Doctoral Thesis

Real-Time Communication for Distributed Control Systems

September 2005

Alimujiang Yiming

Department of Computer Sciences

Kitami Institute of Technology

165 Koen-cho, Kitami, Hokkaido 090-8507, Japan

Contents

1. Introduction

1.1 Motivation

Networks for industrial communication are usually some kind of fieldbus. The common characteristics for these networks are higher reliability, lower data throughput, and a high price tag. Communication networks were introduced in the industrial digital control systems in the 1970's. The main motives for introducing communication networks were to the change from 4-20 mA analog point-to-point wiring to a digital bus wiring where many devices can be connected to one wire. Each device on the fieldbus must have a unique physical device tag and a corresponding network address. Another reason is to provide the ability to distribute some of the control and input/output (I/O) sub system functions from the control system to the fieldbus devices. This may reduce the number of rack mounted controllers and remote mounted I/O equipment needed for the system design.

In addition to this, applying communication networks also bring about reduced cost for cabling, modularization of systems, and flexibility in system setup *comparing to* the analog systems. Since then, several types of communication networks have been developed.

One of the commonly used communication networks developed by Fieldbus Foundation was Fieldbus Control System (FCS). The FCS is an all-digital, serial, two-way communications system that interconnects intelligent measurement and control devices such as sensors, actuators and controllers. At the base level in the hierarchy of plant networks, it serves as a Local Area Network (LAN) for devices used in process control and manufacturing automation applications and has a built-in capability to distribute the control application across the network.

FCS is not only an open, interoperable communication network which is based on the OSI seven-layer communications model, but also a Total Distributed Control Systems.

As the connection of the intelligent device, it unites each device to become a network node, and links the node with the fieldbus to become a network system, and then, makes the automation system to carry out the universal automation functions of basic controlling, compensation calculating, parameter modifying, alarming, displaying, monitoring, optimizing and managing.

However, the centralized design, special-purposed many protocols and high costs are the main limiting factor for these fieldbus control systems to meet with the strict time-limited hard real-time requirements.

On the other hand, current Ethernet standards promise network segmentation and full duplex that greatly enhance Ethernet's deterministic performance. Also, Ethernet provides more bandwidth --- transfer speeds of 100Mbit/s (Fast Ethernet) and more (Gbit Ethernet). These together can minimize or completely eliminate collisions on the network. Therefore fast Ethernet also seems to be a good choice for connecting industrial devices with real-time requirements.

Traditionally, Ethernet meets the IEEE 802.3 standard, in which the channel access is random since it is for a CSMA/CD LAN (which will be explained in the subsection 3.4.1 of chapter 3). This property leads to the unpredictable timing when sending and receiving data on the network. Therefore Ethernet is non-deterministic, which is not suitable for real-time communication applications.

The development of new equipment for network interconnection, i.e. Ethernet switches, has made a different approach possible. Data terminal equipment (DTE) connected to a switch communicating in full duplex does not have to use the CSMA/CD access control, since there are only two ports per network segment (switch port and device port), the forward and backward channel of the segment which can be used simultaneously. In this sense CSMA/CD can be no longer used. With switched Ethernet, collisions no longer occur in any of the network segments. The full-duplex operation theoretically doubles the bandwidth of the network. For example, in a 100 Mbps

full-duplex switched Ethernet, the bandwidth of the network can be reached up to 200 Mbps. In addition to this, over the years, Ethernet transmission rate and communication reliability have increased. This together with serious attempts to adapt fast Ethernet hardware to industrial environments, make it an interesting alternative for real-time communication.

The proposed work is to develop a protocol for industrial hard real-time communication over switched Ethernet network, without modifying any Ethernet hardware in the end nodes.

Our main research problem is to schedule and control the hard real-time traffic without changing the underlying protocols, while still supporting existing best-effort protocols for non or soft real-time traffic (e.g., web based best-effort traffic which is highly desirable to coexist with the real-time traffic), in order to avoid the nondeterministic behavior of Ethernet. Another aim is to make it possible to establish end-to-end communication between office and industrial production field.

The research is motivated by the strong interest of using cheap existing hardware and software whenever possible in industrial and embedded real-time control systems. The research efforts on hard real-time communication over switched Ethernet LAN technologies have so far been concentrated on multimedia and similar applications, while there is a large need of research attempts in the field of industrial and embedded communication systems. Industrial real-time Ethernet communication has a broad target market including application areas such as: (1.) industrial communication industry with products used in manufacturing industries where the fieldbus technologies are commonly used. (2.) Communication in embedded systems like robot signal processing and communication systems. (3.) Broadband access networks, and (4.) Traditional local area networks (LANs) where, e.g., the enhanced support for multi-media traffic is needed. Our focus is on industrial and embedded communication systems (first two application areas). For the other application areas, we see that the solution method

should be suitable, since they have very similar demands for real-time traffic support.

1.2 Research overview

Our research is aims to develop a protocol to support industrial hard real-time traffic of end-to-end communication, with a switched Ethernet based network concept.

The network is set up with nodes and switches, and real-time communication is handled by software (protocol) added between the Ethernet protocols and the TCP/IP suites. The proposed protocol manages hard real-time traffic to bypass the TCP/IP stacks. This makes considerably reduce the dwell time in the nodes, and increase the achievable data frame rate. After the bypassing, traffic schedule is performed according to dynamic-priority EDF algorithm.

The protocol does not need any modifications in the Ethernet hardware and coexists with TCP/IP suites, and then the LAN with the protocol can be connected to any existing Ethernet networks. It can be adopted in hard real-time applications fields such as industrial embedded systems, distributed industrial systems, parallel signal processing and robotics.

An Ethernet LAN using the switch was constructed and several experiments have been performed to evaluate the proposed protocol. Through comparing with the conventional hard real-time communication protocols, we proved that the proposed Ethernet protocol has better real-time performances, and meets the requirements of reliability for hard real-time systems without any modification of the original Ethernet hardware.

