet” protocol works this way. Having a lightly loaded network is also beneficial in case of plant emergencies which increase traffic; if the network becomes too heavy loaded the actual throughput may decrease just when it is most needed.
- Use a master/slave protocol; either the master, or exactly one of the slaves may have a transmission pending. As there is always only one device active, collisions can never occur.
- Use a token-bus protocol; only one of the devices has possession of the token, and is allowed to transmit.
- Use a time-triggered protocol. Each device will be assigned its own time slot during a (continuously repeating) network cycle, and only during that time slot is the device allowed to transmit.

In practice, it is always advisable to use a switch, which is then used with one of the protocol variants. See chapter 6 for a description of modern network protocols.

Be careful when using TCP/IP; even when implementing a master/slave protocol on top of TCP/IP, there is still a small chance of collisions. This is caused by the way TCP/IP works internally: autonomous behaviour that starts transmission of messages even when the application is not active.

3.5 The network connector

UTP-based Ethernets always use the “RJ45” (Registered Jack) style connector. It is a well-designed connector, good for use in the office and at home, but it lacks certain qualities needed for industrial use:

- No tension relief;
- Plastic of connector easily breaks;
- Connector can move in three dimensions while plugged in;
- Not resistant against vibration;
- Not closed to dust or splash water;
- Not resistant against chemicals;
- Etc.

The resistance against vibrations was the subject of an investigation by Rockwell. Due to the permanent movement of the RJ45 connector, the gold layer wears off, exposing the underlying metal to oxidation and subsequent degradation of the signal quality.

Given the disadvantages of the existing RJ45 connector and the need for an industrial-quality connector for Ethernet, and the absence of a company large enough to force a ‘standard’ on the market, there are now 10 different industrial Ethernet connectors in existence. The ideas behind these connectors are based on:

- Usage of an existing connector for Ethernet (sub-D, M12), or
- Development of an entirely new connector, or
- Modification to RJ45 to solve its deficiencies (figure 3-8).

The new industrial connectors or the modified RJ45 connectors are not as cheap as the RJ45 (small market!), and many vendors continue using the normal RJ45. In many cases this connector performs very well, as experience of the last 10 years has shown.
The well-known M12 connector seems to be becoming one of the more popular industrial Ethernet connectors, due to its small size and efficient use. A special M12 variant called “D” has been developed for industrial Ethernet, which with its coded plug prevents connection errors between Ethernet cables and sensor/actuator cables.

The RJ45 compatible connectors all use a standard RJ45, extended with a sturdier housing, tension relief, IP65-sealing, etc. Such connectors have been developed by Bulgin, Phoenix Contact, Harting, Woodhead, Pulse, Siemon, Rockwell, Siemens and Yamaichi. Some connectors (i.e., Harting’s) even have special connector pins to allow for provision of power supply. This is different than specified in PoE (see section 3.2), but allows larger currents to be used.

For users, the RJ45 compatible connectors have the advantage that cables with the standard RJ45 can still be used. For the supplier this has the advantage that two different classes of users can be satisfied with one connector.

The near future
The large diversity of industrial Ethernet connectors is not a prime example of good standardisation, despite the fact that they are all standardised in IEC 61076-3-101. So far there has not been much uproar, because the new connectors were not all used in products. This is about to change, and I assume more complaints will follow. The lack of standardisation can make wiring industrial Ethernet more expensive than is necessary. Undoubtedly most new connectors will disappear from the market again; it is just a matter of time.

3.6 Using special protocols
The extended TCP/IP family of protocols is very often used in office automation. TCP (Transmission Control Protocol) and IP (Internet Protocol), combined with UDP (User Datagram Protocol) are the foundation stones for an extended set of applications:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TELNET</td>
<td>Remote Terminal</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>NFS</td>
<td>Networked File System (disk)</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transport Protocol (html)</td>
</tr>
<tr>
<td>POP</td>
<td>Post Office Protocol (email)</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>BOOTP</td>
<td>Bootstrap Protocol</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-Time Protocol</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
</tbody>
</table>
The experienced Internet user will recognise these protocols as being present on both Windows and Linux. Despite their usefulness in office work, these applications are not suitable for communication between controllers in industrial applications. Of course, a PLC can send an email to another PLC asking it to perform a certain task. Technically this is no problem, but practically it is not a workable solution. Unfortunately there is no standard protocol for industrial use. This does not mean such protocols do not exist; many suppliers have developed their own solutions. But since these were all proprietary solutions, communication between devices of different suppliers was not possible, despite everyone using TCP/IP. The resulting customer lock-in was seen as a disadvantage of Ethernet, also preventing its large-scale use. Now that there is more agreement about the use of “open” protocols, the advancement of Ethernet has started.

**Current TCP, UDP and IP developments**

Many new industrial Ethernet protocols use TCP/IP, but only for that functionality that doesn’t require high speed I/O or real-time behaviour, such as during initialisation, network management, configuration, and diagnostics functions. TCP/IP is not practical for high speed remote I/O; the existing industrial networks have much better speed-optimised protocols that can easily outperform TCP/IP. This is caused by an optimisation in the run-time behaviour of TCP/IP that makes it very good for office use where large amounts of data have to be transmitted efficiently. This optimisation has a drawback: transport of small amounts of data (as is typical for I/O) is handled inefficiently. The new industrial Ethernet protocols thus want to bypass TCP/IP for high speed I/O, which is simply done by developing new protocols with very little overhead and a good match with the way Ethernet works internally (TCP/IP can not do this, because it is network independent and has to be able to run on any underlying network).

The ‘loss’ of TCP is not really a problem, as almost all applications do not require the typical TCP functionality for traversing Internet and intranets. TCP’s little brother, UDP can handle I/O much more efficiently, because UDP is a very simple protocol (specification only 4 pages), and thus also very fast. It has two other advantages as well: first, UDP is message-oriented (TCP is byte-oriented), so all the bytes of a network message are delivered together (or none at all); second, UDP support “broadcasts” where one network message is delivered to all devices on a network simultaneously. Disadvantages of UDP are: there is no guarantee that data is delivered on the destination device without errors; and that data can arrive in a different order than sent. Both disadvantages can be countered with simple measures to be employed by the network protocol.

Despite the much higher speed of UDP it is still considered “too slow”. Protocols like IDA, Profinet and Powerlink have developed their own remote I/O handling, which completely bypasses TCP, IP and UDP. In fact, these protocols almost directly control Ethernet interface hardware. A major speed increase is the result. Also, it is possible to have much better synchronisation between devices, because the uncontrollable software overhead of TCP, IP and UDP is no longer significant. A disadvantage is that co-existence of the new protocols with other Ethernet-users may become difficult as the new protocols assume that they are in full control of all Ethernet activity. The open character of Ethernet is degraded by these developments.

**Tunnelling**

A completely different use of TCP/IP is to send its messages via another, non-Ethernet, network, for example an industrial network like Profibus. This technique is called “tunnelling” and in fact has nothing to do with Ethernet; it is just ‘another wire’ over which a group of bytes is sent which together make up a TCP/IP message. For the fieldbus, it is just a group of bytes; it doesn’t know the data is actually another protocol. Any fieldbus that is capable of sending a reasonable amount of bytes can implement tunneling; the difficulty with many industrial networks is usually that they can send only small amounts of data (i.e., Interbus: 2, 4 or 8 bytes; AS-Interface: 4 bits), too small to hold one complete TCP/IP message. The solution that is usu-
ally employed breaks up the TCP/IP message into several smaller parts, which are transmitted one by one, and then reassembled at the receiving device. For the sending device and for the receiving device the fact that a fieldbus has been involved in the transmission of the message is then completely invisible.

When TCP/IP tunnelling is supported on a fieldbus, the whole family of TCP/IP application protocols can be used as well. Many devices already have embedded (built-in) web servers; it is possible to retrieve web pages from a device with any standard browser. This is not only useful for remote diagnostics and maintenance, but also for configuring equipment – no special software packages are necessary. Another use of a TCP/IP application protocol is email – a device can send an email when its internal diagnostic functions have found an error, for example or to warn that a failure is imminent. Note that the speed of tunnelling TCP/IP application protocols over a fieldbus is usually quite slow – devices do not have a 3 GHz Pentium on board but often only a simple processor with a clock speed of 50 MHz or less, and then all TCP/IP protocol overhead must be handled concurrently with the device’s main function (i.e., I/O, control loops, etc.).

3.7 Cost
Due to its immense popularity Ethernet-hardware is very cheap; network interface cards for a PC do not have to cost more than several pounds, and even this may not be needed as more and more PC motherboards have Ethernet integrated.

No other network, industrial or otherwise, can compete with this. The low CAPEX Ethernet prices are often used as argument against existing industrial networks, whose network interface cards on first sight can be 10 to 100 times as expensive. But this is not an honest comparison; the total cost of supporting a network interface on any processor (PC or PLC) must be taken into account. Adding a network interface card to a PC is only the last piece of a larger puzzle:

- PC motherboard support + connector + housing
- Using the PC’s power supply
- Development of the network drivers (supplier)
- Development of the network protocols (TCP/IP)
- Operating system support (Windows) for running network protocols
- Using part of the hard disk for storage of software files
- Using RAM memory for run-time execution of software
- Pentium processor to run the network protocols.

Much of the cost needed to support a functioning network interface in a PC is paid for invisibly, i.e. the power supply, hard disk and memory and motherboard are hidden in the purchase price of the PC, the protocol support as part of Windows, etc. When a network interface card is developed for a PLC, the total cost of the card becomes more visible. And industrial controller Ethernet interface cards thus appear to be much more expensive.
SPEED

As with most industrial networks, speed is of prime importance to Ethernet. At first sight, Ethernet seems to be very luxuriously endowed, giving bit rates of 10 or 100 Mbit/s, which is quite high in comparison to most existing industrial networks operating at speeds of 1 or 2 Mbit/s and sometimes substantially less than this. But as is the case on any network, a bit rate is just a ‘raw’ indicator of speed, and not a ‘net’ indicator. The difference between these two is quite substantial for Ethernet, due to the large overhead it incurs when transmitting small amounts of data.

4.1 Overhead per network message
Ethernet has substantial overhead: each network message has a minimum size of 84 bytes, of which 46 are reserved for application data. But if the application has only a few bytes of data to transmit, for example when using I/O systems (1 digital channel = 1 bit, 1 analogue channel = 16 bits), this results in a substantial overhead. Figure 4-1 gives the figures for the efficiency of Ethernet with various amounts of data. Note that with large quantities of data Ethernet is very efficient. This is not surprising: Ethernet having been originally developed for office applications, it has been optimised for the most common usage: sending large quantities of data.

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Amount of data (bytes)</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bits / 8 bytes</td>
<td>1</td>
<td>1,2%</td>
</tr>
<tr>
<td>48 bits / 6 bytes</td>
<td>2</td>
<td>2,4%</td>
</tr>
<tr>
<td>48 bits / 6 bytes</td>
<td>5</td>
<td>6,0%</td>
</tr>
<tr>
<td>16 bits / 2 bytes</td>
<td>10</td>
<td>11,9%</td>
</tr>
<tr>
<td>Data</td>
<td>20</td>
<td>23,8%</td>
</tr>
<tr>
<td>Minimum 368 bits / 46 bytes</td>
<td>50</td>
<td>56,8%</td>
</tr>
<tr>
<td>CRC</td>
<td>100</td>
<td>72,4%</td>
</tr>
<tr>
<td>32 bits / 4 bytes</td>
<td>500</td>
<td>92,9%</td>
</tr>
<tr>
<td>Silence</td>
<td>1000</td>
<td>96,3%</td>
</tr>
<tr>
<td>96 bits / 12 bytes</td>
<td>1500</td>
<td>97,5%</td>
</tr>
</tbody>
</table>

Figure 4-1: The fixed structure of each Ethernet messages (left) makes very inefficient use of the network when sending small quantities of data, something that is quite common in industrial applications.
4.2 Comparison of speed with CAN

When we compare the speed of a 10 Mbit/s Ethernet with a 1 Mbit/s CAN, it may look at first sight as if CAN will be the loser, but this is not so—it depends how both networks are used. If they are used in a remote I/O system, there is not much real performance difference (table 4-1).

The first example is the transmission of 2 bytes of data. Each CAN message has a 47 bit overhead, and the total amount of data transmitted is then 47+2*8=63. At a speed of 1 Mbit/s this takes 63 µsec. The same 2 bytes on Ethernet take 672 bits of transmission time, and this takes (at 10 Mbit/s) 67 µsec. Surprisingly, CAN at 1 Mbit/s is faster than Ethernet at 10 Mbit/s. How is this possible? CAN has a much higher efficiency (25%) than Ethernet (2.3%), so its 10x lower bitrate is compensated by its more than 10x better efficiency.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>CAN</th>
<th>Ethernet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>1 Mbit/s</td>
<td>10 Mbit/s</td>
</tr>
<tr>
<td>Max. data</td>
<td>8 bytes</td>
<td>1500 bytes</td>
</tr>
<tr>
<td>Min. data</td>
<td>0 bytes</td>
<td>46 bytes</td>
</tr>
<tr>
<td>Overhead</td>
<td>47 bits</td>
<td>38 bytes</td>
</tr>
<tr>
<td>Sending 2 bytes</td>
<td>1 message needed</td>
<td>1 message needed</td>
</tr>
<tr>
<td>Bits to transmit</td>
<td>47 + 2*8 = 63</td>
<td>(46+38)*8 = 672</td>
</tr>
<tr>
<td>Transmission time</td>
<td>63 µsec</td>
<td>67 µsec</td>
</tr>
<tr>
<td>Efficiency</td>
<td>(2*8)/63 = 25.3%</td>
<td>(2*8)/672 = 2.3%</td>
</tr>
<tr>
<td>Sending 1500 bytes</td>
<td>188 messages needed</td>
<td>1 message needed</td>
</tr>
<tr>
<td>Bits to transmit</td>
<td>187*(8<em>8+47)+(4</em>8+47) = 20836</td>
<td>(1500+38)*8 = 12304</td>
</tr>
<tr>
<td>Transmission time</td>
<td>20.8 milliseconds</td>
<td>1.23 milliseconds</td>
</tr>
<tr>
<td>Efficiency</td>
<td>(1500*8)/20836 = 57.5%</td>
<td>(1500*8)/12304 = 97.5%</td>
</tr>
</tbody>
</table>

Table 4-1: Speed comparison between CAN at 1 Mbit/s (left) and Ethernet at 10 Mbit/s, in 2 scenarios.

The comparison becomes more interesting if we use Ethernet’s capability of working in full duplex mode. CAN cannot do this: each device can either send or receive network messages, but not simultaneously. But it is very interesting to have full-duplex capability especially when working with remote I/O, because reading inputs can be done concurrently with setting outputs. When using CAN, two messages need to be transmitted sequentially, taking twice as much time (2 * 63 = 126 µsec). For Ethernet, one would still need 67 µsec, making it a winner in this scenario.

A third scenario is the transmission of 1500 bytes of data. With CAN, this is not possible as each message can only contain 8 bytes. The 1500 bytes must then be split up into 187 CAN messages with 8 data bytes each, and a last message with 4 bytes of data. These 188 network messages in total take 20836 bits (20.8 milliseconds at 1 Mbit/s). With Ethernet, the 1500 bytes fit in a single message, taking 12304 bits (1.23 milliseconds). In this scenario Ethernet is the clear winner. Why? The CAN messages require 188 times 47 bits overhead. The efficiency of CAN has reached its limit (at 57.5%) but Ethernet operates at 97.5%. Also, even though CAN is 10x slower “raw”, it is 17x slower “net”.

From these scenarios it is clear that both CAN and Ethernet have an operating optimum; CAN when sending small amount of data, and Ethernet large amounts of data. This is not surprising; both networks were optimised deliberately for their respective application areas. Both networks function sub optimal in the reverse situation.

Identical comparisons can be made for all other industrial networks. It then becomes clear that most existing industrial networks can compete very well with 10 Mbit/s Ethernets. When using
100 Mbit/s, Ethernet of course always wins. However, the calculations shown above assume that there is never any software delay incurred in the processing of network messages.

4.3 Software delays

In any network, software delays due to processing incoming network messages may have a substantial influence on the total performance. In many industrial networks, the influence of software delays is sometimes reduced to zero when the protocol is fully implemented in hardware (i.e., for Profibus/DP, Interbus, CAN, etc.). For Ethernet, this is not yet possible; software is always needed in the form of a network driver, operating system, protocol stack, and application. For a 100 Mbit/s Ethernet this is especially important as its high speed sometimes makes transmission times of messages so small that the performance of the network is completely determined by software.

An example of this phenomenon is the Ethernet-based remote I/O system of a well-known supplier S. In this example we’ll assume a network with 6 remote I/O modules, each with 16 digital inputs (2 bytes of data) and 16 digital outputs (also 2 bytes of data). The protocol used is one of the master/slave type; a remote I/O module is sent messages with new values for its outputs, and it responds with another message containing the actual values of the digital inputs. This is a half-duplex method of communication. The half-duplex behaviour is determined by software; whether the Ethernet being used is half-duplex (with hubs) or full duplex (with switches) is not important. Theoretically, the total time needed to access all I/O modules (at 10 Mbit/s) is:

\[ 6 \times 2 (= \text{inputs and outputs}) \times 672 \text{ bits} / 10 \text{ Mbit/s} = 0.8 \text{ milliseconds}. \]

Faster is not possible, but because the software delay has not yet been taken into account, a slower performance is possible! For example, supplier S quotes a total time of 1.9 milliseconds. This is considerably more than the 0.8 millisecond total transmission time. The difference can only be the software delays incurred on the controller and on the 6 I/O modules, being \[ 1.9 - 0.8 = 1.1 \text{ milliseconds}. \]

If the 1.9 millisecond is not acceptable, it can be argued that a speed increase of Ethernet from 10 to 100 Mbit/s would bring a better performing network, hoping for a tenfold increase in I/O speed (0.19 millisecond). A simple calculation shows an entirely different result: the transmission time of the network messages indeed decreases by a factor of 10 (from 0.8 to 0.08 milliseconds), but the software delays remain at 1.1 millisecond. The total cycle time then becomes 1.2 millisecond \((1.1 + 0.08)\), which is only 1/3 faster and not 10 times.