1.3 Main contributions

The objective of the work conducted in this thesis has been to develop an efficient and reliable hard real-time communication protocol over switched Ethernet network.

We made a LAN with a full-duplex switched Ethernet and end-nodes, by using desktop computer and several embedded Ethernet development boards, and real-time communication is handled by software (protocol) added between the Ethernet protocols and the TCP/IP suites.

The proposed protocol establishes a way (virtual link) between source nodes and destination nodes, by applying admission control based upon the requested QoS. However, in our work, a key strategy to realize hard real-time communication is "bypassing" of TCP/IP suites. The proposed protocol manages hard real-time traffic to bypass the TCP/IP stacks. This makes considerably reduce the dwell time in the nodes, and increase the achievable data frame rate by evasion of the non-deterministic behavior inherent in the TCP and IP stacks. This should be the main contribution of our work.

Besides, we provided a simple and practical way of design and implementation of a switched Ethernet protocol for the industrial hard real-time applications, by using cheap and existing hardware and/or software whenever possible. This requires that the network used should at least be based on existing hardware. Customized hardware would drive up the price of devices considerably and take up a lot of time to design and implement.

Only a thin layer (RT layer) is added between the Ethernet protocols and the TCP/IP suite in each end nodes and switch to support both bit rate and maximum latency guarantees for hard real-time traffic. The added RT layer in the nodes enables applications to reserve real-time channels on the network subject to a feasibility analysis. Real-time channel (RT channel) is established between source nodes and destination nodes according to the RT layer, not defines a way of encapsulating other layer 2 and layer 3 protocols that the other protocols does.

The proposed protocol does not need any modifications in the Ethernet hardware on the network interface cards, and coexists with TCP/IP suites. Which means that non-real-time nodes (nodes without RT layer added) can coexist in the network, and the

proposed protocol support for co-existing standard TCP/IP and UDP/IP Internet traffic without disturbing the real-time traffic, in order to support existing network systems. Therefore, the LAN with the proposed protocol can be connected to the existing Internet networks. It can be adopted in hard real-time applications such as industrial distributed control systems, embedded control systems, parallel signal processing and robotics.

1.4 Outline of the Thesis

The thesis consists of six chapters, which are briefly summarized below.

In Chapter 2, firstly described the traditional computer-based industrial control systems and the previous practical implementation by using the traditional control systems. And then, as the main part of the chapter, the general descriptions of the Fieldbus Control Systems are presented. Which mainly includes Foundation Fieldbus technology, its system management and system configurations. Also, connection of the Fieldbus control systems with high speed Ethernet has outlined. The practical applications of the Fieldbus control technology in intelligent building are described briefly. The limitations of the Fieldbus control systems are pointed out at the end of the chapter.

In Chapter 3, firstly the concept of real-time and real-time communications is introduced. Then the real-time Ethernet communication requirements and the reality analysis of implementing switched Ethernet in the real-time communications are presented. Last, some conventional protocols and advanced network technologies for supporting hard real-time communications are discussed, and described briefly of the proposed hard real-time communication protocol, and the differences with those existing protocols and network technologies.

As for the main part of the thesis, chapter 4 presents the implementation ways of the proposed hard real-time communication protocol, including the proposed network architecture, real-time traffic channel establishment, hard real-time communication

management and frame scheduling method.

The practical performance implementation by using the proposed hard real-time communication protocol has also given by the end of the chapter. And then, evaluated the proposed hard real-time communication protocol with several experimental results, by comparing with the other hard real-time supporting protocols.

Chapter 5 contains conclusions and suggestions for future research.

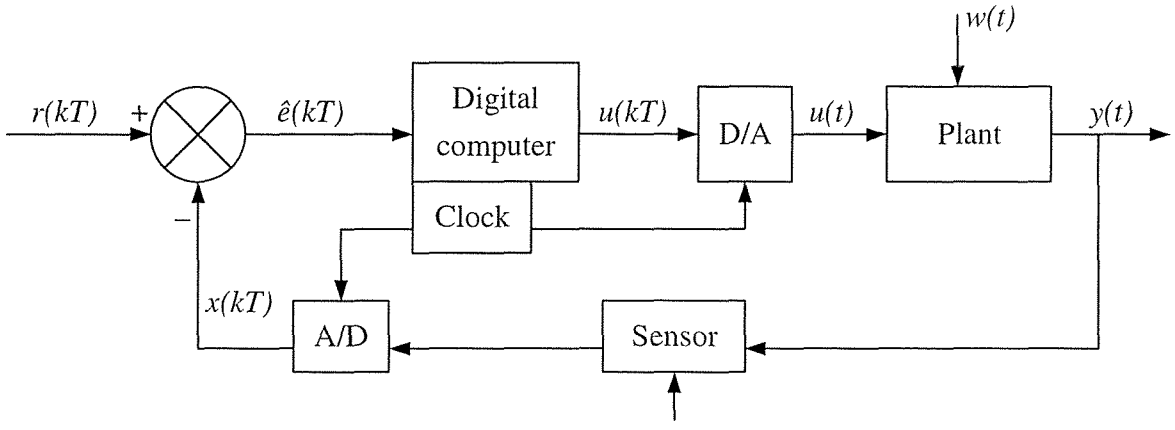
2. Descriptions of Fieldbus Control Technology

Before introducing the Fieldbus control technology, we briefly describe traditional computer based industrial control systems.

2.1 Traditional industrial control systems

Traditional industrial control systems are known as 4-20 mA systems, so named for the signal levels that control the devices. These devices take input measurements and send the information to a control unit for processing. The computer then performs the necessary calculations and tells devices what their outputs should be.

A typical topology of the computer-based industrial control systems is sketched schematically in Figure 2-1.



Notation:

r --- reference or command inputs
u --- control or actuator input signal
y --- controlled or output signal
x --- instrument or sensor output, usually an approximation to or estimate of y.
$\hat{e}$ --- $r - x$ = indicated error
e --- $r - x$ = system error
w --- disturbance input to the plant
v --- disturbance or noise in the sensor
A/D --- analog-to-digital converter
D/A --- digital-to-analog converter

Figure 2-1. Block diagram of a basic control system

The process to be controlled is called the plant and may be any of the physical processes whose satisfactory response requires control action. By "satisfactory response" we mean that the plant output, *y(t)*, is to be forced to follow or track the reference input, *r(t)*, despite the presence of disturbance inputs to the plant [*w(t)* in the Figure 2-1] and despite errors in the sensor [represented by *v(t)* in the Figure]. It is also essential that the tracking succeed even if the dynamics of the plant should change somewhat during the operation. The process of holding *y(t)* close to *r(t)*, including the case where $r = 0$, is referred to generally as the process of *regulation*. A system that has good regulation in the presence of disturbance signals is said to have good *disturbance rejection*. A system that has good regulation in the face of changes in the plant parameters is said to have low *sensitivity* to these parameters. A system that has both good disturbance rejection and low sensitivity we call *robust*.

For a practical implementation of the computer-based industrial control systems, we made the data communication between STD industrial control computer and the Great Wall 0520 personal computer [1], and illustrated how to program by both BASIC and ASSEMBLY language. In this work, we used STD-5000 computer as a field control instrument and the Great Wall 0520 personal computer as a monitor, by using asynchronous serial transfer method to perform the data communication between the two devices (see Figure 2-2). The main program is applied in BASIC and use CALL function to execute the other part used ASSEMBLY language.

Another implementation is introduced of the application of STD bus industrial control computer in AC speed adjusting system with variable frequency [2]. The control system adopts a Great Wall 0520 personal computer as a supervisory computer, a STD bus industrial control computer as a control computer, and a speed adjusting system as controlled object (see Figure 2-3).

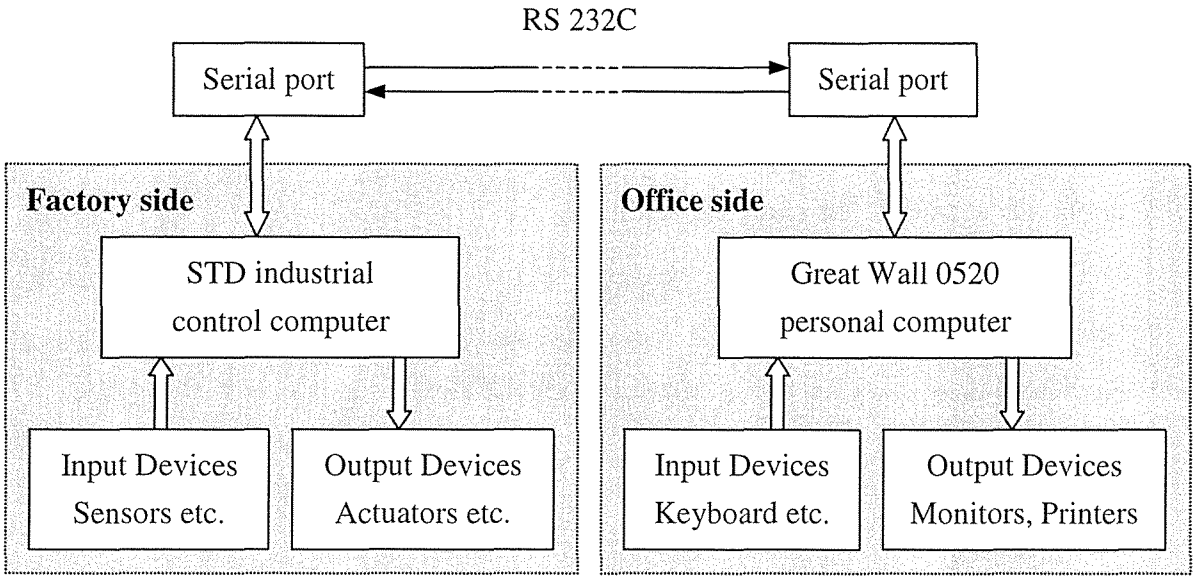


Figure 2-2. Data communication architecture

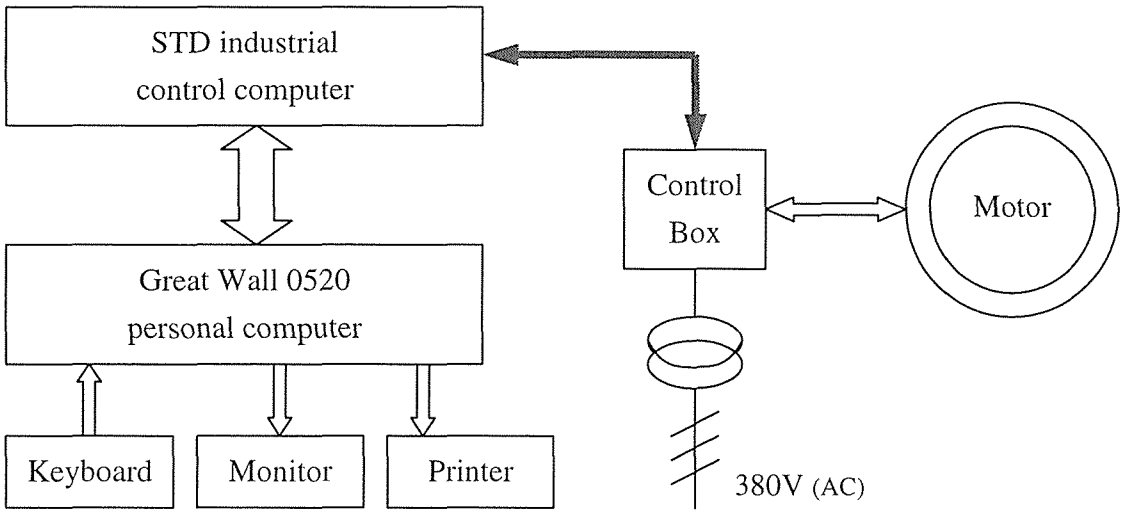


Figure 2-3. AC digital speed-controller using industrial control computer

In traditional systems, a computer or expensive controller unit provides control for the network of devices, and the devices would go into some pre-defined fail-safe mode, leaving the pumps, valves, and other equipment without any interactive control until communication with the computer resumes.