Using Ethernet at 100 Mbit/s and at full capacity requires an optimal implementation of the network protocol. This is not achieved as a matter-of-course, it may be interesting to compare the real performance of different suppliers’ products, which is then not listed in commercial literature; the meaningless indication “this product works at 100 Mbit/s Ethernet” is usually only given.

As stated before, the existing industrial networks can handle the competition with a 10 Mbit/s Ethernet very well, and sometimes also with a 100 Mbit/s Ethernet. Figure 4-2 shows a speed comparison between Ethernet and several industrial networks. The X-axis displays the number of remote I/O modules, assumed to have 16 bits for inputs and 16 bits for outputs. The Y-axis displays the cycle time of the network: the time needed to read the current values of all inputs, and to set the outputs to their current known values.
Figure 4-2: Cycle times for a few well-known industrial networks, and Ethernet both at theoretical maximum speeds and the current implementation of supplier S.

Figure 4-2 also shows the cycle time of supplier S’s remote I/O product working at half-duplex, and the theoretical maximum speed for this product. Figure 4-3 zooms in on the lower part of figure 4-2, and the figures for a half-duplex and full-duplex network are also shown. In the end a 100 Mbit/s Ethernet always beats the existing industrial networks. Only Sercos-III is faster, but this is caused by several optimisations to the protocol, making it no longer a ‘real’ Ethernet (see chapter 6 for the technical details on Sercos-III).
4.4 Comparison between industrial Ethernet systems

Chapter 6 discusses various application protocols that have been especially designed for high-speed motion applications. These applications require the highest speed networks, and suppliers are currently waging a fierce struggle to squeeze the maximum out of Ethernet. Several different solutions have been developed, and this allows for an interesting comparison between the new application protocols. The underlying hardware is the same for all (100 Mbit/s Ethernet), so the real speed is determined by the application protocol and its implementation.

According to the Sercos users group the following cycle times are possible on a network with 12 axes and 12 bytes data per axis:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Network Speed</th>
<th>Cycle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sercos-III</td>
<td>100 Mbit/s</td>
<td>64 µsec</td>
</tr>
<tr>
<td>ProfiNet V3</td>
<td>100 Mbit/s</td>
<td>106 µsec</td>
</tr>
<tr>
<td>Powerlink V2</td>
<td>100 Mbit/s</td>
<td>205 µsec</td>
</tr>
</tbody>
</table>

The difference between Powerlink and ProfiNet can largely be explained by the fact that Powerlink uses a half-duplex Ethernet, making it twice as slow (this will change with Powerlink V3). The difference between Sercos and ProfiNet can be explained by the changes that have been made to Ethernet, allowing for less overhead.

In comparison, some cycle times for non-Ethernet based protocols are:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Network Speed</th>
<th>Cycle Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sercos-II</td>
<td>16 Mbit</td>
<td>195 µsec</td>
</tr>
<tr>
<td>Profibus DP</td>
<td>12 Mbit/s</td>
<td>1056 µsec</td>
</tr>
</tbody>
</table>

Figure 4-3: Enlargement of figure 4-2. Additionally, the cycle times for a 100 Mbit/s full-duplex Ethernet and the Sercos-III version of Ethernet are drawn.
This shows that an existing system such as Sercos-II with a 6x lower bitrate is still able to compete with 100 Mbit/s Powerlink, as Sercos-II has a protocol with very low overhead.

All figures are based on implementations available on the market at the end of 2004. It is expected that all the systems will come up with better figures, due to better optimised network protocols and implementations done in hardware (removing all software overhead). As the battle for the fastest motion network is not finished yet, I expect lots of new developments the coming years.
5 

HUBS AND SWITCHES

Hubs and switches are used in every modern Ethernet. From the outside there is little difference, and in many cases it does not matter at all whether hubs are used or switches are used. Because hubs and switches are essential to the functioning of an Ethernet, and over time the basic functionality of hubs and switches has been extended greatly, this chapter is completely devoted to the way hubs and switches work internally, and their differences.

5.1 Required functionality
The coax-based Ethernet versions do not require hubs or switches, because the coax cable is used for transmission and reception of network messages over the same physical wire. The twisted-pair and fibre-optic versions of Ethernet work differently, because they have separate transmit (tx) and receive (rx) wires in the network cable.

A very simple network with two devices can be made easily by connecting the tx-wires of one device to the rx wires of the other device. This can be done easily with a so-called “crossover” cable. It is a perfectly acceptable way of connecting two devices. But when a third device has to be added, this won’t work anymore- the tx wires of two devices have to be connected to the rx wire of the third device. Ultimately all the tx and rx wires of all the devices are connected together, including even a device’s own tx wires to its own rx wires. Of course this can’t work and it doesn’t. When using fibre-optics it simply isn’t possible to wire a network this way.

An intermediate solution would be to wire a network completely with point-to-point connections. In a network with 5 devices, each device would require four network interface modules, linking each device to all the other devices. The final result is a star-wired network, like a spider’s web. Although technically there is nothing wrong with this solution, it is financially unacceptable, especially when there are more than a few devices. Additionally, the wiring costs are considerable and its ability to be adapted to future extensions is problematic.
Ethernet offers a better solution. A “hub” is used as a network exchange, just like a telephone exchange. Every wired or wireless phone is connected to an exchange, rather than having separate wires to all the other phones in the world. Every conversation is first transmitted to the exchange, which is then responsible for sending the transmission further on to the dialled receiver. Ethernet works in exactly the same way: a device sends its network messages to the hub (figure 5.1), which forwards the message to the destination device, or possibly to another hub. How a hub this does is usually not of importance for the sending device; it does not even have to know how many hubs are in the communication path to the destination device.

5.2 Terminology
In due course two types of hubs came into being; we now not only have “hubs” but also “switches”. From a theoretical point of view, a switch is also a hub. The official names are:

- Switching hub, and
- Repeating hub.

In daily use this is shortened to:

- Switching hub becomes “switch”
- Repeating hub becomes “hub”.

In this publication we will follow popular usage.

5.3 Hub operation
The internal operation of a hub is very simple. The hub has a variable number of connection ports (8, 12, 16, 24, 32... depending on the supplier and type of hub) to which just as many devices can be connected, each with its own cable connected to one of the ports. There is only one way for a device to send a message to the network: send it to the hub first. The hub will then copy the messages to all the other (n-1) devices attached to it.
Figure 5-2: A hub simply transmits copies of each received message to all the devices connected to it.

Figure 5-2 shows a hub with 5 devices connected to it. Suppose that device C wants to send a network message to B. The hub will send a copy of the message to A, B, D and E simultaneously. This does not lead to problems, as A, D and E detect (with their MAC-filter) that the message is not meant for them, so the message will be ignored. The presence of a MAC-filter in every device allows a hub to work in this way. The sender of the message fills in the MAC-address of the destination, so only this device will accept the message and process it.

**Internal structure**
The internal structure of a hub is simple, as shown in figure 5-3. At any moment only one of the connected devices may have a transmission active; if not, a collision will occur. This means that a device may either receive or transmit, but not both simultaneously. This behaviour is called “half duplex”. It is identical to the way the older, coax-based Ethernet works.

**5.4 Switch operation**
A switch handles Ethernet messages different from a hub, made possible by having local intelligence. For every received Ethernet message, the switch extracts the destination MAC-address from it. The switch thus ‘knows’ to which device the message must be delivered. Because the switch “knows” to which connection port this device is attached, it can just transmit the message...
on this port, and do nothing on all other ports. Of course, broadcast messages are retransmitted on all ports.

Figure 5-4 shows a network with a switch and 5 connected devices. Suppose that device C wants to send a network message to B. The switch, knowing which port B is connected to, only re-transmits the network message on this port; devices A, D and E hear nothing. It wouldn't make much sense to send them the message, since (due to their MAC-filter) they would ignore the message anyway.

**Internal structure**
Figure 5-5 shows the internal structure of a switch with three ports in the form of a diagram. The “port selector” routes all network messages to the outgoing port. In figure 5-5 the port selector is shown as a switch, but in reality it is an electronic circuit.
Because all the other ports are not used, they are free for the transmission of other network messages that are destined for the other devices. It is thus possible to have multiple transmissions in parallel, with in principle one transmission per device. Each device may transmit simultaneously to the switch. This is called “full duplex”, in contrast to a hub’s “half duplex” behaviour (receive or transmit).

The full duplex capability is one of the major advantages of a switch as compared to a hub, as this greatly increases the bandwidth of the network. On a 10 Mbit/s network with three devices the total bandwidth is 3 * 2 * 10 = 60 Mbit/s. Please note that this is the capacity of the network as a whole, each individual device still has a 10 Mbit/s connection. If there is symmetric transfer of messages back and forth, the device can have 20 Mbit/s bandwidth; in practice this is usually difficult to achieve.

Please note that the full-duplex bandwidth doubling is only attainable when the network traffic is equally spread over all the devices and over the sent and received messages. When a supplier states that a switch doubles your network bandwidth, a statement is made that can only honestly be made when the traffic pattern of the application is known. If the application follows a master/slave structure, or a master/slave network protocol is used, a switch will not increase the bandwidth. This is because a master/slave protocol either transmits or receives (but not simultaneously), and at any moment only one slave may transmit. Network protocols that frequently use broadcasts also cause the switch advantage to be lost, as all messages must be sent to all devices – broadcasts make a switch behave like a hub.

**Port selector operation**

A port selector can function in two different ways:

In the process of receiving a network message, the switch waits until the destination MAC-address field is received. This happens very quickly, as this field is the first field of any message. The switch calculates the outgoing port, and starts transmitting that part of the message that has already been received. This is a very efficient way of handling messages, as only a very short delay is incurred and transmitting is done in parallel with receiving a message. Switches operating in this way are called “Cut-through” switches.

Or:

First, the complete network message must have been received. It is stored internally. Only then can the destination MAC-address field be extracted, and the switch calculates the outgoing port. The stored message is then transmitted. Switches operating in this way are called “Store and Forward” switches.

The disadvantage of cut-through switches is that they cannot detect errors in received messages, so any damaged network message is sent on. A normal switch will drop any damaged network message: why continue sending it on when the final destination device would ignore it anyway? Because the speed differences between store-and-forward switches and cut-through switches are now quite insignificant, cut-through switches have disappeared from the market. For industrial applications, where many network messages are small, the advantage of cut-through switches would be even less significant than in office equipment.

**Non-blocking switches**

At speeds of 100 Mbit/s and with dozens of devices connected to it a switch can have a lot to do. A network load that is too heavy may cause an overload on the switch, making it no longer capable of processing all received messages. The only way to handle an overload is to start ignoring messages. The total bandwidth of such a switch is thus not linear with the number of devices. The messages that were dropped have to be handled by higher level protocols (such as
TCP); the extra network traffic that this generates further increases the load on the network, and result in unpredictable message delivery.

Some suppliers sell switches that have such a high internal processing capacity that they can handle any possible network load from all connected devices. Such switches are called “non-blocking switches”.

**Internal buffering**

Even though a switch is designed to handle the parallel transmission of multiple network messages, a problem may occur when two (or more) devices want to send a network message to the same destination device. A switch can only send one network message to any device at a time; there is no parallelism possible at device level. A switch can handle this in two different ways:

- **Ignore**: a network message that arrives first for a certain device is transmitted normally. While this transmission is going on, all incoming messages for the same device are silently ignored. Higher-level protocols have to repair this.

- **Temporary storage**: a network message that arrives first for a certain device is transmitted normally. While this transmission is going on, all incoming messages for the same device are stored in internal memory in a first-in, first-out queue. The number of messages that can be stored depends on the internal memory capacity of the switch, which is vendor-specific. When the memory is full, incoming messages will be ignored, leading to the same behaviour as above. The advantage is that short peaks of network load do not cause message loss.

Ask the switch vendor about the internal processing strategy of his products.

**The switch table**

As described above, a switch must forward each network message it receives to the port on which the destination device is connected. But how does a switch know which port this is? The simplest way to take care of this is to have the user configure the switch, but this is not practical in larger networks or in networks that change regularly. Therefore, switches are always “self learning” (figure 5-6). This works as follows:

1) As soon as a switch receives a network message on a port, it knows from the sender MAC-address field who sent it. The switch stores the combination (port, MAC-address) in an internal “switch table” for future use. The size of the switch table is vendor-specific, but is usually at least 1000 entries.

2) Every network message also contains the destination MAC-address field. The switch looks in the switch table to see whether this MAC-address is already known, and which port this device is connected to.

3) If the MAC-address of the destination device is known, the switch forwards the network message to the port indicated in the switch table. If the MAC-address is not known, the network message is forwarded to all the switch’s ports (broadcast).

There is one exception to item 3. In larger networks it is possible that a hierarchy of hubs and switches is used, and that a message received in a switch has to be forwarded over the same port as the port on which it was received. This can happen when there has been a hub in the message path between the sending and receiving device. In such a case, the network message
has already reached its destination; the switch does not have to re-transmit it. If it did so, the destination device would receive the network message twice.

In the following examples we’ll show the operation of a switch on a network consisting of 5 devices, 2 hubs and 1 switch.

Initially, after power-up, the switch table is empty. A transmission from device 3 to device 2 will then force the switch to broadcast the message to all ports: in effect the switch behaves like a hub. It has to do this because the switch does not yet know which port device 2 is connected to. By forwarding the message to all ports, the switch ensures that the message will arrive on device 2.

Because both hubs are also receiving messages from the switch, they will forward the message to all their ports too. In the end all the devices (1, 2, 4 and 5) will have received a copy of the message. Because the message is only meant for device 2 (as indicated in the MAC destination-address field) only device 2 will act on it, and all the other devices will ignore the message (it is filtered out by their MAC-filters).

After having forwarded the message, the switch will now update its switch table. Because it sees that the sending MAC-address (of device 3) is connected to port 3, it can enter this information in the switch table for later use.
Figure 5-7: Device 3 sends a message to device 2. Because the switch table is empty, the switch forwards the message to all ports. The connected hubs do this too. In the end all the devices on this network will receive a copy of the message, but only device 2 will accept it for further processing.

When a little later the switch table is completely filled (it is now known which devices are connected to which ports) the processing of a message sent from device 3 to device 2 is done completely differently. The switch now knows (figure 5-8) that device 2 is connected directly or indirectly to port 1. The message is now only forwarded to port 1, and no longer to port 4. Of course, the hub still forwards the message to device 1 (which ignores it).
Figure 5-8: When the switch table is completely filled, a transmission from device 3 to 2 will no longer be forwarded to port 4, but only to port 1.

As a final example we'll discuss the transmission of a message from device 4 to device 5 (figure 5-9). The message will first be received on the hub, which sends the message on to device 5, but also to the switch. According to the switch, device 5 is connected to port 4, but the message has also been received on port 4. Because of this, the switch 'knows' that it does not make sense to forward the message: it is ignored. This prevents device 5 from receiving the message twice. This would cause a lot of problems for some protocols because they cannot filter out duplicates (TCP/IP does not have this problem).
Figure 5-9: A transmission from device 4 to device 5 will not be forwarded by the switch as both devices are directly or indirectly connected to the same port.

Size of the switch table
The switch table is usually large enough to store sufficient MAC-addresses for a reasonably sized network, usually 1000 MAC-addresses or more. However, nowhere is it specified that the switch table must have such a size. Depending on the internal processing capacity of the switch, a smaller switch table may have been implemented. When in doubt, ask the switch vendor. Usually the switch table size is given in the documentation, together with information about the management of the switch table (timers, pre-programmed settings, MAC-address lockout, etc.).

5.5 Advanced switch capabilities
All the functionality described in the previous section is just the standard functionality that is present in every modern switch. Some vendors have added much more functionality to their equipment, and this is described in this section.

L2 ... L7 switches
As described at the beginning of this chapter, the internal operation of a switch is quite simple. However there are more intelligent switches that base their internal operation on other information that is stored in a network message. The names of such devices are based on the OSI 7-layer reference model for networks. In every network protocol, administrative information for the layers 2 … 7 is present in network messages.

A switch that reads the administration belonging to OSI-layer 2 from a message and reacts on it is called a “Layer 2” (L2) switch. In Ethernet, the L2 information is: the sending device’s MAC-address, the MAC-address of the destination device, the length of the data field (in bytes), and the upper layer (OSI layer 3) protocol. A switch that reads the administration belonging to OSI-layer 3 is called an “L3” switch, and the same holds for L4, L5, L6 and L7 switches. The higher the layer in the OSI-model, the more knowledge is available and accessible about the network protocol being used and the user’s application. Because of this extra information, the switch can make more intelligent decisions about how a certain message should be processed.
An "L2 switch" functions as described earlier in this chapter. Because the switch needs little knowledge of the content of network messages, just the locations where the MAC-addresses are stored, an L2 switch does not have to have much intelligence and can therefore be very fast. A low "latency" (time between message reception and message transmission) of a switch is an important quality aspect for a switch.

An "L3 switch" is more like a router than a switch, and will usually not be present on smaller networks. The advantage is that L3 switches are faster than routers, but they support fewer protocols and applications. L3 switches are used locally in LANs, while routers are used to connect separate networks (i.e., subsidiaries or branch offices of a company) to the main company network.

An "L4 switch" goes further: it has knowledge about TCP/IP and UDP/IP protocols. By ‘reading’ the contents of the network messages the switch can deduce what the user or application is doing: network telephony? Web surfing? Email? File transfer? Printing? The switch can be configured to handle certain traffic with high priority, and other with low, or lower, priority. For example, IP-telephony can be given the highest priority, email a middle priority, and web surfing the lowest priority. Prioritising network traffic is sometimes called “QoS” or “Quality of Service”, which is also very important for real time applications. L4 switches also have the capability to balance network traffic; for example on heavily used websites the switch can be configured to send 50% of all traffic to one web server and the other 50% to another web server, to prevent one of the servers from becoming 100% loaded while the other is doing nothing.

The differences between L5 and L7 switches are not very clear and depend on the vendor (L6 switches do not exist). With a technology called “webswitching” the switch can extract the URL (http://www….) from network messages it receives, and redirect the message to the correct server.