The main drawback of these systems is that they are proprietary and thus not interoperable. However, Fieldbus control system is an open standard that allows the field devices to run *both* the input/output and the control. It is the open, interoperable, interchangeable systems that we introduce the next section.

2.2 Fieldbus Control System

Fieldbus Control System (FCS) is an all-digital, serial, two-way communications system that interconnects measurement and control equipment such as sensors, actuators and controllers (see Figure 2-4). At the base level in the hierarchy of plant networks, it serves as a Local Area Network (LAN) for instruments used in process control and manufacturing automation applications and has a built-in capability to distribute the control application across the network [3].

FCS is not only an open communication network, but also a Total Distributed Control Systems. As a connection of the intelligent instruments, it unites them that is a network node and is linked up with the fieldbus, and becomes network systems, then makes up automation systems to carry out the universal automation functions of basic controlling, compensation calculating, parameter modifying, alarming, displaying, monitoring, optimizing and managing. This is a universal technique, mainly including intelligent sensor, control, computers, digital communications and networks [4].

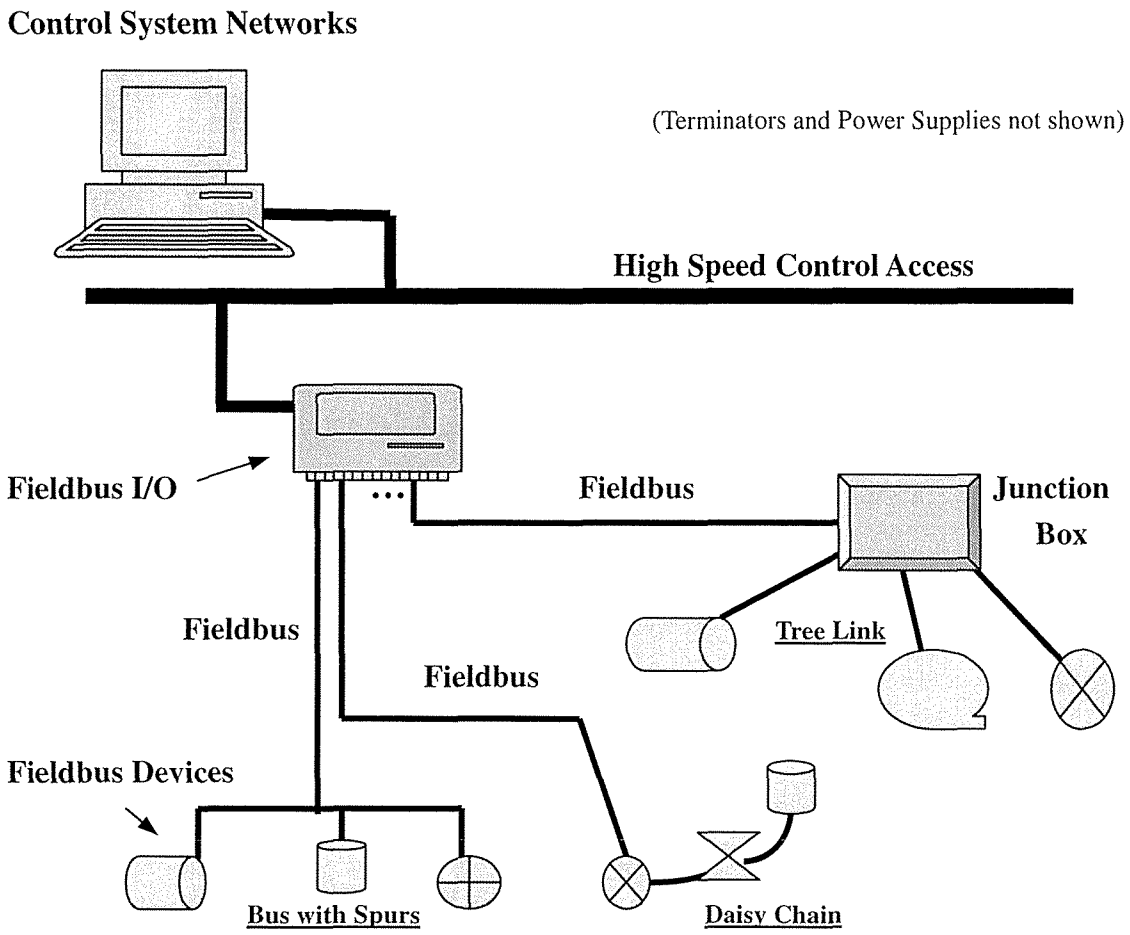


Figure 2-4. Fieldbus control system networks

2.2.1 Technical Characteristics of FCS

The main technical characteristics of FCS as follow:

- *Open characteristics of systems*

Which means it can connect with other instruments or other systems in anywhere around the world, if it is comply with the same standard. The communication protocol is consistently opened, for accomplishing the information exchange between different devices provided the different producer.

- ***Interoperability***

This refers to the ability to plug digital instruments onto a fieldbus network and have them communicate with each other and supervisory control systems. Users want to operate their entire plant from one operator interface and maintain all devices on the system with one maintenance application.

- ***Intellectualized field devices and functions of self-control***

It distributes the functions of sensor measurement, compensation calculating, engineering volumes process and control to field devices. The field devices can complete the basic functions of automatic control, and can diagnose the working state of devices at all times.