Most industrial switches are in fact L2 switches. The extra features of L4...L7 switches are very useful in web servers and LAN-servers, but are not necessary for the majority of industrial applications.

Quality of service
With a standard switch it is not possible to handle certain received messages with priority. Usually the switch processes all messages on a “First In, First Out” basis. The standard IEEE 802.1p is an extension to Ethernet that adds message prioritisation capability. This allows critical or delay-sensitive messages to be handled first, which is necessary for audio, video and real-time data. It works as follows: the original sender of a message assigns a “priority class” to the message, which is administered in a “tag” of 4 bytes that is added to the network message. When the switch receives a message, it reads the tag in order to decide how to handle the message. High-priority messages are processed first, and placed in a high-priority output queue on a port, from which they are transmitted before messages placed in the low-priority queue(s).

802.1p supports eight priority classes, where “1” is the lowest and “8” the highest priority. Class “0” means “normal transmission” or “no priority class assigned”.

Because there is a lot of Ethernet equipment that does not support 802.1p, a switch usually has to be configured for each port whether the connected equipment supports 802.1p or not. If a device does not support 802.1p, the switch will remove the tag from any message before sending it to this device. In effect, this means that the priority information is lost. Please note that many switches that do claim to support 802.1p may not support all of the eight priority classes. Very often, only two groups of classes are supported: 0...3 and 4...7. This gives “low priority” and “high priority” messages, with no further distinction between the levels in a group. When using such equipment the application must be aware of this behaviour.
The existence of 802.1p is often mentioned as a reason for Ethernet having real-time capability. In many cases Ethernet has enough real-time capability without 802.1p, but not for all applications, so using 802.1p might be a possible solution. However, there are many switches on the market that do not support 802.1p. And even if a switch does support it, the sending application must be capable of providing the 802.1p tag to an outgoing network message, and this is sometimes not even possible. Undoubtedly this will change in the near future, but don’t count on it today.

Port trunking

Switches many be connected to each other to build larger networks or to build a hierarchy in the network. Using one cable between the switches gives a bandwidth of 100 (or 10) Mbit/s. This is sometimes not enough, for example when 150 Mbit/s is needed. Using two 1 Gbit/s switches instead is a logical step, but financially not always attractive. The “port trunking” feature of switches may help here (note: not all switches support it!). Switches that support port trunking may be connected to each other via more than 1 one cable. For example, 2 cables give a 200 Mbit/s bandwidth; three cables a 300 Mbit/s bandwidth, etc. The switches must be configured in advance, otherwise they’ll use only one cable.

Another reason to use port trunking is the redundancy it offers. When there are two cables between two switches and one cable is broken, 2/3 of the original bandwidth still remains. Another advantage is that the switches themselves do the trunking management: no special network protocols (like the slow Spanning Tree Protocols) are necessary.

Port Mirroring

In contrast to a hub, a switch will forward all incoming network messages to exactly one of the outgoing ports. This is sometimes cumbersome when a network-analyser or network-monitor is connected with the task of monitoring all traffic to / from another device; the analyser will simply ‘see’ nothing of all this traffic. This sometimes makes connecting any analyser or monitor to a switch futile. “Port mirroring” support on the switch is then needed.

A switch with “mirroring” support has the functionality to copy all messages sent to / received from a certain port to another port, so that a network analyser is able to see all traffic (or copies of it) to a certain device. Depending on the supplier of the switch it is sometimes impossible to send messages to a mirror port. This is seldom a problem because a network analyser itself does not have to do anything on the network being monitored.

Flow control

Even though a switch may be capable of communicating full duplex at 100% network load with one device, this does not mean that it is capable of doing this with all connected devices simultaneously. In general (not limited to Ethernet!) a device on a network can only do one thing when it receives a network message which it cannot store (due to lack of buffer memory) or not process: ignore it. Usually this can be done without causing immediate problems, because network protocols at higher levels (such as TCP) detect that a message is missing, and will attempt to ‘repair’ this by forcing the message to be re-sent, in the hope that it will be accepted and processed this time.

It is of course a waste of network capacity and CPU capacity to send a message to its final destination and then have it ignored there. Repair attempts, such as retries, don’t help to lower the network load; in fact they even increase it. It would be much better to only send a message when the destination is able to accept and process it. Mechanisms to do this exist, and are called “flow control”.

Flow control can be executed on all levels of the OSI 7-layer model. For example, RS232 has flow control via its dedicated handshake signals (OSI layer 1). Ethernet has no such signals. A switch has knowledge of OSI layer 2 only, and thus cannot look into the fields for the higher-
level protocols. The switch thus cannot use the flow control capabilities of TCP, which can control the flow of all traffic between any two devices. The switch may only execute flow control on its own part of the network, i.e. the Ethernet-devices which are connected to it. This is always one cable, with the switch at one end and a device at the other end. There is no flow control over the whole message trajectory from sending device to receiving device; higher protocol layers (i.e. TCP/IP) have to handle this. The purpose of flow control in the switch is only that the switch can prevent itself from being overloaded by the devices connected to it, and the devices can prevent the switch from overloading them.

Flow control per cable segment is an option that a supplier can support in his equipment. Whether a device supports flow control or not is determined during the auto-negotiation phase. The “PAUSE” bit indicates that a device supports flow control. If both devices indicate that they support flow control, and if full-duplex transmission is used, then flow control will be enabled.

The flow control functionality itself is handled by Ethernet’s “MAC Control” module, which is part of the data-link layer (OSI-layer 2) of Ethernet. At present, there is only one function supported (flow control); but in the future the MAC Control module may be given more responsibilities. The MAC Control module may autonomously transmit network messages, which all are standard Ethernet messages, but some fields have a special meaning.

- The MAC destination-address field (hexadecimal) is 01-80-C2-00-00-01. This is not the MAC-address of the receiving device, but a multicast MAC-address. The receiving device must have the capability to recognise and process these types of MAC-addresses. Actually, there is no reason to use multicast addresses at all - after all, there can be only one device with one MAC-address at the other end of the cable. The reason for using a multicast MAC-address is that devices do not have to know each other’s real MAC-address.

- The type-field is set to the value (hexadecimal) 88:08.

- In the data field (46 bytes) only the first four bytes are used. The first two will be set to the value (hexadecimal) 00:01; this “Opcode” (Operational Code) distinguishes the “flow control” command from other commands (of which there are now none, but this certainly allows for future extensions). The next two bytes give the value for the “pause number”, in steps of 512 bits. The remaining 42 bytes are set to zero.

The receiver of a flow control message uses the pause-number to calculate how long it should wait before sending anything. For example, if the pause-number has the value 10, and the network operates at 10 Mbit/s, silence should be kept for 10 * 512 / 10M = 0,512 msec. The maximum silence period is 65535 * 512 / 10M = 3,355 sec. At higher bit rates (100 Mbit/s and up) the silence periods are of course proportionally shorter. If a device has sent a certain pause-number but would like to shorten (or prolong) the silence time of the other device, it may send a new flow control message. The silence can be stopped immediately by sending a flow control message with the pause-number value 0. Note that the flow control messages themselves are not subject to any silence interval; they may be sent at any moment.

**Flood control**

“Flood control” is functionality in a switch that helps to limit excessive amounts of messages that are sent by a device, which may impede the proper operation of the remainder of the network.

The most extreme example of excessive message transmissions is a “broadcast storm”, which is started by one device and then kept going by all other devices on the network. The device that starts the broadcast storm on purpose sends a message by broadcast to destination MAC-address FF-FF-FF-FF-FF, but with the sender MAC-address also having this value. The data in the network message itself is deliberately set to such a content that all devices feel im-
peled to reply, but this is broadcast back (because the original message had not set the sending MAC-address field to a real MAC-address, but to the broadcast value). Now everybody receives all the replies, and this may re-start the cycle. Everybody on the network is busy, sending messages to everyone else. The network load will become so high that no normal network message can pass through, and all devices’ CPUs are very busy processing all the received broadcasts, which may cause the application programs to be halted. This is why a broadcast storm is sometimes also called a “network meltdown” (of course, physically the network is not damaged). The broadcast storm may continue until all the devices have been switched off or disconnected.

Broadcast storms can also start in a legitimate way, when errors are made in the network configuration or its software. Such broadcast storms sometimes disappear automatically, but may start up again some time later. For the users of the network this is sometimes noticeable because the network is very slow for a short time, and then functions normally a little bit later.

A switch can counter a broadcast storm by limiting the number of messages that may be sent via broadcast in a certain time interval. When this limit is reached, one of the following occurs:

- The excessive broadcast messages are discarded or
- All broadcast messages are discarded or
- The port on which the excessive broadcasts were received is shut down.

What exactly happens is vendor specific, as are the values of the limits.

Besides broadcast storms, there are also multicast storms and unicast storms. A switch may also support functionality to counter these, but this is only seldom found in switches.

**IGMP Snooping**

The “IGMP Snooping” feature of a switch can help to decrease the number of multicast transmissions that are sent by the “IGMP” (Internet Group Management Protocol). This protocol handles the real-time audio and video transmissions sent by a single source to a group of devices on the network. A simple way to do this is to have the server send separate messages (containing the audio or video data) to every subscriber. This has the disadvantage that many messages, all with the same content, have to be sent over the network. This leads to a high network load, which diminishes the real-time characteristics of the audio and video signals and may cause interruptions, noise, signal loss, etc. A better solution would be to prevent copies of the same message from being sent over the network, or to make copies only when necessary.

When a switch supports IGMP snooping, the switch will help to keep the number of copies of the same message as low as possible. It does this by monitoring (snooping) the IGMP traffic. When the switch detects that two (or more) of its own devices want to connect to the same server, it will let all IGMP traffic pass by the first device, but block the IGMP traffic of the second device internally. When a message from the server then arrives, the switch will send it to the first device, make a local copy, and send this copy to the second device. For the server this has the advantage that it sees only one subscriber, which diminishes the load on the server. For the network between the server and the switch there is also less network load, as only one message has to be sent to the two subscribers. The copy is made at the last possible moment.

Sending data by multicast has been especially developed for the transport of audio and video data via Internet. In the majority of industrial Ethernets this will probably never occur, and so the support for IGMP Snooping in switches might seem to be a bit extreme. However, some industrial protocols can use IGMP Snooping in a very practical way. When a publish/subscribe (or producer/consumer) protocol is used, changes to I/O or process data are always sent (by broadcast) to *all* the devices on the network. Although fast, it has the following disadvantages:

- Those devices not interested in the data must also process broadcasts.
Routers do not forward broadcasts.

In smaller applications this is not a problem, but for larger applications the use of broadcasts (or too many broadcasts) is disadvantageous. A solution is not to use broadcasts but to use multicasts in combination with IGMP; anyone interested in the data can subscribe themselves and will receive the data.

A “multicast group” consists of a server (as data source), and one or more subscribers. A device can join (or leave) the multicast group by sending a message to the server. On any network, multiple multicast groups can be active simultaneously; IGMP is responsible for the administration of these groups. Via a command “IGMP Report Group” or “IGMP Leave Group” a device can join or leave any multicast group it wants. How this is done exactly depends on the IGMP version in use: 1, 2 or 3. Version 2 is still very common; i.e. Microsoft supports version 3 only in XP, and in the Unix / Linux variants version 3 is not always supported.

Without IGMP snooping a switch can process multicasts in one way, like broadcasts, i.e. forward on every port. With IGMP snooping support a switch will forward multicasts only to those ports to which members of the indicated multicast group are connected.

Why does a switch without IGMP snooping handle multicasts like broadcasts? This is caused by the usage of the special “Group Destination Address” (GDA); this is a MAC-address with the special value 01-00-5E-XX-YY-ZZ (see chapter 2). The value for XX-YY-ZZ specifies the multicast group. The GDA is always written to the destination MAC-address field in the Ethernet message, and never in the source MAC-address field. A switch will thus never learn which port a device with the GDA is connected to. If it then needs to send a message to this MAC-address, the switch will revert to its default behaviour: forwarding copies of the message to all ports on the switch.

A switch with IGMP snooping handles the IGMP traffic in a more elegant way. It intercepts all IGMP messages, and checks to see if a certain device wants to join a multicast group with the “IGMP Report Group” command. This message contains the device’s own MAC-address. The switch now knows which port this device is connected to, and it also knows the multicast group (XX-YY-ZZ). Whenever the switch receives a message for any possible multicast group, it checks if there are any local subscribers, and sends copies of the messages to all of them. If there are no subscribers, the switch will simply do nothing with the IGMP traffic. This means that the IGMP traffic is discarded as soon as possible (= as close as possible to the server), instead of its having to traverse the whole network and be discarded anyway.

The reverse happens when a device wants to leave a multicast group. It must send an “ICMP Leave Group” message to the desired server. The switch will intercept this message too. In contrast to joining it, is not possible to stop all IGMP traffic for the specified multicast group; only when the last device of a multicast group indicates a desire to leave the group can the switch stop forwarding multicast traffic to its own devices. Of course, the switch must forward the last “ICMP Leave Group” message to the server; it will stop its transmissions when the very last device has left the group.

Switch start-up time
After a switch has been turned on, it will require some time to initialise. Exactly how long this takes depends on the presence of STP (Spanning Tree Protocol). If this protocol is not used, the start-up time will be in the order of a few seconds. If STP is used, then the STP reconfiguration has to be done first, because the switch activation may cause a network reconfiguration, which needs to be calculated by STP (in cooperation with the other switches on the network). Depending on the topology of the network, this may take 30...60 seconds. Switches with support for “Fast STP” will take less time.
Switch firmware update
The functionality in a switch is usually handled by an extensive amount of software (sometimes also called firmware). This software is not only necessary to control the hardware of the switch, but also to handle all the network protocols needed for switch management that are supported, like SNMP (Simple Network Management Protocol), Telnet (for remote login), an embedded web server, etc. It is not uncommon for errors (“bugs”) to be found in the software, or for a security leak to be discovered. In both cases the supplier will provide new software, which must be loaded into the switch. Usually this can be done via the network itself. After having loaded the new software, it will be activated after a restart. Please note that the loading of new software is sometimes a very critical action, as it may not be interrupted in any way.

5.6 Connecting equipment
The RJ45 connector is used on 10BaseT and 100BaseTx Ethernet, which uses four pins for the bi-directional transmission channel: two for the “transmit” (TD+ and TD-) and the other two for the “receive” (RD+ and RD-) signals. A connection between two devices can simply be made by connecting both devices’ TD-signals to the other devices’ RD-signals.

The signals TX+, TX-, RD+ en RD- can always be found on pins 1, 2, 3 en 6 of the RJ45 connector. Which signal is on which pin differs for “standard” Ethernet devices and hubs / switches. This gives the following cabling possibilities:

- An Ethernet-device connected to a hub / switch (see “A” below)
- Two Ethernet-devices connected to each other (“B”)
- Connection of two hubs / switches (“C”).

A. An Ethernet-device connected to a hub / switch.
This situation is the most common. On both devices the electrical Ethernet signals are connected to the RJ45 connector as follows:

<table>
<thead>
<tr>
<th>Ethernet-device</th>
<th>Hub / Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>RD+</td>
</tr>
<tr>
<td>Pin 2</td>
<td>RD-</td>
</tr>
<tr>
<td>Pin 3</td>
<td>TD+</td>
</tr>
<tr>
<td>Pin 6</td>
<td>TD-</td>
</tr>
<tr>
<td>Pin 1</td>
<td>TD+</td>
</tr>
<tr>
<td>Pin 2</td>
<td>TD-</td>
</tr>
<tr>
<td>Pin 3</td>
<td>RD+</td>
</tr>
<tr>
<td>Pin 6</td>
<td>RD-</td>
</tr>
</tbody>
</table>

A “1:1” cable will suffice to allow communication.

B. Two Ethernet-devices connected to each other.
Ethernet devices may be connected directly to each other, without using a hub or switch. A disadvantage is that the network cannot be larger than these two devices, but in some cases this is no problem at all. On both devices, the following signals are found on their connectors:

<table>
<thead>
<tr>
<th>Ethernet-device 1</th>
<th>Ethernet-device 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>RD+</td>
</tr>
<tr>
<td>Pin 2</td>
<td>RD-</td>
</tr>
<tr>
<td>Pin 3</td>
<td>TD+</td>
</tr>
<tr>
<td>Pin 6</td>
<td>TD-</td>
</tr>
<tr>
<td>Pin 1</td>
<td>RD+</td>
</tr>
<tr>
<td>Pin 2</td>
<td>RD-</td>
</tr>
<tr>
<td>Pin 3</td>
<td>TD+</td>
</tr>
<tr>
<td>Pin 6</td>
<td>TD-</td>
</tr>
</tbody>
</table>

With a 1:1 cable, it is not possible to connect these devices to each other: pin 1 (2) of the first device must be connected to pin 3 (6) of the other device. A so-called “crossover” cable is needed. Because such cables are not often used, it is advisable to label these cables very clearly.
C. Two hub / switches connected to each other.
Sometimes it is necessary to connected two hubs or switches to each other, usually when the number of ports per hub / switch is too small, or when a hierarchy in the network is desired. The Ethernet signals on both the hub and the switches are connected to the RJ45 as follows:

<table>
<thead>
<tr>
<th>Hub / Switch 1</th>
<th>Hub / Switch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>TD+</td>
</tr>
<tr>
<td>Pin 2</td>
<td>TD-</td>
</tr>
<tr>
<td>Pin 3</td>
<td>RD+</td>
</tr>
<tr>
<td>Pin 6</td>
<td>RD-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hub / Switch 1 uplink port</th>
<th>Hub / Switch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>RD+</td>
</tr>
<tr>
<td>Pin 2</td>
<td>RD-</td>
</tr>
<tr>
<td>Pin 3</td>
<td>TD+</td>
</tr>
<tr>
<td>Pin 6</td>
<td>RD-</td>
</tr>
</tbody>
</table>

This too requires a “crossover cable”.

Many suppliers anticipate their customers’ wish to be able to connect hubs and switches to each other in a simple way (= without crossover cables). Usually one port on the hub / switch is labelled as the “uplink port”. The Ethernet signals are then reversed on the RJ45. This allows a standard cable to be used.

<table>
<thead>
<tr>
<th>Hub / Switch 1 uplink port</th>
<th>Hub / Switch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>RD+</td>
</tr>
<tr>
<td>Pin 2</td>
<td>RD-</td>
</tr>
<tr>
<td>Pin 3</td>
<td>TD+</td>
</tr>
<tr>
<td>Pin 6</td>
<td>RD-</td>
</tr>
</tbody>
</table>

Note: sometimes the uplink port is not an extra port, but is only made available as an extra connector that is internally wired in parallel with another port (very often the highest numbered port). This means that a port has two connectors, only one of which may be used!

MDI / MDI-X
A solution that doesn’t require an extra connector is simply to toggle the RD and TD-signals on the connector, for example with a small (push-button) switch. Usually one of the ports will have such a push-button, which is then labelled “MDI / MDI-X” (Medium Dependent Interface, normal or ‘crossed’). It enables you to use a port either as an uplink port, or as a normal port.

A hub / switch with MDI/MDI-X capability or with a separate uplink port can only connected to one other hub / switch. If such multiple connections need to be made, the only solution is to use crossover cables again.

Auto-crossing
Some vendors support the so-called “auto-crossing” functionality on their equipment. The switch itself calculates which wires offer the TD and the RD signals, and internally route this to the corresponding electronic parts. It is then no longer necessary to have an MDI/MDI-X button, or to separate uplink ports. Crossover cables may be used, but don’t need to be. The main advantage to the user is ease of use.

5.7 Industrial switches
As the name implies there are a several differences that distinguish industrial switches from their ‘office’ versions:

- Sturdy housing
- Usage of special connectors
- Power supply 24V, usually redundant
- Small number of ports (4/8/12/16)
- Higher MTBF
- Extended temperature range (i.e., −40...+60 degrees C)
- No need for forced ventilation (no ventilator)
- High resistance against EMC / vibration / shock etc.
- Relay or potential-free contact for indicating events.

Many vendors sell industrial switches. According to a report (2003) by ARC (Automation Research Council) the industrial Ethernet market will grow at an annual rate of 84% in the coming years. With such optimistic figures it is not surprising that even the “office equipment” switch vendors (such as Cisco) are releasing industrialised versions of their products.

![An industrial switch (source: Harting).](image)

Because the uptime of the switch is an important factor in the calculation of the uptime for a whole network, many vendors of industrial switches support a redundant (usually 24V) power supply for their equipment. Figure 5-11 shows the (simplified) electrical circuit of a Phoenix switch.

![Diagram of a switch with redundant power supply to be connected to US1 and US2 (source: Phoenix Contact).](image)

The price of an “industrial switch” is much, much higher than an “office” switch. Whereas a standard office switch can cost as little as £ 10 per port, an industrial switch can sometimes cost as much as £ 50...100 per port. Unless this price goes down, I do not see industrial Ethernet competing with existing industrial networks like AS-Interface, which do not require switches and use far cheaper cable as well.
5.8 Spanning trees

All sorts of network topologies can be made with hubs and switches, except one: a ring. If this happens, the network will cease to function. Often, a ring is created by accident, because the documentation of the current network wiring and equipment is not up-to-date. But sometimes a ring is actually desired, because of its advantage: the redundancy that it offers – a network message can reach its destination either clockwise or counter clockwise. A broken or shorted network cable or a non-functioning switch is not fatal for the operation of the network.

But, as stated, a ring cannot be made with standard hubs or switches. When hubs are used, every message is forwarded from hub to hub to hub, and finally it arrives back at the original hub, and then the cycle repeats itself. The message will circulate forever. The network load becomes 100%, the number of collisions becomes excessive, and the network cannot be used for any production until the moment that the ring is broken (opened).

When a ring is built with switches, then (directly after power-on) each switch will behave like a hub, because the switch table is still empty. The first message will be continuously retransmitted. It is possible to think of a scenario where the network will become silent again and usable for production, but this cannot be predicted and is therefore unusable.

Despite these disadvantages, the redundancy aspect of a ring-wired network is so attractive that solutions have been devised to counter these problems. The solution is actually quite simple: open the ring somewhere, and close it in case of trouble. This can be done in one of the following ways:

- Manually open the ring, simply by removing one of the cables between two switches or hubs. If a problem occurs in the network, re-insert the cable. This way of working has the advantage that it doesn’t require software and that it can work with any switch or hub. The disadvantage is obvious: it can take quite a long time to detect the original problem, and then a manual action must be performed by someone within physical range of the switch / hub. Although technically this solution will work, for industrial applications it is not acceptable.

Or:

- Open the ring under manual software-control, by sending a command to a hub / switch to disable one of the ports with which it is connected to another hub / switch. Some vendors offer this functionality on their equipment. The disadvantages are almost the same as those in the previous solution: it can take quite a long time before the original problem is detected, and then a manual action is needed (with the difference that it can be done from any location with access to the network involved).

When switches are used, there is a third way:

- Open the ring under automatic software-control, via the “STP” protocol. This protocol is also capable of detecting trouble in the network, and can automatically enable and disable redundant links. No human intervention is needed.

The “STP” protocol (Spanning Tree Protocol) is completely unrelated to “STP” wiring (Shielded Twisted Pair). Spanning tree was originally developed by Sun, and is now standardised in IEEE 802.1d. It is thus not Ethernet-specific; it can work on any network. In practice, it is usually used on Ethernet.

**STP operation**

Switches with STP support communicate with each other, and together they calculate a map on which all connections (links) between these switches are drawn. It then becomes clear which
links in the network are redundant. This does not always have to be a ring, for example with two links wired between two switches there is also a redundant link.

After some time (average 30…50 seconds) a decision will be made about which redundant link(s) can be deactivated. The result is, mathematically speaking, a “spanning tree” where the switches are the branches and the Ethernet devices the leaves. Everyone on the network is reachable via one path only. It is impossible for network messages to circulate, because a message never comes back to its original switch.

An example of a possible configuration is given in figure 5-12 (left): a network with 4 switches and 4 connections (cables) between them. This is a redundantly wired network, with the loop. After STP has calculated the network topology, it knows that there are four possible configurations without redundant wiring (figure 5-12, right). One of the connections between the switches must be turned off. Despite this, all switches are still reachable for everyone.

![Figure 5-12: a network with a ring built from four switches (left). The four possible “spanning trees” are shown on the right. The fourth link is still there, but deactivated (not drawn).]

The user can configure his preferences – for example, have STP de-activate the slowest redundant links first, or the most expensive, etc. One the network is operational, nothing need be done until the moment that one of the links is no longer operational. A new spanning tree has to be built, bypassing the faulty link to make the network operational again. As an example, take one of the networks drawn in figure 5-12. There are three links each; a failure of any of them causes a new spanning tree to be calculated, which (in this example) will always cause the deactivated redundant link to become operational (figure 5-13).

![Figure 5-13: After a link failure the spanning tree will be recalculated by the switches; in this case by activating the previously de-activated link.]
In the examples above we have assumed that STP operates on a ring-wired Ethernet. However, it can work with any network topology.

**Rapid STP**
The speed at which STP reacts to changes in the network topology, usually in the order of 30 seconds or more, is no longer sufficient, not even in office environments. Industrial applications usually require much faster reaction times. A newer version of STP called “Fast STP” but also “Rapid Reconfiguration STP” or “Rapid STP” (RSTP) has been released as standard IEEE 802.1w. Under normal circumstances, a reaction time of 1...10 seconds is possible. But even a time of 1 second is ‘eternity’ in many industrial applications. Many vendors have therefore developed their own solutions.

**Even faster reconfiguration**
Several vendors offer STP variants that are considerably faster than the “normal” STP of Fast STP implementations. The drawback is that the network topology may only be a ring. Also, all switches must be bought from the same vendor.

Hirschmann’s “HiperRing” enables a reaction time of ca. 500 msec. This time is almost independent of the number of switches and the size (length) of the network. These are two advantages of HiperRing in comparison with RSTP, which is only faster with a small number of switches (max. 3), and has an unpredictable reaction time. Another aspect of RSTP is that there is a chance, though small, that network messages are delivered in another order than originally sent, or that network messages may be duplicated. For some protocols (like TCP/IP) this is not a problem, but other protocols may not be able to handle these unexpected messages.²

The Moxa “TurboRing” protocol claims a reconfiguration time of less than 300 msec with 120 devices. The prize for the quickest reconfiguration goes to OnTime’s “FRNT” (Fast Reconfiguration of Network Topology), with times of less than 30 msec. Of course, care should be taken when reading such data – the conditions (network topology, network protocol, network load, etc.) under which the times mentioned can be reached are not specified. Usually “best case” figures are published.

**Alternative**
If it is not necessary to have redundant network cabling everywhere, it might be economically advantageous and technically interesting to use switches that support “port trunking”. No protocol like STP or Fast STP is necessary, with its associated reconfiguration delays if problems occur.

**Disadvantage of STP**
A disadvantage of STP or Fast STP is that some network cables and some ports of switches are not used because they are used for the stand-by circuits. Given the price of switches and cabling, this may be an expensive solution. Port trunking may be a better alternative, because all ports and all network cabling is used all the time; under normal circumstances the capacity of the network is much better than with STP, and if problems occur, it is as good as anything STP can offer.

**5.9 Always use switches?**
Many suppliers of switches heartily recommend them³ instead of hubs. As described earlier, this is technically not always advantageous; in quite a number of cases the advantages of using

---
² Source: ComConsult GmbH, “Product Analysis of Redundancy Aspects of Hirschmann Switches (HiperRing vs. RSTP), 15 September 2003 (in German only; to be found on the website).
³ Switches are usually more expensive than hubs, although the price difference has become much less significant than it was 5 years ago. Some suppliers don’t even sell hubs anymore.
a switch are negated by the network protocol or software network protocol being used. A few examples:

- A switch allows a double bandwidth per device (full-duplex: receive and transmit simultaneously). When the network protocol works according to a master/slave principle (half-duplex: either send or receive) the full-duplex capability of the switch remains unused.

- A switch allows multiple devices to be active on the network concurrently. When an application is used where there is a central controller who communicates with all devices, but the devices do not communicate with each other, an important switch feature is again not used.

- A switch only forwards a network message to the port to which the destination device is connected. If a network protocol is used where broadcasts are used extensively (i.e., in a producer/consumer or publish/subscribe protocol), then every message has to be forwarded to all ports. Such a network protocol makes a switch behave like a hub (unless VLAN's are used).

It thus depends on the application protocol(s) being used (see chapter 6 for a summary) whether the advantages of a switch are realised. Of course, one could wonder whether it is better to always use a switch, due to the functionality supported and the extra bandwidth it offers for future extensions to an application. My advice is: if it is technically possible to use cheaper hubs, why not use them? If, in the near future, it turns out that switches are needed, an upgrade is easily done: remove the hub, insert the new switch and turn the power on: ready.
As discussed earlier, Ethernet is only a small part of a complete network: 1) cabling, and 2) the capability to transmit and receive messages. This corresponds to the OSI-model layers 1 and 2. The functionality to implement the remaining layers 3...7 has to be added, to give a complete “protocol stack”. Only then are two devices capable of communicating with each other.

Protocols have been under development since 1999, but as there is no dominant supplier or organisation in the industry, there is no standardisation and subsequently many new protocols have been launched in the past, or will be launched in the coming years. An incomplete summary:

- ProfiNet, by the Profibus Trade Organisation (PTO);
- Ethernet/IP, by the Open DeviceNet Vendors Association (ODVA);
- Powerlink, by the Ethernet Powerlink Standardisation Group (EPSG);
- IDA, by the Modbus/IDA-group;
- Modbus/TCP, by the Modbus-group;
- FF HSE, by the Fieldbus Foundation;
- Sercos-III, by the Sercos Interest Group (IGS);

Many suppliers also have their proprietary protocol, including:

- SRTP (Service Request Transfer Protocol);
- EGD (Ethernet Global Data);
- Safe Ethernet;
- RTEthernet (Real Time Ethernet);
- NDDS (Network Distributed Data Services);
- Etc.

All these protocols claim to be “Ethernet”, which is partially true because the implementation of the OSI-layers 1 and 2 is indeed Ethernet, but the higher layers 3..7 are always unique. This means that any two of the listed protocols can co-exist on the same network, but devices with different protocols cannot communicate with each other. I have the feeling that many suppliers are reluctant to mention this, as it might discourage the usage of industrial Ethernet.

Some of the better-known protocols are discussed in more detail in the following section. Section 6.9 discusses time synchronisation protocols, especially IEEE-1588 as this is rapidly gaining acceptance. It is used as part of Powerlink, ProfiNet and Ethernet/IP; a derivative is used in JetSync, and similar technologies in other protocols.

6.1 Modbus/TCP
Modbus/TCP is currently (2004) the most popular industrial Ethernet protocol. The reasons for this popularity are most likely the fact that it has been on the market for a relatively long time (since 1999), its simplicity (can be implemented in about 1 week by someone knowledgeable about TCP/IP), its compatibility with the existing Modbus protocols, free access to its specification (see www.modbus.org), and its ease of use (can be learned in 1 day).
As the name indicates, Modbus/TCP is a member of the large Modbus-family of protocols: Modbus/ASCII, Modbus/RTU, Modbus+, Jbus, Modbus/SFB). There are large similarities amongst all these versions; Modbus/TCP is 95% identical to Modbus/RTU, with the remaining 5% of differences necessary for the usage of TCP/IP and several editorial clarifications regarding inconsistencies in the original Modbus/RTU specification.

Modbus/TCP uses TCP/IP as a reliable transport mechanism for the delivery of commands and responses. Usually TCP/IP itself will have an underlying Ethernet; hence Modbus/TCP is classified as an “industrial Ethernet” protocol. But an advantage of TCP/IP is that it is also able to use other media; as you will probably have noticed at home or at work, TCP/IP traffic can also be carried over serial interfaces (RS232), ISDN or ADSL telephone lines, mobile phones (GSM/UMTS/GPRS), wireless LANs, etc. The Modbus/TCP protocol can thus also be run over these networks. This is an advantage that most of the other industrial Ethernet protocols do not possess (or not any longer), as they are not always using TCP/IP.

The advantages of Modbus/TCP are:

- The existing Modbus/RTU message structure has been largely retained.
- Higher speed, as a 100 Mbit/s Ethernet is inherently faster than a serial line.
- The master/slave limitation is removed, now allowing for multi-master Modbus networks.

A minor disadvantage is that broadcasts are no longer supported, because TCP does not support broadcasts. The reason it is only a minor issue is that many Modbus/RTU implementations lack broadcast support too.

The existing Modbus/RTU message structure has been retained, except for two fields. First, a 6-byte header has been added to allow the usage of TCP/IP and the introduction of possible future extensions. This new header remains invisible for the application. Second, it is no longer necessary to add one parity bit to each byte, or to add the 16-bit Cyclic Redundancy Check field (CRC) at the end of the message. The parity bits and the CRC were necessary in Modbus/RTU to detect transmission errors and corrupted data. But as both TCP/IP and Ethernet add similar functionality, it was not necessary to add a 3rd error-detecting and error-correcting layer. This has the additional advantage that the CRC-calculation does not have to be done by the Modbus/TCP protocol stack, which saves on CPU-intensive calculations and thus increases the speed of Modbus/TCP.

Until now, Modbus/RTU systems have always been based on RS232 or RS485, with a speed limited by the capabilities of serial ports, i.e. 19.2 Kbit/s or 38.4 Kbit/s. Because of the substantial overhead in the Modbus/RTU protocol, the net speed of any network was quite low. Modbus/RTU was at a serious disadvantage here compared to other, more modern, industrial networks. The usage of a 10 or 100 Mbit/s Ethernet gives Modbus/TCP a huge performance boost.

Modbus is by origin a master/slave network, as was in vogue in the 80’s. But any master/slave network puts severe limitations on its users, as only the one master can communicate with the slaves, but the slaves cannot communicate with each other. This is not acceptable in distributed systems, where every device must be able to communicate with every other device. Since the introduction of TCP/IP this limitation is no longer present, because TCP/IP supports peer-to-peer communication. All that is needed for a device to be a ‘master’ is the capability to send a message, and for a device to be a ‘slave’ it has to have a capability to receive messages, process them, and send an answer back. This is trivial to implement on TCP/IP; any programmer capable of working with “sockets” (either on Unix or Windows) can implement this within one week. What is more difficult is a capability to be able to be a master and a slave simultaneously; many commercial Modbus/TCP implementations do not allow this. In such cases, Modbus/TCP is still limited to master/slave relations between devices.
Using TCP has one disadvantage: a broadcast (from the master to all slaves simultaneously) is no longer possible. TCP is a ‘peer-to-peer’ protocol that can be compared to a “data pipe”: data put in the pipe at one end is delivered at the other end. A solution could be to copy the same message \( n \) times and send it to all \( n \) slaves separately, but this gives a much higher network load, and there is no guarantee that all \( n \) slaves receive the message at the same moment. The semantics of a simulated TCP-broadcast are thus not identical to a ‘real’ (Modbus/RTU) broadcast. However, as many commercial Modbus/RTU implementations did not support broadcasts anyway, this is not seen as a real limitation of Modbus/TCP.

Some Modbus suppliers have launched the idea of making a Modbus/UDP protocol, using UDP instead of TCP, as UDP does allow for broadcasts. However these ideas are very preliminary, non-standard, and certainly not available for sale.

In 2002, Modbus/TCP was offered to the Internet Engineering Task Force (IETF) as a proposal for a new Internet standard. The purpose behind this offer was not very clear, as Modbus/TCP would in no way become a better network by having it stamped as an Internet standard. Probably the only reason was to make network administrators more familiar with Modbus/TCP; if a user wants to send Modbus/TCP traffic over Internet (i.e., for remote diagnostics purposes) all intermediate firewalls have to be configured to allow Modbus/TCP traffic to pass through. One of the “TCP/IP ports” has to be set open for this (port 502), and I guess it is easier to convince a network administrator if he can be referred to an Internet standard document stored somewhere on the IETF website. Unfortunately, the Modbus/TCP proposal did not get enough “yes” votes to become an Internet standard, not because there were too many “no” votes, but only because nobody made an effort to collect enough “yes” votes.