- ***System configurations are highly distributed***

Fieldbus has constituted a kind of new and total distributed control system structure, and radically changed the framework of old Concentrated and Distributed Control Systems, simplified the system structure and improved the system reliability.

- ***Adaptability to the field environment***

Fieldbus are specially designed for field environment, so it can support twisted pair, coaxial cable, optical fiber, radio frequency, infrared wire and electric wire, it has strong ability of interference resistance, and it can meet the demand of Intrinsically Save (I.S.) and flame-proof.

2.2.2 The benefits of FCS

By enabling digital interoperability among field devices and systems from a variety of companies, fieldbus offers users the flexibility to add new devices with the confidence that active control functions on the fieldbus will not be disrupted. Changes to improve plant operation can be made without waiting for a scheduled plant shutdown.

The other benefits of FCS include reduced wiring, multi-variables from a single field

device, enhanced field-level control, simpler integration and easier maintenance.

Ultimately, fieldbus will be the key to greater manufacturing flexibility and productivity, higher quality products, and improved regulatory compliance.

2.3 About Foundation Fieldbus

There are many kinds of Fieldbus Control Technique, such as Foundation Fieldbus (FF), Lonworks, Profibus, Control Area Network (CAN), Highway Addressable Remote Transducer (HART) and others. They all have their own technical characteristics and have made great contributions to the development of Fieldbus Control Technique.

Foundation Fieldbus (FF) Technique is organized and developed by Fieldbus Foundation. The Fieldbus Foundation is the leading organization dedicated to a single international, interoperable fieldbus standard.

Established in September 1994 by a merger of WorldFIP North America and the Interoperable Systems Project (ISP), the foundation is a not-for-profit corporation that consists of more than 120 of the world's leading suppliers and end users of process control and manufacturing automation products. Working together, these companies have provided unparalleled support for a worldwide fieldbus protocol, and have made major contributions to the IEC/ISA fieldbus standards development.

The main differences between the Fieldbus Foundation and other fieldbus initiatives is that, the foundation's technology --- Foundation Fieldbus is unique insomuch as it is designed to support mission-critical applications where the proper transfer and handling of data is essential. Unlike proprietary network protocols, Foundation Fieldbus is neither owned by any individual company, nor controlled by a single nation or regulatory body. Rather, it is an open, interoperable fieldbus that is based on the International Standards Organization's Open System Interconnect (OSI/ISO) seven-layer communications model [5-6].

2.4 Foundation Fieldbus Technology

Foundation Fieldbus Technology consists of three major parts as following:

The Physical Layer The Communication "Stack" The User Application

The Open Systems Interconnect (OSI) layered communication model is used to model these components. The Physical Layer is OSI layer 1. The Data Link Layer (DLL) is OSI layer 2. The Fieldbus Message Specification (FMS) is OSI layer 7. The Communication Stack is comprised of layers 2 and 7 in the OSI model (see Figure 2-5).

The Fieldbus does not use OSI layers 3, 4, 5 and 6. The Fieldbus Access Sublayer (FAS) maps the FMS onto the DLL.

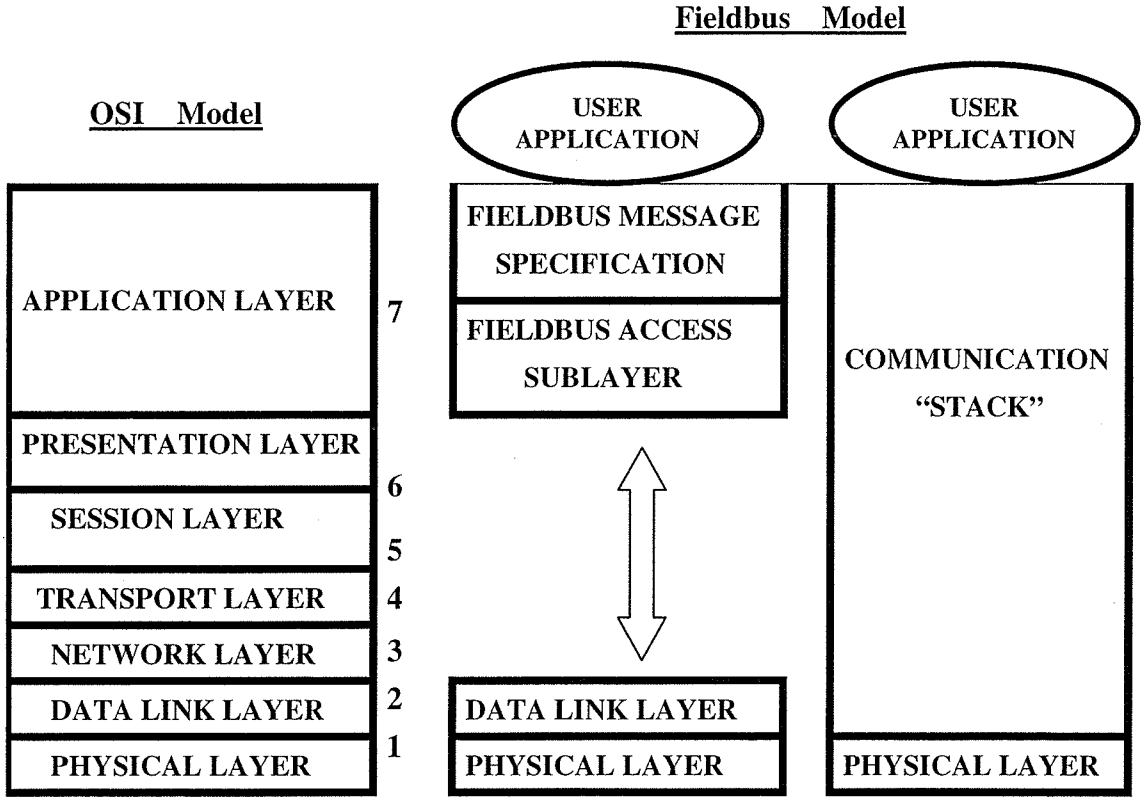


Figure2-5. Foundation Fieldbus communication model

The User Application is not defined by the OSI model. The Fieldbus Foundation has specified a User Application model. Each layer in the communication system is responsible for a portion of the message that is transmitted on the fieldbus.