### 6.2  ProfiNet

ProfiNet is the industrial Ethernet version of the Profibus family. Actually it is a completely new protocol, in that it does not resemble any currently existing protocols (FMS, DP, PA, FDL) now in use for more than 15 years.

The development of ProfiNet started in 1999, and the plan was to bring the first products onto the market in 2003. In 2000 it was decided not to wait so long, but to launch intermediate versions as well; these are now known as ProfiNet V1, V2 and V3, or under their names “ProfiNet”, “ProfiNet SRT” (Soft Real Time) and “ProfiNet IRT” (Industrial Real Time). Profibus V3 / IRT is now in the last phase of development, and its release is expected in the first half of 2005. ProfiNet V2 has been available since 2003. After the release of IRT, development will continue on a version for the process-automation market (expected in 2007), and a safety version based on the existing ProfiSafe standard.

#### ProfiNet versions

In the first years of the development of industrial Ethernet, the idea was to use Ethernet in combination with TCP/IP and a specific application protocol. This would give maximum leverage to the available knowledge, allow re-use of infrastructure components (switches, routers), software (web browsing, file transfer, network management) and would connect easily and cheaply to PCs and Unix workstations. The first version of ProfiNet followed these ideas. Unfortunately, no excessive demands could be placed on speed and determinism: the existing protocol-stacks were not designed for high-speed real-time applications. This made ProfiNet V1 sometimes considerably slower than existing Profibus/DP networks. ProfiNet V1 was just put on the market by the Profibus Trade Organization to show that progress was being made and that Profibus would not be surpassed by its competitors. A few updates of V1 were released in quick succession, stopping at V1.2 before the more innovative V2 was announced.

In ProfiNet V2, also called “SRT” (Soft Real-Time), TCP/IP is no longer used for the transfer of I/O data and alarms. ProfiNet now completely bypasses TCP/IP, and sends and receives
frames directly on Ethernet. This gives very little overhead during transmission (increasing network efficiency) and allows Windows / Unix protocol stacks to be completely bypassed, giving Profinet more influence on the handling of network messages (= higher speed, better real-time characteristics). Cycle times in the order of 5..10 milliseconds are now possible, about identical to Profibus/DP. TCP/IP is still used for start-up, configuration, downloads, web services, emails, file transfers, etc. Additionally, Profinet SRT no longer depends on Microsoft’s DCOM technology, as was the case in V1. It had received considerable criticism for using DCOM, which firmly linked it to the Microsoft Windows platform and a few other platforms for which a DCOM implementation is available (i.e., VxWorks), thus implying a connection to Microsoft’s development strategy, now migrating to .Net. Since Profinet V2, it is possible to negotiate other protocols than DCOM for the communication between devices on the network.

Profinet V2 is much faster than V1, but not spectacularly fast despite it being based on a 100 Mbit/s Ethernet; a Profibus/DP network offers a comparable speed even though its bit rate is only 12 Mbit/s. The first products entered the market in the 2nd half of 2004.

In an attempt to make Profinet usable for high-end motion applications, like printing presses, CNC controllers, plastic extrusion, packaging machines etc. a much higher network speed is needed than could be offered by Profinet V2. These types of applications also demand a very constant speed, with guarantees that certain messages on the network are transmitted at very well defined moments. This so-called “jitter” must be as small as possible, and current technology allows jitters < 1 microsecond. But this is only possible when as much as possible of a network’s protocol is handled in hardware. This is the basis of Profinet V3, also called “IRT” (Industrial Real-Time). Special network controller-chips, called Eltec 200 and 400, are designed for this (by Siemens). These chips are already available for prototyping; full release was expected in the beginning of 2005, but has been delayed.

An additional feature of the Profinet chip is that it is going to make a bus topology possible; each chip is a 4-port switch (called “machine distributor” in Profinet). This makes it no longer necessary to buy a separate (industrial) switch. This gives Profinet an important financial advantage in comparison to most of its competitors.

Like any other I/O network, Profinet will continuously execute the same cycle. The user sets the duration of a cycle. Every cycle is split in two: in the first part all real-time data will be transmitted, and in the time that is left the non-real time (“asynchronous” or “acyclic”) data and/or TCP/IP traffic will be transmitted. Because all network cycles have the same duration, real-time

---

**Figure 6-1: The architecture of the Profinet protocol stack. Note the presence of Microsoft’s DCOM, and Profinet’s own I/O protocol which completely bypasses TCP/IP.**

---
data always has priority; if there is not enough time to transmit all non-real time data, the transmission of any remaining data must wait. This way of working is not unique to ProfiNet; many other Ethernet systems (such as Powerlink and Sercos-III) and existing industrial networks operate in the same way (but with subtle differences).

**Cabling**

ProfiNet assumes that a 100 Mbit/s Ethernet is used, but it explicitly mentions that a 10 Mbit/s network Ethernet can also be used. Cabling is standard STP (Shielded Twisted Pair) of optical fibre (max. 2 km for multi-mode fibre, 14 km for single mode). The recommended connector is the standard RJ45, for which a IP67 version is also specified. The M12-connector (D-version) is also allowed for ProfiNet. Finally, a hybrid connector consisting of an RJ45 and 4 additional connector pins to be used for providing power is specified (supplied by Harting).

**System architecture**

In contrast to the existing members of the Profibus protocol family, ProfiNet does not just describe the protocol, but also how an application is going to be programmed. Whereas older networks just transmit bits and bytes back and forth, ProfiNet has a much higher level of abstraction: connecting “components” together programs an application. A component can be just a physical device, but also a software module (i.e. a PID-controller). An application is literally drawn on the screen; rectangles representing components with lines between these components that indicate how data flows. Anyone familiar with CAD-systems for designing electronic circuits will immediately recognise this way of working. The software package that provides this functionality is called the “Interconnection editor”.

The editor “knows” which components exist, as the user fills a database with component descriptions. These are stored in “ProfiNet Component Description” (PCD) files. An equipment supplier must provide the PCD files. When a user wants to create his own components, the “ProfiNet Component Editor” can be used. This software tool is provided by the Profibus User’s Group, but only for its members.

Finally, an application is build by connecting components together. The next step is to map the components to hardware devices, and to configure every device. Via ProfiNet itself each component is sent its own configuration, together with the “Interconnection Data” that describes who provides the values for the input-side of a component, and to whom the own outputs must be sent. This completes the ‘engineering phase’ of ProfiNet; the ‘run-time phase’ can now start.

**Integration**

Even though ProfiNet is a new protocol, the wheel is not re-invented, as existing protocols from the TCP/IP family of protocols are re-used where possible. This makes it easier to integrate ProfiNet into existing networks, re-use existing tools, and re-use knowledge. We already mentioned the use of Microsoft’s DCOM (Distributed Common Object Model), but DHCP (Dynamic Host Configuration Protocol) and SNMP (Simple Network Management Protocol) are also part of ProfiNet. Additionally, there is support for HTTP (Hypertext Transport Protocol), HTML (Hypertext Mark-up Language), XML (extensible Mark-up Language) and OPC (Object linking & embedding for Process Control).

The configuration of ProfiNet devices can be done in two ways: the own “DCP” (Discovery and Basic Configuration Protocol), or the better-known “DHCP” (Dynamic Host Configuration Protocol). Support for DCP is mandatory for all ProfiNet devices, but DHCP support is optional.

SNMP (Simple Network Management Protocol) is used for network management and diagnostics. Despite the “simplicity” of SNMP it is actually a sophisticated standard, widely in use in the LAN world. With existing SNMP tools it is possible to monitor ProfiNet devices, read diagnostics data and set network parameters in a device. However it has been decided to support only the mandatory SNMP variables; ProfiNet-specific variables are not accessible via SNMP.
Web integration is possible because of the support of HTTP. Web pages can be accessed, and sent back to the browser in either HTML or XML format, so that any standard browser can be used. Every Profinet device can support a local web server, although this is not mandatory.

**Connections to other systems**
A coupling between Profinet and OPC is not difficult. The technological basis of both systems is the same: Microsoft’s DCOM-technology. But a large difference is that Profinet uses real objects, while OPC uses "tags" (names). In principle every Profinet-device can act as an OPC-server. Additionally Profinet supports OPC/DX, allowing an easy coupling of Profinet to OPC-servers using other protocols. Finally, any OPC server can be given a Profinet-interface; a special software converter ("OPC Objectizer") performs the conversion from OPC to Profinet.

It is of course possible to make a connection between Profinet and OPC. This makes it possible to use existing DP networks and add a Profinet layer, making migrations easier and protecting investments in DP. Of course it is not possible to directly connect DP equipment to Profinet, due to the differences in physical interfaces: a 100 Mbit/s full-duplex 4-wire Ethernet vs. a 12 Mbit/s half-duplex 2-wire RS485-based network. A conversion module is necessary (figure 6-2). Profinet calls such modules a “proxy”.

The result is a network architecture of 2 layers, with Profinet on the top level and Profinet on the bottom level. It is possible to connect multiple proxies, each with their own DP network. The proxy itself functions as a DP master, handling the communication to all DP slaves. The proxy hides this on Profinet, in effect simulating multiple DP slaves. The proxy will accept commands via Profinet, and either handles them locally or executes a conversion and sends the command to the DP slave. The proxy then waits for the answer, and sends the data in a Profinet message back to the original requester.

Phoenix Contact has decided to integrate its Interbus with Profinet. A special committee has been established to write the specification. At the time of writing no further details were known. Most likely other industrial networks will also develop proxies for Profinet in the near future.

**Documentation**
Documentation about Profinet is available from the Profinet User’s Group at [www.profibus.com](http://www.profibus.com). The SRT / V2 specification is also available for download, but in order to understand this fully one needs to have knowledge about Microsoft’s DCOM technology. The 21-page “Profinet Technology & Application System Description” and the “Profinet Installation Guideline” are easier to understand.
6.3 Ethernet/IP
Ethernet/IP (Industry Protocol\(^4\)) is a development by Rockwell / Allen-Bradley. Apart from the wiring, there is a great resemblance to this company’s other two industrial network protocols: DeviceNet and ControlNet. All three systems have a common application protocol called CIP (Control & Information Protocol). In short, Ethernet/IP is nothing else than CIP with TCP/IP added on an Ethernet wiring infrastructure.

![Diagram of network architecture](image)

Figure 6-3: A comparison of the network architecture of DeviceNet, ControlNet en Ethernet/IP. All three have a common protocol CIP; of course the wiring and lower-level protocols are different for all three.

The specification of Ethernet/IP can be found on the website [www.ethernet-ip.org](http://www.ethernet-ip.org) (or [www.odva.org](http://www.odva.org)). Although it is claimed that this specification is free, this is not strictly true as many chapters simply consist of referrals to the ControlNet specification, which is not free. An open-source example of a limited-functionality Ethernet/IP implementation is also available.

An interesting feature of DeviceNet, ControlNet and Ethernet/IP is the identical application layer protocol (OSI layer 7). This makes these three systems attractive to users, as they only have to learn one protocol, and allows for easier migrations. Also, CIP-extensions can be used on all three systems, for example CIPSafety, for safety applications. None of the other industrial Ethernet systems offers this commonality; for example Siemens has three completely different protocols: AS-Interface, Profibus and ProfiNet, and each of them in a separate safety-version. This does not make life easier for users, who have to learn three different systems with little common functionality.

6.4 Powerlink
Powerlink is a development by the Austrian company B&R. Initially Powerlink was meant as a protocol for B&R’s own PLC for high-speed motion applications, but later a user’s group (which is chaired by a B&R employee) has been formed that handles the further development of Powerlink. A Swiss university, contracted by B&R, provides Powerlink implementations. Although promises were made to “open up” the specification, as of 2004 this has still not happened.

---

\(^4\) This other use for the abbreviation IP has given and will give considerable confusion, as it has nothing to do with the IP in TCP/IP (Internet Protocol).
Powerlink differs from all other industrial Ethernet protocols because it specifies that switches may not be used, and that hubs are essential for the high speed and real-time characteristics of Powerlink. This will probably change in the 3rd version of Powerlink, where the IEEE 1588 protocol is going to be introduced for ultra-accurate clock synchronisation. Hirschmann is one of the first companies to provide switches with support for IEEE 1588.

Because hubs are used, collisions have to be prevented as much as possible to be able to guarantee real-time behaviour. Powerlink assigns one device the role of “SCNM” (Slot Communication Network Manager) that determines which of the other devices is allowed to transmit. All transmissions are done per broadcast, giving each device the opportunity to communicate with all the others.

The requirement that no Ethernet switches may be used but only hubs is caused by the wish to be as fast as possible. Switches are slower than hubs, because every message must be received in full before it can be resent. This takes a double amount of transmission time, to which the switch internal processing delay must be added. In many applications this is not a problem, but for high-speed motion applications each microsecond gained translates into a higher application speed and higher accuracy. Additionally, due to the internal working of a switch it is unpredictable exactly how much variation in internal processing delay can be expected. This so-called “jitter” results in lower application accuracy, as it is never clear at which exact moment data has been sampled. Powerlink prevents these problems by disallowing switches and using hubs instead, although it is the only industrial Ethernet system that does so – it is interesting to note that, given identical requirements, different systems arrive at fully contradictory solutions.

Every Ethernet-user knows the rule that never more than 4 hubs may be connected in series. This rule limits the maximum size of the network, related to the timing constraints for detecting collisions. But because Powerlink uses Ethernet in such a way that collisions can never occur, the maximum-4-hubs-in-series rule no longer applies. According to B&R, up to 10 hubs can be connected in series, considerably extending the maximum size of a network.

As with most other high-speed motion networks, Powerlink executes I/O cycles of a fixed duration. Each I/O cycle consists of a real-time part and a non-real time part. TCP/IP messages can be sent in the non-real time part. Because TCP/IP has no knowledge of real-time networks, TCP/IP transmissions must be prevented from starting at any moment, possibly causing delays and / or network collisions. Powerlink does this by handling the transmission of TCP/IP messages by itself; large messages are split up in smaller ones (of 256 bytes max.) and individually transmitted; the receiver assembles the parts again. This functionality requires a special Powerlink Ethernet-interface. This is not really practical for PCs as many PCs have their Ethernet-interface integrated on the motherboard, to which no modifications can be made. It is therefore possible to communicate from a standard Ethernet interface to a Powerlink device, but this device can no longer participate in real-time data exchange.

6.5 IDA

The German “Interface for Distributed Automation” is a completely new development in the industrial Ethernet arena. ‘New’ here means no history, thus no backwards compatibility to older versions, and no technical compromises are necessary that limit functionality or performance. IDA therefore focuses totally on developing a very modern and efficient protocol. Figure 6-4 depicts the IDA architecture, based on TCP/IP and Ethernet, in combination with existing application protocols (FTP, http, SMTP, BOOTP, etc.) and a new set of functionality for handling remote I/O (NDDS), inter-PLC communication (Modbus/TCP) and safety applications. An application program runs on top of this that can access all this functionality.
The part of IDA responsible for the fast remote I/O handling is not new, but based on a product that has been commercially available for some time: NDDS from the US company RTI (Real Time Innovations, [www.rti.com](http://www.rti.com)). NDDS is a so-called “publish/subscribe” (p/s, also called producer/consumer) protocol, a way of working with networks not used in many existing industrial networks, but becoming more and more popular. The basic way of working in a p/s protocol is that a producer has knowledge about its own process variables (local or otherwise), and has a list of devices (subscribers) that use these process variables for their own purpose. Every time a process variable changes value, the producer informs all subscribers by sending them the new value. Each subscriber works with his own copy of the process variable. This is the reason why a p/s network is sometimes also seen as a “distributed database”, where every device has its own part of the database. The advantage of a p/s network above other types of network is the much lower network load, as network bandwidth is only consumed when a process variable changes; additionally all subscribers are informed about changes in a process variable at exactly the same moment. This makes synchronisation between devices very easy to implement. The high speed that results makes a p/s network of interest for motion applications.

Another important part of IDA is a well-known and very familiar protocol: Modbus/TCP. I stated in the introduction that the purpose of IDA was to develop a modern protocol, but with the introduction of Modbus this purpose seems to have been defeated. This is due to the admission of Groupe Schneider to the IDA User’s Group, which until 2002 only consisted of a group of smaller German companies. With the introduction of Schneider the IDA User’s Group became more viable, but at the expense of having to integrate Schneider’s own favourite protocol (Modbus/TCP).

At the end of 2004, it appears that the development of IDA has stalled. The IDA User’s Group has been merged with the (much larger) Modbus User’s Group, in effect IDA has lost its identity and for two years now not much publicity has been generated and no new technical developments released. Phoenix Contact has left IDA and joined with Profinet instead and Jetter, one of the original developers, has lost interest in IDA. The only development in 2004 was IDA being admitted to the IEC standardisation process as a pre-standard (IEC 62030). How IDA will develop further is not clear.
6.6 Foundation Fieldbus HSE

The “High-Speed Ethernet” version of the Foundation Fieldbus was one of the first Ethernet versions of a modern industrial network. Basically, it uses the existing FF H1 application layer protocol, with TCP/IP as transport protocol. This means that existing FF H1 functionality is available on FF HSE as well. FF HSE is not a replacement for H1, but an extension. On an Ethernet so-called “linking devices” can be connected that convert between FF HSE and the existing FF H1 protocol, which is perfectly capable to handle the communication with field devices like pressure, level, and flow transmitters. The linking device takes care of the protocol conversion from FF HSE to FF H1 and back, handles the difference in transmission speeds (100 Mbit/s vs. 31.25 Kbit/s), and handles the different electrical standards of Ethernet and FF H1.

The FF H1 speed of 31.25 Kbit/s, once thought (in 1988) to be sufficient for process control applications, has become too slow according to experts, who claim that not only process data, but also remote diagnostics, embedded web servers, maintenance and other functionality has to be present in a field device. This requires more bandwidth than a FF H1 can handle. Although the development of a new FF H2 system had already been started, the wise decision was taken to use Ethernet and TCP/IP instead of a proprietary development, and development on H2 was stopped.

The combination FF HSE and H1 results in a two-layer network architecture, connected via linking devices. It is also possible to connect a device directly to Ethernet. The usage of H1 allows existing FF networks to be connected to Ethernet, and also alleviates the need to develop an extension to allow devices to be powered over Ethernet (which is possible with Ethernet, but not for intrinsically safe applications).