2.4.1 Physical Layer (31.25 kbps)

The Physical Layer is defined by approved standards from the International Electrotechnical Commission (IEC) and the International Society for measurement and control (ISA). The Physical Layer receives messages from the communication stack and converts the messages into physical signals on the fieldbus transmission medium and vice-versa.

Fieldbus signals are encoded using Manchester Biphase-L technique. This signal is called "synchronous serial" because the clock information is embedded in the serial data stream. Data is combined with the clock signal to create the fieldbus signal. The receiver of the fieldbus signal interprets a positive transition in the middle of a bit time as a logical "0" and a negative transition as a logical "1".

The transmitting device delivers $\pm$10mA at 31.25 kbps into a 50 Ohm equivalent load to create a 1.0 volt peak-to-peak voltage modulated on top of the direct current (DC) supply voltage. The DC supply voltage can range from 9 to 32 volts. However, for I.S. applications, the allowed power supply voltage depends on the barrier rating.

31.25 kbps devices can be powered directly from the fieldbus and operate on wiring previously used for 4-20mA devices.

The 31.25 kbps fieldbus also supports I.S. fieldbuses with bus-powered devices. To accomplish this, an I.S. barrier is placed between the power supply in the safe area and the I.S. device in the hazardous area.

The length of the fieldbus is determined by the communication rate, cable type, wire size, bus power option and I.S. option.

Table 2-1 gives a summary of examples of options available in the Physical Layer

standard. The number of devices possible on a fieldbus link depends on factors such as the power consumption of each device, the type of cable used, use of repeaters, etc.

2.4.2 Communication Stack

The Communication stack mainly consists of three parts: the Data Link Layer (DLL), the Fieldbus Access Sublayer (FAS) and the Fieldbus Message Specification (FMS). The following will describe the layers in the Communication Stack.

Table 2-1

Characteristics	**Data Rate**		
	31.25 kbps	31.25 kbps	31.25 kbps
Type	Voltage	Voltage	Voltage
Topology	Bus/tree	Bus/tree	Bus/tree
Bus Power	None	DC	DC
Classification		Intrinsically Safe	
Number of Devices	2-32	2-6	2-12
Cable Length	1900 m	1900 m	1900 m
Spur Length	120 m	120 m	120 m

- ***The Data Link Layer (DLL)***

Layer 2, the Data Link Layer (DLL), controls transmission of messages onto the fieldbus. The DLL manages access to the fieldbus through a deterministic centralized bus scheduler called the Link Active Scheduler (LAS).

The DLL is a subset of the emerging IEC/ISA DLL standard.

Two types of devices are defined in the DLL specification:

Basic Device and **Link Master**

Link Master devices are capable of becoming the Link Active Scheduler (LAS). **Basic Devices** do not have the capability to become the LAS.

Scheduled Communication: The Link Active Scheduler (LAS) has a list of transmit times for all data buffers in all devices that need to be cyclically transmitted. When it is time for a device to send a buffer, the LAS issues a Compel Data (CD) message to the device. Upon receipt of the CD, the device broadcasts or "publishes" the data in the buffer to all devices on the fieldbus. Any device configured to receive the data is called a "subscriber". Scheduled data transfers are typically used for the regular, cyclic transfer of control loop data between devices on the fieldbus.

Unscheduled Communication: All devices on the fieldbus are given a chance to send "unscheduled " messages between transmissions of scheduled messages. The LAS grants permission to a device to use the fieldbus by issuing a pass token (PT) message to the device. When the device receives the PT, it is allowed to send messages until it has finished or until the "maximum token hold time" has expired, whichever is the shorter time.

- ***The Fieldbus Access Sublayer (FAS)***

The FAS uses the scheduled and unscheduled features of the Data Link Layer to provide a service for the Fieldbus Message Specification (FMS). The types of FAS services are described by Virtual Communication Relationships (VCR).

The VCR is like the speed dial feature on your memory telephone. There are many digits to dial for an international call such as international access code, country code, city code and finally the specific telephone number. This information only needs to be entered once and then a "speed dial number" is assigned.

After set up, only the speed dial number needs to be entered for the dialing to occur. Likewise, after configuration, only the VCR number is needed to communicate with

another fieldbus device. Just as there are different types of telephone calls such as person-to-person, collect, or conference calls, there are different types of VCRs such as: **Client/Server VCR Type, Report Distribution VCR Type, Publisher/Subscriber VCR Type,** etc. Summary of VCR Types is shown in Table 2-2.

Table 2-2

FIELDBUS ACCESS SUBLAYER SERVICES		
Client/Server VCR Type	Report Distribution VCR Type	Publisher/Subscriber VCR Type
Used for Operation Messages Set point changes Mode changes Tuning changes Upload/Download Alarm Management Access display views Remote diagnostics	Used for Event Notification and Trend Reports Send process alarms to operator consoles. Send trend reports to data historians.	Used for Publishing Data Send transmitter PV to PID control block and operator console.

- ***Fieldbus message Specification (FMS)***

Fieldbus Message Specification (FMS) services allow user applications to send messages to each other across the fieldbus using a standard set of message formats.

FMS describes the communication services, message formats, and protocol behavior needed to build messages for the User Application. Data that is communicated over the fieldbus is described by an "object description". Object descriptions are collected

together in a structure called an "object dictionary" (OD). The object description is identified by its "index" in the OD. Index 0, called the object dictionary header, provides a description of the dictionary itself, and defines the first index for the object descriptions of the User Application. The User Application object descriptions can start any index above 255. Index 255 and below define standard data types such as Boolean, integer, float, bit string and data structures that are used to build all other object descriptions.

2.4.3 User Application—Blocks

The Fieldbus Foundation has defined a standard User Application based on "Blocks". Blocks are representations of different types of application functions.

- ***Resource Block***

The Resource Block (RB) Describes characteristics of the fieldbus device such as the device name, manufacturer, and the serial number. There is only one resource block in a device.

- ***Function Block***

Function Blocks (FB) provides the control system behavior. The input and output parameters of Function Blocks can be linked over the fieldbus. The execution of each Function Block is precisely scheduled. There can be many function blocks in a single User Application. The Fieldbus Foundation has defined sets of standard Function Blocks. There are defined ten standard Function Blocks for basic control as follow:

Function Block Name	Symbol
Analog Input	**AI**
Analog Output	**AO**
Bias	**B**
Control Selector	**CS**

Discrete Input	**DI**
Discrete Output	**DO**
Manual Loader	**ML**
Proportional/Derivative	**PD**
Proportional/Integral/Derivative	**PID**
Ratio	**RA**

Function blocks can be built into fieldbus devices as needed to achieve the desired device functionality. For example, a simple temperature transmitter may contain an AI function block. A control valve might contain a PID function block as well as the expected AO block. Thus a complete control loop can be built using only a simple transmitter and a control valve.

- ***Transducer Blocks***

Transducer Blocks decouple Function Blocks from the local input/output functions required to read sensors and command output hardware. They contain information such as calibration date and sensor type. There is usually one transducer block for each input or output function block.

- ***Fieldbus Device Definition***

The function of a fieldbus device is determined by the arrangement and interconnection of blocks. The device functions are made visible to the fieldbus communication system through the User Application Virtual Field Device (VFD).

The header of the User Application object dictionary points to a Directory which is always the first entry in the function block application. The Directory provides the starting indexes of all of the other entries used in the Function Block application.

The VFD object descriptions and their associated data are accessed remotely over the fieldbus network using Virtual Communication Relationships (VCRs).

2.5 System Management

2.5.1 System Management

Function Blocks must execute at precisely defined intervals and in the proper sequence for correct control system operation. System management synchronizes execution of the Function Blocks and the communication of function block parameters on the fieldbus. System management also handles other important system features such as publication of the time of day to all devices, including automatic switchover to a redundant time publisher, automatic assignment of device addresses, and searching for parameter names or "tags" on the fieldbus. All of the configuration information needed by System Management such as the Function Block schedule is described by object descriptions in the Network and System Management Virtual Field Device (VFD) in each device. This VFD provides access to the System Management Information Base (SMIB), and also to the Network Management Information Base (NMIB).

2.5.2 Device Descriptions

A critical characteristic required of fieldbus devices is interoperability. To achieve interoperability, Device Description (DD) technology is used in addition to standard function block parameter and behavior definitions. The DD provides an extended description of each object in the Virtual Field Device (VFD).

The DD provides information needed for a control system or host to understand the meaning of the data in the VFD including the human interface for functions such as calibration and diagnostics. Thus the DD can be thought of as a "driver" for the device.

The DDs are similar to the drivers that your personal computer (PC) uses to operate different printers and other devices that are connected to the PC. Any control system or host can operate with the device if it has the device's DD.

- ***Device Description Tokennizer***

The DD is written in a standardized programming language know as Device Description Language (DDL). A PC-based tool called the "Tokenizer" converts DD source input files into DD output files by replacing key words and standard strings in the source file with fixed "tokens".

The Fieldbus Foundation provides DDs for all standard Function Blocks and Transducer Blocks. Device suppliers will typically prepare an "incremental" DD which references the Standard DDs. Suppliers may also add supplier specific features such as calibration and diagnostic procedures to their devices. These features can also be described in the incremental DD.

The Fieldbus Foundation makes the Standard DDs available on a CD-ROM. The user can obtain the incremental DD from the device supplier or from the Fieldbus Foundation if the supplier has registered their incremental DD with the Fieldbus Foundation. The incremental DDs can also be read directly from the device over the fieldbus, if the device supports the upload services and contains a Virtual Field Device (VFD) for the DD.

- ***Device Description Services (DDS)***

On the host side, library functions called Device Description Services (DDS) are used to read the device descriptions. Note that DDS reads descriptions, not operational values. The operational values are read from the fieldbus device over the fieldbus using FMS communication services.

New devices are added to the fieldbus by simply connecting the device to the fieldbus wire and providing the control system or host with the standard and incremental DD for the new device.

DDS technology allows operation of devices from different suppliers on the same fieldbus with only one version of the host human interface program.

- ***Interoperability***

Each manufacturer will provide an interoperability test report for each device. The test report identifies the Universal, Function Block, Transducer Block, and Manufacturer Specific Parameters in the device. An identifier called the Manufacturer's Identification is used to correlate the device type and revision with its Device Description and DD revision.

Any host using the Device Description Services (DDS) interpreter will be able to interoperate with all parameters that have been defined in the device by reading the device's DD.

2.6 System Configuration

Fieldbus system configuration consists of two phases:
System Design and Device Configuration

2.6.1 System Design

The system design for fieldbus-based systems is very similar to the Distributed Control Systems (DCS) design with the following differences.

The first difference is in the physical wiring due to the change from 4-20 mA analog point-to-point wiring to a digital bus wiring where many devices can be connected to one wire. Each device on the fieldbus must have a unique physical device tag and a corresponding network address.

The second difference is the ability to distribute some of the control and input/output (I/O) sub system functions from the control system to the fieldbus devices. This may reduce the number of rack mounted controllers and remote mounted I/O equipment needed for the system design

2.6.2 Device Configurations

After the system design is completed and the instruments have been selected, the device configuration is performed by connecting Function Block inputs and outputs together in each device as required by the control strategy.

After all of the function block connections and other configuration items such as device names, loop tags, and loop execution rate have been entered, the configuration device generates information for each fieldbus device.