6.7 EtherCAT

EtherCAT is a development by the German company Beckhoff, which released the first version in 2004 (www.ethercat.org). EtherCAT has some innovative new ideas that remove the two biggest drawbacks of Ethernet in industrial use: the cumbersome star wiring topology, and the protocol’s large overhead.

The way EtherCAT works internal is quite unique. Instead of sending separate messages, each with their own overhead, to all individual I/O modules, EtherCAT sends only one message for all devices. This single message contains all output data for all devices; each device extracts its own data. Each device sends its outputs to the master simultaneously. As an Ethernet message can have up to 1500 bytes (9000 bits) of data, this amount of I/O can be transmitted in a single message.

Beckhoff states the following figures for EtherCAT:

- 1000 digital I/O in 30 µs;
- 200 analogue I/O channels (of 8 bits) in 50 µs;
- 100 servos in 100 µs (= 20 kHz sample rate) with a jitter < 1 µs;
- 12000 digital I/O in 350 µs;
- In general: 10 Kbyte / msec.

EtherCAT I/O devices can have three connectors (in, out, branch) that allow for flexible cabling topologies. In principle all devices are daisy-chained, but branches can be made with the 3rd connector. This makes the network less vulnerable to cable disruptions or device removals, both well-known disadvantages of daisy-chain wiring. A hub or switch is not necessary, so the disadvantageous Ethernet star topology is not needed; additionally it makes an EtherCAT system cheaper than its competitors as no hub / switch need be bought.
A network can contain up to 65536 devices, over a distance of 500 km. The maximum distance between 2 devices is limited to 100 metres (limit of Ethernet CAT5 UTP wiring), or 2 km for fibre optics. Branches may be up to 10 metres in length.

In principle EtherCAT is a gigantic “shared memory” of up to 4 Gbyte in which each device can be allocated its own 64 Kbyte. Every device can map the memory of other devices in its own memory space; EtherCAT automatically takes care of the updates. In an identical way a device’s own data can be mapped by other devices; a change in a device’s own data is automatically reflected in the memory spaces of the other devices. This is a very simple way for a network to work, and also a very fast one – access to memory is available in every programming language.

Another reason for EtherCAT’s high speed is the hardware support for the protocol; no software is necessary and so software delays do not exist in EtherCAT. A special “FMMU” chip (Fieldbus Memory Management Unit) has been developed for this purpose. No FMMU is necessary in the controller; any standard (cheap) Ethernet interface can be used. This makes the implementation of EtherCAT possible on devices with cheap 8 or 16 bit micro controllers.

6.8 Sercos-III

Sercos is an industrial network specially designed for high-speed motion applications. It does not try to be a universally applicable industrial network, but just to be very good in controlling motion devices. The cabling structure (ring of optical fibre), functionality and speed are fully optimised for its intended usage. At the moment Sercos-II is the current version, also known as IEC-61491. The industrial Ethernet version of Sercos, called Sercos-III, is again fully optimised for controlling motion devices.

The hype surrounding industrial Ethernet in the first years did not cause any panic amongst the members of the Sercos User Group (www.sercos.de and www.sercos.org). Due to the inefficiency of Ethernet (large overhead) and the lack of specialised hardware it was always slower than Ethernet, even though Sercos-II ran at ‘only’ 16 Mbit/s and Ethernet at 100 Mbit/s. The Sercos vendors tried to remain pragmatic in a world in which a lot of other systems were promoting their ‘raw’ speed of 100 Mbit/s but did not mention their ‘net’ speed.

The creation of the special industrial Ethernet-protocols intended only for high-speed motion applications did cause more concern, as ProfiNet V3, Powerlink V3 and EtherCat could be considered competitors for Sercos-II, due to the technical innovations introduced in these systems to remove the Ethernet disadvantages. At the end of 2003 the Sercos User’s Group launched the Sercos-III project.

Important characteristics of Sercos-III are:

- Use of standard Ethernet wiring and connectors.
- Redundancy when using double ring wiring (optional).
- “Hot plugging” of devices (when using the double ring).
- Minimum cycle time 31.25 µsec (was 62.5 for Sercos-II).
- TCP/IP traffic possible.
- Communication between devices possible.
- Usage in safety-applications possible.
- Hardware-controlled synchronisation.

The way Sercos-III works internally resembles that of its predecessors. An added capability is the possibility of using TCP/IP in parallel with the I/O control. There is one disadvantage: because Sercos-III has optimised the Ethernet message structure, it is no longer compatible with standard Ethernet devices, which may not be connected in the same network as Sercos-III devices (but only via a gateway).
**Cabling**

In contrast to Sercos-II that uses fibre-optic cable, Sercos-III has specified the use of standard Ethernet twisted-pair cabling. This makes Sercos-III electronics interfaces about 50% cheaper than those for Sercos-II. Additionally, a special network interface chip has been designed, allowing “single chip drive” equipment.

What has remained identical between Sercos-II and III is the ring topology of the network. However with Sercos-III the ring is completely invisible because the standard Ethernet cable contains both the forward and backward channel of the network. A completely new feature is the capability to wire a double ring, giving redundancy: a broken cable, non-functioning cable segment or a faulty or missing device can be tolerated. This can also be used for application purposes: the so-called “Hot plugging” or “Live insertion / removal” of devices in a network is allowed, because the remainder of the network continues to function.

![Figure 6-5: The double ring of Sercos-III](image)

The ring topology of Sercos-III has the advantage that no switches are necessary. This gives Sercos-III a cost advantage over its direct competitors, but not for long, as these competitors (ProfiNet, EtherCat) are also aiming to allow this.

**Modifications to Ethernet**

Ethernet on its own is not well suited for high-speed motion application, due to the overhead present in each message: it always takes at least 84 bytes of bandwidth, even if only a few bytes of data are actually present. This is the paradox of “industrial Ethernet” – well suited for transmission of large amounts of data (as usual in an office environment), but in many industrial applications only small amounts of data are sent in each message.

Other Ethernet protocols suitable for high-speed motion (such as ProfiNet, Powerlink and EtherCat) have implemented several optimisations to become more efficient when sending small amounts of data. For Sercos-III a modification to the standard Ethernet message has been implemented:

- A normal Ethernet “Destination Address” takes 6 bytes, in Sercos-III only one byte.
- The same holds for the “Source Address”.
- Ethernet needs two bytes to encode the length of a message.
- Ethernet *demands* at least 46 bytes of data in each message, Sercos-III doesn't.
For the detection of transmission errors Ethernet needs 4 bytes per message, but Sercos-III only two bytes.

A simple example: if we want to transmit 8 bytes of data on Ethernet, a total of 84 bytes are sent. The efficiency is then approx. 10% (8/84). On Sercos-III a total of only 24 bytes are sent, giving an efficiency of 8/24 = 33%. With other words, at the same raw 100 Mbit/s network speed, Sercos-III can be three times as fast as some competitors. This directly translates into shorter cycle times, and thus also a faster control of all axes, allowing for higher speed or higher accuracy or a combination of the two. In comparison with Sercos-II the smallest cycle time goes down from 62.5 to 31.2 µsec. This is twice as fast, although a factor 6 might be expected at first sight. The reason for this 6-fold increase not being reached is the inefficiency of Ethernet; despite optimisations to the message format there is more overhead present than in Sercos-II, but the 100 Mbit/s Ethernet speed compensates a lot.

Of course there is a drawback to these special-format Ethernet messages: non-Sercos-III equipment will not recognize them. On a Sercos-III network only Sercos-III equipment can be connected. A connection to a standard Ethernet can be made with a special converter module. If this is not possible, all traffic to / from the standard Ethernet must be handled by the controller (PLC or PC).

The network cycle
A Sercos-III network always executes the same cycle (identical to Sercos-II). This way of working is no longer specific to Sercos, but is now also found in many other industrial networks such as Profinet and Powerlink. Each cycle consists of two parts; first the “cyclic data” is exchanged, and the second part is available for exchange of “non cyclic data” (TCP/IP). How much time the cyclic data takes depends on the number of axes, and the amount of data per axis; the user, depending on the expected amount of traffic, sets the time available for non-cyclic data. The total cycle time remains constant; if there is no acyclic data the network will just wait; if there is too much acyclic data that cannot be transmitted in one cycle, the remaining data must wait. Sercos thus guarantees that the cyclic data is always sent at predictable moments.

Following the cyclic data there is transmission time available for “normal” Ethernet messages, usually TCP/IP with one of its application protocols. The total amount of time available can be configured. Every communication cycle thus always takes a fixed amount of time; a sudden increase in TCP/IP traffic does not influence the cycle time and subsequently the application program does not notice a loss of communication speed; all axes are still controlled at the same frequency as before.

By working this way, Sercos requires no coordination between devices; if the configuration of the network has been properly calculated there can never be two devices transmitting at the same moment. On a standard Ethernet this would cause collisions, and extra delays before the collision is resolved. By completely preventing this, Sercos-III wastes no bandwidth on collisions, which translates into a higher speed (read: better control of motion devices).
Sercos-III to make a modification to the standard Ethernet message format. If this had not been done, Sercos-III would not be able to distinguish itself from its competitors, and because Sercos is traditionally only a motion-network this would probably cause its demise as the other systems are more general-purpose and thus have a larger market. By using the smaller messages Sercos-III is again faster than its competitors; table 6-1 gives a number of cycle times for a few scenarios.

<table>
<thead>
<tr>
<th>Data per axis</th>
<th>Number of axes</th>
<th>Cycle time</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>31.25 µsec</td>
<td>Torque / position</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>62.5 µsec</td>
<td>Speed / position</td>
</tr>
<tr>
<td>16</td>
<td>30</td>
<td>125 µsec</td>
<td>Speed / position / I/O</td>
</tr>
<tr>
<td>12</td>
<td>72</td>
<td>250 µsec</td>
<td>Speed / position</td>
</tr>
<tr>
<td>32</td>
<td>36</td>
<td>250 µsec</td>
<td>“Wide” interface</td>
</tr>
<tr>
<td>12</td>
<td>150</td>
<td>500 µsec</td>
<td>Speed / position</td>
</tr>
<tr>
<td>53</td>
<td>100</td>
<td>1 msec</td>
<td>100 axis</td>
</tr>
<tr>
<td>32</td>
<td>153</td>
<td>1 msec</td>
<td>“Wide” interface</td>
</tr>
<tr>
<td>16</td>
<td>254</td>
<td>1 msec</td>
<td>Max. number of axis</td>
</tr>
<tr>
<td>40</td>
<td>254</td>
<td>2 msec</td>
<td>Max. number of axis</td>
</tr>
<tr>
<td>65</td>
<td>254</td>
<td>4 msec</td>
<td>Max. number of axis</td>
</tr>
</tbody>
</table>

Table 6-1: Speed of Sercos-III in different scenarios (source: Sercos User's Group).

These figures are only valid when no TCP/IP traffic is sent. This would add considerably to the cycle times; for example a maximum size (1500 bytes data) Ethernet message takes 127 µsec transmission time.

### 6.9 Clock synchronisation protocols

A man with two watches... never knows the right time, a man with one watch does. This proverb is as valid for humans as it is for systems with multiple controllers each with their own clock (real-time or otherwise). It is not simple to set two clocks to exactly the same time; worse, once they are synchronised they inevitably drift apart. For example, two PCs can accumulate 10 seconds clock drift per day.

In many applications it is not so important to have the same time on all the clocks; usually the application is controlled by a main processor, and the application program uses only this clock for keeping logs, and for starting and stopping activities due at certain moments.

However in many motion applications, control of electricity distribution networks, test and measurement systems, and software development testing, accurately synchronised clocks are a necessity. For example, in distributed applications parts of the application programs run on different processors, and when something goes wrong the exact sequence of events must be reconstructed. If the logbooks of these processors are using unsynchronised clocks, it is usually very difficult to find the root cause of an event as it never clear what came first and what came second. A 1-millisecond clock difference between two systems may not seem much, but a modern 3 GHz processor does more work in that time than a first-generation PC in a whole second. With such processor speeds there is an ever-increasing need for higher-accuracy clock synchronisation.

### Drift

Even though clocks of different controllers can be set to exactly the same time, they will not keep the same time for very long. Under the influence of inevitable small deviations between the electronic circuits of each controller, the clocks start to drift immediately. Of course one could
Clock synchronisation protocols are not simple to understand, as they use various statistical algorithms to counter the inevitable and unpredictable delays found in operating systems, processors, hardware, and of course networks. But by not too complicated means it is possible to synchronise two clocks to within a minute. This is possible in Windows, with the command "net time \serverpc /yes" which sets your PC’s own clock to the time of the clock on "serverpc" somewhere on your LAN. If the time difference is more than one minute, Windows sets the local clock to the other clock immediately. If the time difference is less than one minute, the local clock is slightly increased / decreased in speed, and in due course will become synchronised. This gradual increase prevents sudden jumps in time perception, as many software applications cannot handle missing time events or a clock being turned back. Windows also supports more elaborate protocols for clock synchronisation. Identical possibilities exist on Unix and Linux systems.

Synchronisation to fractions of a millisecond
Using a network to synchronise clocks is not very difficult if a deviation of a few tens of seconds is acceptable (as in offices). But for industrial applications only a deviation of less than a millisecond is wanted. Using a network to accomplish this is not very easy, as the transmission of a message containing the current time itself takes a variable time, depending on the current load on the network, and the software processing time (dependent on the processor speed and application load) must be added to this time. If no correction is applied, the receiver’s clock is never more accurately synchronised than the sum of these two times. It is not easy to design a protocol that can run on any processor or any network and any application and that will still be able to maintain a high level of synchronisation.

In the last two decades various time synchronisation protocols have been developed, i.e. NTP (Network Time Protocol) and SNTP (Simple NTP). NTP has become a worldwide standard; a PC’s local clock can be synchronised with one (or more) “timeservers” connected somewhere on the Internet. The best (most accurate) timeservers are those of national standardisation organisations, which use atomic clocks. Many companies have their own time-server which synchronises itself via Internet, and they use this time-server for all their own systems (thus preventing a lot of unnecessary Internet traffic). It is even possible to build a hierarchy of timeservers; the lower in the hierarchy the lower the accuracy. A disadvantage of NTP is that it is quite slow in synchronising systems. NTP has security measures to protect itself against rogue time-servers that offer a wrong time. The NTP protocol is not simple, but for Windows and Unix systems there is an abundance of software (free and paid) to be found on Internet. The website www.ntp.org is the starting point for further information about NTP, and http://tf.nist.gov/service/its.htm is a good starting point for software.

Synchronisation to microsecond accuracy
The most recent development in time synchronisation is the standard IEEE 1588, also known as PTP or “Precision Time Protocol”. It was originally developed by Hewlett-Packard (now Agilent), together with several other US companies. The first version ran on Ethernet only, but PTP is now network-independent. PTP was especially developed for industrial applications requiring sub-microsecond synchronicity. The protocol itself is quite simple (when compared to NTP), allowing implementations on cheap (slow) processors. It also generates almost no network load; one network message per second suffices. Finally, it requires no configuration, and is thus very easy to use.

A fundamental difference with NTP is that PTP tries to eliminate as many software-processing delays as possible. The network interface module (hardware) of the sender of a message is re-
required to fill in the current time in the message, just before transmission starts. It is also advisable to use switches and routers that give PTP messages priority, so they are put in front of any queue of messages waiting for transmission, instead of at the end. Because the total delay is dependent on the size of the queue, this would add delays that PTP cannot anticipate. By putting the message at the front, the maximum delay is just one Ethernet message.

There is very little equipment supporting PTP at the moment. In September 2004 the second international congress of suppliers and users was held at the US National Institute of Standards and Technology (http://ieee1588.nist.gov). One of the first large-scale PTP applications is the control of electric distribution grids, which requires synchronisation accuracies between controllers of less than 1 microsecond. A first measurement of Hewlett-Packard in a simple network shows an average deviation of 22 nanoseconds (standard deviation 99 ns).

Many Ethernet-based industrial systems have already stated that they will use PTP: ProfiNet, EthernetIP/DeviceNet/ControlNet, and Powerlink. An implementation on LON is also known. Hirschmann and OnTime Networks have committed themselves to build switches supporting PTP. The NIST website (see above) is a good starting point to find further information about PTP developments and available products.

**Using GPS**

Of course there are different methods for synchronising clocks, for example via GPS (Global Positioning System) or special low-frequency radio transmitters. When correctly implemented, GPS can offer synchronisation differences of no more than 100 nanoseconds. However, giving every device its own GPS receiver can be costly. An intermediate solution is to use GPS on one device, and then distribute this time via NTP or PTP to other devices. When devices are geographically separated and not connected to the same network, GPS is probably the best solution.

**Disadvantages**

It is not always good when clock time is identical on many controllers. The author experienced this when synchronising clocks on controllers still using MSDOS. After installation, some controllers could not start anymore. After investigation it appeared that the network protocol DHCP was responsible; it needed to calculate a random number on all PCs that was used to uniquely identify certain network messages. As the random number generator used the real-time clock for this, identical clock settings on different PC’s meant that identical random numbers were generated. This caused DHCP to fail. Windows-based PCs do not have these problems.

**The description (simplified) of a clock synchronisation protocol**

A simplified clock synchronisation algorithm is described in the figures below (source: IAONA Switzerland). One device has the ‘master clock’, and this master wants a slave device to synchronise its clock with the master’s. This is done in three steps. At the beginning (figure 6-7) we see a 2.6 units time difference between the two devices.
The first step in the clock synchronisation process between a master and a slave is taken by the master, who sends a message to the slave containing the time at which the message itself is transmitted, say in this example $t_m=1$. Suppose that the slave receives this message at $t_s=4.8$ (his time). It is thus clear that there is a time difference between the master and the slave. But exactly how much is not known, or not yet, because a message can be 'in transit' between the master and the slave for some time (message transit delay).

After the slave has received the master’s message at $t_s=4.8$ (his time), the slave will set this clock back to the time specified in the message ($t_s=1$). This has an immediate positive effect, because both clocks now deviate by only 1.2 units, in comparison to the previous 2.8 units. However there is still a difference between the master time and the slave’s time; this difference is the message transit delay. Another correction is necessary.

After processing the message, the slave will transmit a message back to the master containing the local time ($t_s=2$). The master now knows how much time has passed between the moment
of sending the original message, and the receipt of the second message. This can be used to calculate the message transmit delay. The message is received at \( t_m = 4.4 \) and the original message sent at \( t_m = 1.0 \), so 3.4 units were necessary for a whole cycle (two message transmit delays plus the slave processing time). The slave itself needed 1 unit processing time: the difference between the reception of the original message (\( t_s = 1 \)) and the moment the message to the master is sent (\( t_s = 2 \)). The master deducts this from 3.4, so that it now knows the total message transmit delay is 2.4 units for two messages, or 1.2 unit per message. The master sends this value to the slave, which corrects its own clock again (with 1.2 units). Both the master clock and slave clock are now synchronised.

![Master clock diagram]

Figure 6-9: After the third step both the master and slave clock are synchronised.
7 EQUIPMENT

7.1 Gateways
In many applications Ethernet is used together with another, industrial, network. It is often required that both networks are connected together. This can be done via a controller supporting two network interface modules, but also by standard available equipment called “gateway”. Actually, “interpreter” would be a better word to describe the functionality such equipment provides; in some networks identical functionality can also be called “protocol converter”, “proxy” or “linking device”.

A distinction can be made between two different types of gateways:
- “Shared memory” gateways, connecting Ethernet to remote I/O networks.
- “Interpreting” gateways, for connecting Ethernet to any type of network.

The difference between both types of gateways is in the way they work.

Interpreting gateways
An interpreting gateway literally translates each incoming message according to one network protocol into an outgoing message according to the other protocol with identical meaning. This looks easy, but isn’t – while in natural languages most words have equivalents in the other language, in networks this does not have to be so. The consequence is that sometimes only a small (common) subset of both network protocols is supported, or only the most often used commands. Sometimes the user can program the gateway by himself, allowing an exact match between application requirements and gateway functionality. The drawback is that one needs good knowledge of both network protocols, time that might be better spent on one’s own application.

The disadvantage of interpreting gateways is the processing time involved on the processor of the gateway, and the double transmission time (for both network messages). When connecting Ethernet to a remote I/O network it is therefore better to use a “shared memory gateway” (see next section), which provides much higher speed.

Shared memory gateways
Theoretically, a shared-memory gateway is not really a gateway, as it does not perform a network protocol translation between the two connected network interfaces. But nevertheless a gateway that works in this way is very popular, both for users and for vendors, as it is easy to implement and also easy to install and use.

The communication between the two network interfaces is done via a ‘notice board’, an area of memory where both parties can write to, and read data from. Usually the memory is split into two parts. Network interface A can write to the first half but only read from the other half; network interface B can write to the second half but only read from the first half. This prevents A and B from overwriting each other’s data. Apart from this common (shared) memory there is no other communication between the two network interfaces.
This way of working fits remote I/O systems very well. The two halves of the shared memory are used for inputs and outputs, respectively. New values for the outputs that arrive via network interface A are written, and read by network interface B who then executes the necessary commands on his own network to ensure that the outputs are set to the new values. The inputs that B reads are written in the second half of the shared memory, where A can read them at any moment. Both A and B can work independently of each other; the memory is always accessible. As only the user data is transferred, A and B do not have to worry about protocol-specific details that the other side might see; there is only data. Because every industrial network has functionality for handling I/O, it is easy to implement a shared memory gateway.

A disadvantage of shared memory gateways is that the memory has no memory. This might sound strange, but is not incorrect: if new I/O data is written to the shared memory, the 'old' data is lost. For I/O data this is usually not a problem, because there is no interest in 'old' data, only current data is wanted. Of course, the application program should be designed in such a way that the 'loss' of data does not cause problems due to missed input changes, or output changes that are 'overwritten' because the network is not fast enough. This is not typical of Ethernet; it can happen with any industrial network, but with gateways the calculations are more difficult because two networks are involved.

A shared memory gateway is also at a disadvantage when commands, or configuration data, errors, statuses, etc. are to be transmitted. The difference with I/O data is that they may not be overwritten unless the recipient has seen the command / configuration / error etc. Only then may the shared memory be overwritten. Usually a handshake is necessary, which must be programmed by the application(s) running on controllers on both networks.

The advantage of shared memory gateways for connecting remote I/O to Ethernet is that the high speed of the existing remote I/O systems is available. Ethernet is also used in a high-speed mode, because all remote I/O data can be sent in two large Ethernet messages, instead
of many small messages that give an enormous overhead. Additionally, if multiple gateways are connected, there is an advantage of having multiple remote I/O networks working concurrently, again providing higher I/O speed.

7.2 Serial Device Servers / RS232 Gateways
RS232 has always been one of the workhorses of industrial automation, because it is usually “free” (via a standard port on controller or PC), it is simple, and flexible in connecting all sorts of equipment together. This is about to change soon, because Microsoft has decided that RS232 is no longer a mandatory interface on PCs. It will be replaced by USB. Nowadays, many a new laptop doesn’t even have an RS232 interface anymore.

A disadvantage of RS232 is that only a small distance (15..50 meters) can be wired between two devices. This doesn’t make it an interface for building a network; only two devices can be connected “point-to-point” together.

Both disadvantages can be solved by using Ethernet-based “serial device servers” (figure 7-2). In combination with a software driver, a PC can still have RS232-support, although no longer in the PC, but somewhere else in a building, machine, installation, or elsewhere on Internet. By using Ethernet, the limited distance of RS232 is no longer relevant either. Additionally, it is easy to support a PC with dozens of RS232 interfaces; no plug-in cards are needed and there will be no shortage of motherboard slots. This makes very small (embedded) PCs possible.

The serial device server is placed close to the location where the RS232 connection is physically necessary. It can then be connected to Ethernet. Serial device servers with “power of Ethernet” functionality (now coming on the market) don’t even need a power supply. On the PC the “COM-port redirector” will simulate a COM3 (or other) interface; the application program will not notice that there is no hardware present in the PC itself. All the commands it sends to the COM3 interface are intercepted by the COM-port redirector, transmitted over Ethernet to the
server, which controls the real hardware. Any incoming data is transmitted over Ethernet to the PC, accepted by the COM-port redirector and then passed on to the application program.

Of course this only works when the COM-port is accessed via official Windows functionality. Application programs that directly access the hardware, or employ undocumented Windows tricks (usually to increase speed) will no longer work with a serial device server, as the hardware is no longer present in the PC. It is usually difficult to determine in advance whether a given application will continue to run correctly together with a serial device server; it is best to always try this out in advance. Note that only the transmit/receive data functionality must continue to work, but also the modem signals (RTS, CTS, RI, DTR etc.) present on an RS232 interface.

Using a COM-port redirector is of course only possible on PCs. To allow the use of serial device servers to non-Windows equipment, it is usually possible to control the server directly via TCP/IP functionality. On the controller there should then be the capability to program TCP/IP directly; usually this is called a “socket interface”. Sockets were originally developed for Unix, but are also available on Windows (“WinSock”). Working with sockets requires knowledge of TCP/IP and writing network drivers, but it is not very difficult; ample literature is available. The capabilities of a serial device server can be much greater than discussed in this section. As this is not really Ethernet-related, it is not relevant to this publication; several suppliers of serial device servers have elaborate documentation available on Internet.

7.3 Analysers, monitors, spies and sniffers
Anyone working intensively with an industrial network will sooner or later run into problems whose cause is not very clear. Just as a technician uses a multimeter or an oscilloscope, the network expert has his “network analyser”, “sniffer”, “spy” or “network monitor” (four words for the same type of device). This equipment shows which devices are connected to the network, which network messages they are sending each other and when, which data those messages contain, which errors occur, etc. Analysers for Ethernet are in abundant supply; ask your local (office) network manager to show it, probably he'll have some equipment or software packages in daily use. Equipment to check on the electrical quality of cabling is also quite common (i.e., as sold by Fluke).

Foundation Fieldbus analyser
The disadvantage of all the office-LAN products is that they were not designed for networks and protocols common in industrial automation. Special analysers have been developed for industrial Ethernet and the associated protocols only as an exception, for example for Foundation Fieldbus’ HSE (high-speed Ethernet) version. It is called the “HSE Toolkit”.

Because this kind of equipment is not sold in large quantities, it is not cheap: a license for the HSE Toolkit costs initially $6750, and another $1000 for each additional license. This is expensive even for a network analyser, given the fact that it is only a software-package without the hardware (PC to be provided by the user). But these expenses have to be offset against the costs of downtime of an installation. Additionally a network analyser can help in finding faults in application programs, determining response times, and helping to find interoperability problems between devices of different suppliers. From my own experience I know that any network analyser justifies its purchase price by the first problem solved; once I had five different network analysers (for 5 different protocols) and the amount of time saved in solving problems or gathering evidence of faulty behaviour of equipment (to solve disputes with suppliers!) was enormous.

The HSE Toolkit can be installed on any ordinary Windows-based PC with an Ethernet interface (no problem nowadays). This is possible because FF HSE uses standard Ethernet; only the protocol is different. Because standard hardware is used, the HSE Toolkit cannot measure the cabling quality or signal quality of an FF HSE network; if there are serious cabling problems that prevent the arrival of network messages on the HSE Toolkit PC, it cannot show them to the
user. This is a limitation of any software-based network analyser. In order to determine the cabling problems, more expensive equipment is needed, but this will not be able to help with FF HSE protocol problems. So usually two network analysers are needed for Ethernet in practice; one for the wiring of the network, and one for the protocol.

**Ethereal**

In many cases it is not problematic to use a ‘standard’ network analyser that is not aware of all protocol-specific or application-area specific details. Software developers especially, who can often dream in hexadecimal, find that it is not a disadvantage to work with low-level network analysers. Sometimes it is even an advantage that an analyser does not give its own interpretation of the facts, because it is now exactly clear what happens, when, by whom, and what data is involved; no interpretation by the analyser is added. Of course, a knowledgeable user is needed.

An example of such a network analyser is “Ethereal”, a freeware package that can be downloaded from [www.ethereal.com](http://www.ethereal.com) (figure 7-3). Interesting for us is the fact that it has free protocol modules for IEEE 1588, Ethernet/IP, Powerlink and Modbus/TCP.

![Ethereal screen dump.](image)

In order to be able to work easily with Ethereal, it is a good idea to buy a small Ethernet hub. Ethereal is capable of receiving all traffic send to / transmitted by a certain device, or (even better) by all devices on the network but only via a hub. Why is the hub necessary? When Ethereal is connected to a free port on a switch it will only receive broadcast traffic, but nothing else. The best solution is to use only switches with support “port mirroring” functionality (see chapter 5), but my experience is that such switches are not installed at many sites. And even if they are, the port mirror must be enabled first, something usually only network administrators can do (knowing the switch password).

The network interface card on the system on which Ethereal is installed must have support for “promiscuous mode”. This allows it to accept any network messages it receives, instead of only accepting those network messages whose destination MAC-address matches its own MAC-
address. Most modern Ethernet network interface cards nowadays support promiscuous mode (despite the security leaks it causes!).

**Microsoft Network Monitor**

Microsoft sells its own “Network Monitor”, together with the ‘server’ version of Windows NT, 2000 and 2003. It has the advantage that it is better integrated with the Microsoft platform and the protocols used on it. In contrast to Ethereal, which also runs on Linux, the Microsoft product only runs on its own platform.
8  TIPS & TRICKS

8.1 Adding devices
It is relatively easy to add an extra device to an Ethernet: find a hub or switch with a free port, and connect the new device to this port. Because each Ethernet device has a unique MAC-address worldwide, in principle there can be no address conflicts (unless you assign the MAC-addresses yourself).

If the hub / switch supports “auto negotiation”, both the hub / switch and the new device choose their highest common speed, and as soon this has been done the new device is ready to use the network or can be reached by others. In practice however it is not always so easy; the higher protocol layers may have their own requirements regarding their local configuration and start-up sequence. It is thus not always guaranteed that an application program on the new device can be started. It is very difficult to give some general guidelines applicable to every network protocol; refer to the protocol documentation for detailed information about the conditions regarding configuration and start-up.

8.2 Removing devices
A device may be removed from an Ethernet at any moment. Ethernet itself does not execute any checks on the presence or absence of devices, and thus also gives no warnings of errors. When this functionality is really needed, it has to be done by higher-levels protocols. Most modern systems have this functionality, but don’t be surprised when the reaction times are very slow (i.e., Windows NT may take up to 12 minutes to report removal of a device). Refer to the protocol documentation for detailed information about the possibilities.

8.3 Replacement of defect devices
A device in any system can break down, and may have to be repaired and later put back on the network. Sometimes the repaired device does not become active on the network if another port on the hub / switch is used. This phenomenon is caused by a subtle difference between the internal operation of a hub or a switch.

When a hub is used, the repaired device can become active in the network immediately. It will receive all network messages sent to it. But when a switch is used, the repaired device will not receive any network message for several minutes. Why? The switch doesn’t “know” that the repaired device is now connected to another port, so it will continue sending all the network messages to the original port. Only after the timeout in the switch table for this port has expired, will the administration about the original port be removed, the next message will then be sent out over all the ports, and only delivered on the repaired device.

This situation can be easily prevented: have a device send a network message (any will do) immediately after its start-up. The switch then immediately “knows” the new port the repaired device is connected to. For example, when TCP/IP is supported on the device, use the command "ping"; to force a network message to be sent. It doesn’t mind to whom, as long as the switch sees it. Unfortunately, not all network protocols allow a device to send a message after start-up. For example, a device designated as a slave in a master/slave network may never autonomously send a network message; it must wait for a message from the master.

Ethernet does not specify the timeout values to be used in switches, always refer to the supplier’s documentation for detailed information about how the switch behaves. If there is no pos-
sibility to enforce an (automatic) and fast port assignment for the repaired device, then there are only two possibilities left: the first is just to wait, the second is to switch off/on or reset the switch (but this stops the entire network for a short interval, it depends on your application whether this is allowed or not).

8.4 Assignment of TCP/IP addresses

When the TCP/IP protocol is used, devices must be assigned an “IP address”. On industrial Ethernet this is done identically to office networks. In the last two decades, various methods have been developed for assigning IP addresses:

- Static assignment via dipswitches, keyboard, etc.
- Dynamic assignment via the BOOTP protocol;
- Dynamic assignment via the DHCP protocol;
- Static assignment via a built-in web server;
- Static assignment via the TELNET protocol.

Usage of BOOTP or DHCP requires a corresponding server, usually a PC. Software for these protocols is abundantly available on Internet, and Windows’ Server versions have a DHCP server standard available as well; Windows workstations have a DHCP client.

When Telnet is used, you will get a command-line-driven interface to the device (as with DOS in the early years of PCs). The device will have a default IP-address set by the supplier; otherwise not even Telnet-access is possible. You need a Telnet-client to talk to the device; this software is available on Windows and all Unix / Linux systems as a standard feature. If you have a system for which no Telnet-client is available, just use a PC instead—it is not important where the Telnet-client resides for the device to be configured. Remember to set a password on the device; otherwise everyone with Telnet-access can modify the device’s settings.

DHCP / BOOTP

The DHCP (Dynamic Host Configuration Protocol) is the current standard way to deliver network configuration to a device via the network itself. A simpler (older) version of DHCP is BOOTP (Bootstrap Protocol), which is less flexible but in many cases this is no problem at all. Both protocols execute immediately after a device is started. The device will send a broadcast message, requesting DHCP (or BOOTP) configuration. This message contains the MAC-address of the device. Any listening DHCP (or BOOTP) server on the network receives this (and other) message(s). Previously, these servers will have been configured to recognize certain MAC-addresses, and will have been instructed which IP-address or addresses may be assigned to the known devices. The DHCP (BOOTP) server will send back the IP-address assigned to the device, together with other necessary TCP/IP network configuration. If the MAC-address is not known, the device will be ignored. It is important to remember that the MAC-address is the key element in this way of working.

The trouble with this scheme, which is perfectly acceptable in office environments, is that any change in a MAC-address of a device will lead to either no network configuration being sent, or an IP-address whose value cannot be predicted. A change of MAC-address can occur when a device is replaced, or its network interface module is exchanged for a different one. As on Ethernet it is customary that the supplier of the device, and not the end-user, sets the MAC-address, this will sometimes cause new MAC-address to be introduced in a network.

With DHCP / BOOTP servers that ignore unknown MAC-addresses, the solution is simple: update the administration of the servers. But that this manual action is necessary needs to be remembered; it is necessary after MAC-address changes in the network. Although not difficult, it makes “plug & play” replacement of hardware impossible. An alternative solution - having the DHCP / BOOTP server accept any possible MAC-address - is not a solution, because the server must “know” who’s who in an application – which IP-address must a device get? How is
the device to know who it is and what function it must perform? Usually assigning fixed IP-addresses to known devices is enough.

**MAC-address reprogramming**

Some suppliers offer the capability for users to program a (new) MAC-address in a device. When a device is replaced, the MAC-address is just copied from the old to the new device; the fact that a new device is connected to the network is not even visible to the DHCP (or BOOTP) server: all they care about is the MAC-address in use. No change to the DHCP (or BOOTP) server is then necessary.

Note that the ‘old’ device may never be connected to the network anymore, as this will lead to a MAC-address in use at two devices. Very strange errors may result before this is noticed and more time before the error is solved. Setting your own MAC-addresses offers an easy way out of network configuration issues, but requires a meticulous administration of MAC-addresses and a strict discipline about who is allowed to connect devices to a network.

Within the industrial Ethernet community, there is no agreement about the use of DHCP or BOOTP; one vendor uses DHCP, the other uses BOOTP, and (even worse) for Profinet a private protocol has been invented. For end-users this is cumbersome, as both a DHCP and a BOOTP server may need to be installed on the network.

8.5 Breakdown of a hub or switch

If a hub or switch stops functioning, the consequences for the network, or part of it, are large: all network traffic to / from the connected devices ceases. The breakdown may be caused by the electronics components in the hub or switch, but a very important source of failures is the electronics power supply, due to the high voltages (220V), high currents, and high temperatures that occur in power supplies. In industrial hubs and switches the supplier can take various measures to increase the up time of his products; a simple solution is to have the option of connecting a redundant power supply (usually 24V).

For the controller it is usually of interest to know whether both power supplies are still operational, or whether the hub / switch is already running on the back-up power supply. Many vendors supply one or two potential-free (relay) contacts that can be connected to a remote I/O module's digital input. By reading the value of the input the controller knows the state of the hub / switch. This is a relatively primitive method of conveying status information; it can be argued that the switch could just as well send a network message with the status information to the controller. Technically this is indeed possible, but commercially it is not – since there are so many different network protocols, the switch would have to have support for many protocols, which is expensive. The most ‘portable’ solution is thus to work with potential-free contacts.

A solution that is in the offing is the use of the SNMP (Simple Network Management Protocol) for passing the power-supply status (and much other information). SNMP is the standard protocol in office networks for managing networks and all connected equipment. Ask your own company’s network managers for a demonstration of the network management software, and remember that even though it may not be visible, SNMP is probably the source of all information shown. Many vendors of industrial Ethernet equipment are now starting to introduce SNMP into their equipment as well; check for the so-called “managed switches”.

8.6 Replace a hub by a switch

Despite the many advantages of switches, sometimes using hubs is still attractive (see chapter 5). It is very possible that after some time the higher speed of a switch is needed. Usually it is possible to just take out the hub, insert the switch, plug in all the cables, switch the power on, and continue, without requiring changes to the application program. In situations where it is financially or technically attractive to use hubs it is thus convenient to do so, without sacrificing future extensibility.
8.7 SNMP network management

On its own, Ethernet has very little functionality for network management, apart from one or two LED’s blinking in the vicinity of every RJ45 connector or for every port on a hub or switch. A more advanced form of network management is possible via the SNMP protocol (Simple Network Management Protocol), one of the members of the extensive TCP/IP family of protocols. In office networks SNMP is the “de facto” standard for network management. It is now also being used in industrial networks.

As the name says, SNMP has been especially developed for network management. The first version of SNMP was indeed simple, but this is no longer so in SNMP V3. The protocol is not specifically meant for Ethernet, but for TCP/IP and all supported hardware and software, and this is why Ethernet network management can also be done via SNMP. Support for SNMP in devices is not mandatory; some suppliers have it while others don’t. Usually the word “managed” in a product description (i.e., a “managed switch”) means that support for SNMP is built in. This kind of equipment is usually more expensive than “unmanaged” versions.

SNMP allows a device to return management information about its own operation; it is also possible to configure devices with SNMP. In official SNMP documentation a list of “objects” is usually given, most of which can be read, and some of which can be written or overwritten. The use of the word ‘object’ does not mean that object-oriented programming is necessary; the two have nothing to do with each other. Personally I prefer to use the word “field”.

SNMP lists a huge set of fields in its standardisation documents. Suppliers can also add their own fields. All fields are administered in a “MIB” (Managed Information Base). For example, Phoenix’ MMS-switches support all mandatory SNMP standard fields (some 150), the fields for network management (60), the fields for switch management (50), and a group of 130 Phoenix-specific fields related to the MMS. Some fields may be used to configure the MMS switch: speed of a port, use of redundant cabling, timeouts, VLAN use, etc. An SNMP-client elsewhere on the network is necessary for this. As this is not always common, vendors usually support other methods of setting network configuration in a switch, i.e. with a Telnet-interface or a local web server.

The capability of supporting SNMP in a switch requires it to have a so-called “SNMP Agent”, an ‘agent’ being a software component that runs autonomously. The task of the agent is to collect all management information from the electronics circuits of the switch, check for changes, and send it back to anyone asking for it via the SNMP protocol. Additionally, the agent is allowed to send certain fields unsolicited, i.e. after a change of value of a field. An SNMP-client does not continuously have to read a certain field just to keep track of the changes. Having the agent monitor a field for changes and send it when changed is much easier, and substantially decreases the load on the network. The unsolicited transmission is called an “SNMP Trap”. Somewhere on the network there is usually a device waiting for any incoming SNMP Trap. What is to be done next, depends on the application – the removal of a device from the network, for example, or the insertion of a device, could be signalled to the application program.

Which variables are listed in a MIB is partly standardised, and partly free for suppliers to determine. There is a list of the meaning of the field, its size (in bytes), and the specific properties for all standardised fields. For example, field (object) 1.3.6.1.4.1.4346.11.11.4.1.5 is an Integer32, read-only, and indicates whether the IP-protocol parameters have been stored into non-volatile memory (value 1) or have not (value 2). Each vendor is required to list in his documentation which fields are in his MIB. Knowledge about the SNMP mandatory fields is usually assumed to be present; if it is not, reading the SNMP protocol specification may help and a visit to the website www.snmp.org/protocol or www.simpleweb.org gives a good introduction, as well as information about software and the various SNMP versions.
9 MORE INFORMATION

9.1 Literature

Ethernet standard
The Ethernet standard is officially known as IEEE 802.3. The IEEE (Institute of Electrical and Electronic Engineers) decided in 2001 to allow free access to the standard (and all other 802 standards) for everybody six months after publication of an update. Surf to http://standards.ieee.org/getieee802 for a download of the document in PDF-format.

Ethernet definitive guide
More readable is the book “Ethernet, the definitive guide” by Charles Spurgeon (ISBN 1-56592-660-9, $50), which discusses in depth the various Ethernet wiring possibilities, with all their pros and cons. It is one of the better Ethernet books; it is purely focused on wiring and does not discuss software, Windows or TCP/IP. Unfortunately the industrial aspects of Ethernet are not discussed, but nevertheless I really recommend this book.

Pocket Guide
A small book about industrial Ethernet has been written by Perry Marshall and is titled “Industrial Ethernet: A Pocket Guide”. According to the table of contents an impressive list of subjects is discussed, but given the size of the book (200 pages of A7 size) most subjects are only marginally discussed. The title “Pocket Guide” really has to be taken literally. ISA Press, ISBN 1-55617-774-7, $35. More information about the book’s contents can be found at http://www.perrymarshall.com/ethernet/.

Fuller book
One of the first books about industrial Ethernet was written by F.J. Fuller, and entitled “Ethernet-TCP/IP für die Industrieautomation” (in German). The author uses simulations to try to determine the circumstances in which Ethernet is acceptable for real-time systems, and what the worst-case performance will be. Publisher is Hüthig Verlag, ISBN 3-7785-2641-3. In the second version of the book (ISBN 3-7785-2779-3) new subjects such as “quality of service” features, IPV6, redundancy, safety and software architectures are added.

Ethernet Planning & Installation Guide
The IAONA (Industrial Automation Open Networking Association, also see below) has issued the “Industrial Ethernet Planning & Installation Guide”. As the title indicates, the document describes the planning and installation of Ethernet, focused on industrial aspects. It is not entirely new; the existing standards EN 50173 en IEC 11801 are as valid for industrial Ethernet as for any office-based Ethernet. The architecture of a (large) network is described in four chapters, followed by installation rules, wiring validation measurements, calculations on the maximum size of a network, and specific aspects of twisted-pair and optical fibre installations. The document (as PDF) is available as a free download from www.iaona.org. Note: it is assumed that readers
are familiar with the theory of Ethernet, what hubs / switches and routers are, what the various IEEE 802.3 standards are, etc.

**Articles**
Every year (5th time in 2005) the German publisher Vogel Verlag publishes a collection of all the articles about industrial Ethernet that have appeared in its trade magazines. The original articles (in German) are translated to English, and bundled with the original German articles in a single volume. It is a very efficient way to learn quickly about current developments in industrial Ethernet in Germany, without having to learn German. Size usually some 100 pages, ISBN 3-8259-1925-0, 21 Euro.

**ProfiNet specification**
The ProfiNet User’s Group (PI – Profinbus International, [www.profibus.com](http://www.profibus.com)) by way of exception allows anyone, even non-PI members, to download the ProfiNet 2.0 specification free of charge,. Note: the document is not meant for end-users, but for software developers; a good knowledge of the Microsoft Windows is required.

**Industrial Ethernet Book**
The quarterly “Industrial Ethernet Book” ([http://ethernet.for-industry.com](http://ethernet.for-industry.com)) has been appearing for 5 years now. Despite its name it is a magazine, filled with technical articles about current developments in industrial Ethernet. It also contains a catalogue of industrial Ethernet products. A subscription to the magazine is free.

### 9.2 Associations
At the moment there is only one association active in the area of industrial Ethernet, and that is IAONA - Industrial Automation Open (read: Ethernet) Networking Association. Originally the IAONA was founded in the US and had a branch in Europe, but the latter soon overshadowed the US mother, which now no longer exists. The Swiss IAONA ([www.iaona.ch](http://www.iaona.ch)) is very active; its website contains much interesting material.

The IAONA website can be found at [www.iaona.org](http://www.iaona.org), where you can find the industrial Ethernet cabling guidelines as developed by one of its committees (see above). At the end of 2003 another committee published the security guidelines document, but there has not been much progress in this area since.

The IAONA itself does not develop new networking protocols, but only tries to coordinate the developments of the various user groups (Profibus, ODVA, Foundation Fieldbus, Powerlink, Modbus, etc.), for example in the field of connectors, wiring, security, real-time aspects etc. In some cases the IAONA has been quite successful, but not always – there are now 10 different industrial Ethernet connectors, and the working group “Real-time Protocols” has been overwhelmed by the fast developments in the user groups.

### 9.3 Trainings and seminars
Seminars about industrial Ethernet are given by many large vendors, ranging from a 1-day introduction to a full-week in-depth discussion of Ethernet, its wiring, its working, management, troubleshooting, etc. A distinction should be made between seminars discussing Ethernet as a fieldbus, and seminars discussing the use of Ethernet in industrial applications (i.e. video surveillance of plants), which are quite different things.

Companies active in office automation also give seminars about Ethernet. Although the technology of Ethernet is identical, the industrial aspects are not known and the contents of the seminar will focus more on “software” (protocols, configuration, Windows, maintenance) than on the “hardware” which is seldom an issue in office automation anymore.
Seminars about Ethernet often discuss subjects that have nothing to do with Ethernet, such as TCP/IP and its family of protocols. If you are really interested in TCP/IP, it is better to attend a TCP/IP seminar, instead of a seminar in which TCP/IP is only a small part.

The German company ComConsult [http://www.comconsult-akademie.de](http://www.comconsult-akademie.de) is worth keeping an eye on. Apart from their seminars, they have a very interesting (free) monthly newsletter, and also do product evaluations, the reports of which are made publicly available. For example, in 2004 the Hirschmann’s redundant switches and Siemens’ wireless Ethernet access-points were evaluated.

### 9.4 Websites

Information about industrial Ethernet on Internet is easy to find; just use Google to search on “Ethernet industrial ppt”!

The problem with listing URLs of websites is that they age very quickly and in only a year many references to websites will become outdated. Because of this, the author keeps an up-to-date website with references to a few hundred networks at [http://ourworld.cs.com/rahulsebos](http://ourworld.cs.com/rahulsebos).
INDEX

10BaseFx ........................................... 13
10BaseTx ......................................... 13, 28
10Base2 ........................................... 12, 28
10Base5 ........................................... 12, 28
10BaseFl ........................................... 13
10BaseT ........................................... 12, 28
10GBaseT .......................................... 15
Adapter address .................................... 19
Allen-Bradley (company) ......................... 12, 73
Alteon
   Jumbo frames .................................... 21
Auto-crossing ..................................... 61
B&R (company) .................................... 73
Beckhoff (company) ............................... 76
Bluetooth ......................................... 13, 15
BOOTP (protocol) .................................. 92
Broadcast storm .................................... 57
CAN
   Speed comparison ................................ 40
CAT3 .................................................. 13
CAT5 .................................................. 13
CAT5e ................................................ 13
CheaperNet ........................................ 12
Clock synchronisation
   In Windows ...................................... 81
   NTP ................................................ 81
   Protocols ........................................ 80
   SNTP ............................................. 81
   with GPS ........................................ 82
Collisions
   Occurrence .................................... 32
   Prevention ...................................... 34
ComConsult (company) ............................ 97
Company ID ....................................... 16
COM-port redirector ............................... 88
Connecting equipment .......................... 60
Connector
   Extra power supply ............................ 36
   IEC 61076-3-101 standard ..................... 36
   Industrial quality ............................ 35
   M12D variant .................................... 36
   RJ45 compatible ................................ 36
   RJ45 standard ................................... 35
ControlNet (protocol) ............................ 73
CRC
   Cyclic Redundancy Check ....................... 20
   Crossover cable ................................ 45, 60
   Cut-through switches .......................... 49
   Deterministic behaviour ....................... 32
   DeviceNet (protocol) ........................... 73
   DHCP (protocol) ................................ 92
   DIX Ethernet ................................... 11
   EGD
      Ethernet Global Data (protocol) ............ 67
   EN 50173 .......................................... 95
   EtherCAT (protocol) ............................ 76
   Ethereal ......................................... 89
   Ethernet Global Data .......................... 67
   Ethernet/IP (protocol) ........................ 12, 73
   Fast STP ......................................... 65
   Firewire ......................................... 10
   Flood control .................................... 57
   Flow control ..................................... 56
   Foiled Twisted Pair ............................. 25
   Foundation Fieldbus ............................ 88
   Foundation Fieldbus HSE (protocol) ........ 76
   FRNT
      Fast Reconfiguration of Network Topology .. 65
   FTP ................................................ 25
   Fuller, Frank (author) ......................... 95
   Gateway ......................................... 30, 85
   Shared-memory ................................... 85
   To RS232 ......................................... 87
   GDA ............................................... 18, 59
   Group Destination Address ..................... 18, 59
   HIMA (company) .................................. 12
   HiperRing ........................................ 65
   HSE Toolkit ...................................... 88
   Hub
      Functionality .................................. 46
      Internal structure ............................ 47
      Manageable .................................... 13
      Number of ports ................................ 46
      Operation ..................................... 46
      power supply .................................. 27
      Powerlink use .................................. 74
      Uplink port .................................... 61
   IAB ............................................... 16
   IAONA .......................................... 96
Network monitor...... See: Network analyser
Network sniffer...... See: Network analyser
Network spy....... See: Network analyser
Network Time Protocol........................81
Organisationally Unique Identifier . See: OUI
OUI........................................15
Pair in Metal Foil............................25
PAUSE bit.....................................57
Physical address............................19
PIMF...........................................25
Planning & Installation Guide...........95
PoE............................................27, 36
Port mirroring...............................56
Port selector...............................48
Port trunking...............................65
Power hubs..................................27
Power over Ethernet........................27
PowerDsine (company).....................27
Powerlink.................................43, 73
Preamble.................................20
Precision Time Protocol...................81
Priority tag..................................55
ProfiNet......................................43
2.0 specification............................96
Component Descriptions..................71
Description.................................69
Engineering.................................71
Interbus integration--------------------72
IRT version................................69
Line topology..............................30
Machine distributor.......................70
Proxy module...............................72
SRT version................................69
Versions.....................................69
Promiscuous mode..........................22, 89
Protocol converter.........................85
Proxy........................................30, 72, 85
PTP...........................................81
QOS...........................................55, 95
Quality of Service.........................55, 95
Rapid Reconfiguration STP................65
Rapid STP...................................65
Real Time Innovations
Spreadsheet.................................34
Real-time characteristics..................31
Real-time classes...........................31
Redundancy.................................63, 95
Repeating hub...............................46
RJ45
Rockwell investigation....................35
Rockwell
Ethernet/IP..................................73
RSTP
Rapid STP...................................65
RTEthernet (protocol).....................67
| Safe Ethernet (protocol)          | 12, 67 |
| Sercos (protocol)                | 77    |
| Serial device servers            | 87    |
| SFTP                            | 25    |
| Shielded Foiled Twisted Pair     | 25    |
| Shielded Twisted Pair            | 25    |
| Simple Network Management Protocol | 94.    |
| See: SNMP                        |       |
| Simple NTP                       | 81    |
| Sniffer                          | See: Network analyser |
| SNMP                            | 13, 93, 94 |
| Socket interface                 | 88    |
| Software delays                  | 41    |
| Spanning Tree Protocol           | 31, 59, 63 |
| Speed                           |       |
| Comparison with CAN             | 40    |
| Speed comparison                 | 41    |
| Spurgeon, Charles (author)      | 95    |
| Spy                              | See: Network analyser |
| Sniffer                          |       |
| Service Request Transfer Protocol | 67    |
| Store and forward switches       | 49    |
| STP                              | 25, 31 |
| Disadvantages                   | 65    |
| Improvements                    | 65    |
| Shielded Twisted Pair            | 63    |
| Spanning Tree Protocol           | 63    |
| TurboRing variant                | 65    |
| Switch                           |       |
| Buffering strategy               | 50    |
| Cut-through                      | 49    |
| Firmware update                  | 60    |
| Flood control                    | 57    |
| Flow control                     | 56    |
| Full-duplex capability           | 49    |
| Influence of software            | 65    |
| Internal structure               | 48    |
| Layer 2.7 operation              | 54    |
| MTBF improvement                 | 61    |
| Non-blocking                     | 50    |
| Operation                        | 47    |
| Port mirroring                   | 56    |
| Port-trunking                    | 56    |
| ProfiNet internal               | 30    |
| Self learning capability         | 50    |
| Start-up time                    | 59    |
| Store-and-Forward                | 49    |
| Switch table use                 | 50    |
| Uplink port                      | 61    |
| Switching hub                    | 46    |
| TCP/IP                           | 35, 36, 73 |
| Developments                     | 37    |
| Disadvantages                    | 24    |
| TELNET (protocol)                | 92    |
| Thick Ethernet                   | 12    |
| Thin Ethernet                    | 12    |
| Tunnelling                       | 37    |
| UDP                              |       |
| Compared with TCP                | 37    |
| Unshielded Twisted Pair          | 25    |
| USB                              | 10, 87 |
| UTP                              | 25    |
| Video data transport             | 58    |
| VLAN                             | 21, 23 |
| Vogel Verlag (publisher)         | 96    |
| Webswitching                     | 55    |
| Wireless Ethernet                | 13    |
| Wiring topologies                | 28    |
| Zigbee                           | 13, 15 |