A stand-alone loop can be configured if there is a field device that is a Link Master. This will allow continued operation of the loop without the configuration device or a central console.

The system becomes operational after the field-bus devices have received their configurations.

2.7 Connection with High Speed Ethernet

A Linking Device is used to interconnect 31.25 kbps fieldbuses and make them accessible to a High Speed Ethernet (HSE) backbone running at 100 Mbps or 1 Gbps. See figure 2-6 below.

The **I/O Subsystem Interface** shown in the figure allows other networks such as DeviceNet and Profibus to be mapped into standard Foundation fieldbus function blocks. The I/O Subsystem interface can be connected to the 31.25 kbps fieldbus or HSE.

Since all of the 31.25 kbps Foundation fieldbus messages are communicated on the HSE using standard Ethernet protocols (e.g. TCP/IP, SNTP, SNMP, etc.), commercial off-the-shelf HSE equipment such as Switches and Routers are used to create larger networks. Of course all or part of the HSE network can be made redundant to achieve the level fault tolerance needed by the application.

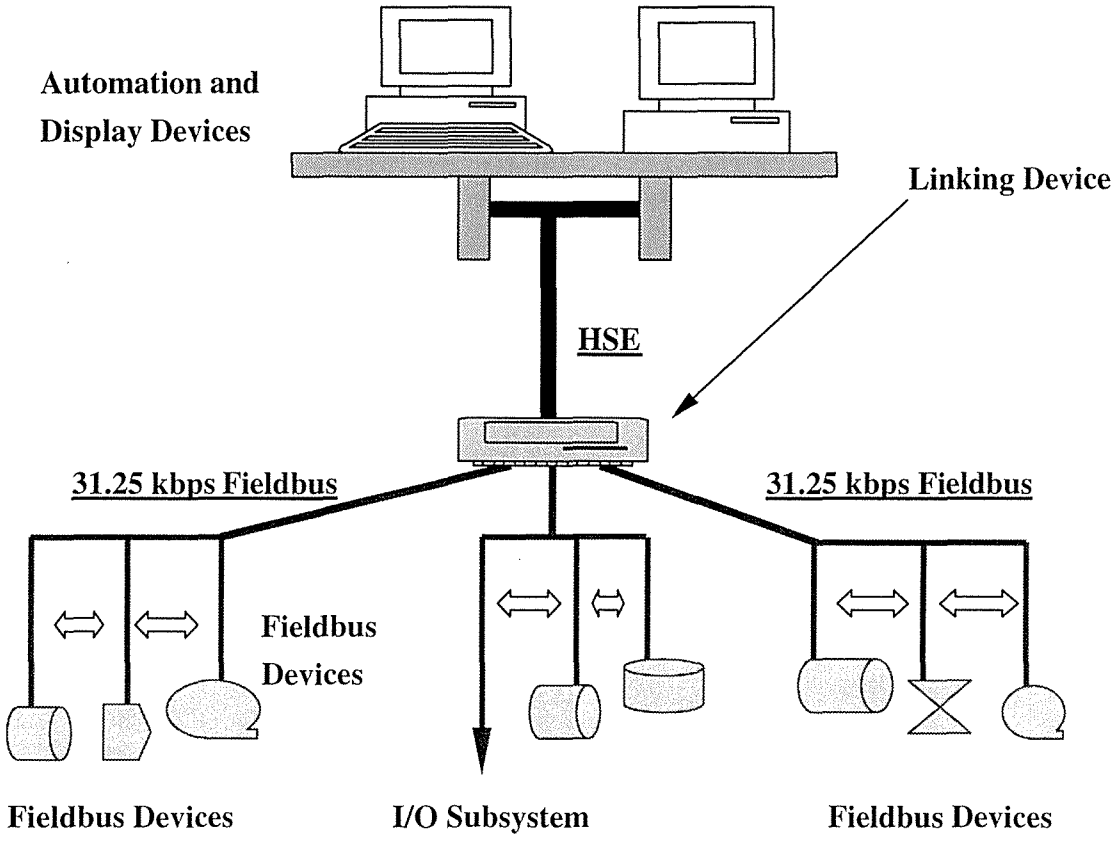


Figure 2-6. Fieldbus control system connection

2.8 Applications of Fieldbus Control Technology in Intelligent Building

We used LonWorks fieldbus control systems for practical application in the intelligent buildings in China [7]. Because LonWorks is originally designed for industrial control systems, the transmission speed of the network was very limited.

On one hand, the intelligent buildings community networks need to communicate very large amount of information; on the other hand, LonWorks network does not support such amount of data transmission. Therefore, the main problem that we need to solve is to reduce as much the network information as possible, solidifies and save the unchanging information to every end-node, such as Chinese character library, main

menu, speech sound, alarm etc., and create code for each of them. In order to reduce the network load, the network only transmits the code of each unchanging information. Therefore, the main design part is end-nodes. Hardware design including monitor and keyboard drive circuit, telephone link circuit, word sound circuit, ultra red checking and smoke, gas, water checking circuit, power circuit etc., software including alarming, medical treatment, menu selection, message leaving etc.

As shown in Figure 2-7, for the system configuration we used three-layer architecture: the first layer is management layer that the central PC linked with LAN and share the resource with other PC on the upper direction. On the lower direction, it linked with LonWorks network via PC adapter node. The second layer is network layer linked with central PC in first layer and its own end-nodes such as network gates; and the third layer is end-nodes which made up of many single-chip computers, to implement the functions of control, monitor, display, communication etc. Communication media is used twisted-pair wiring, and the communication speed of the lower layer (third layer) is 9600 bit/s, by using RS-485 standard bus.

There are three kinds of nodes on the LonWorks network. The first one is PC adapter node which is used to manage the communication between the PC and the LonWorks network, and send some network management information to keep the operation of the network; the second kind of node is network gate which transmits the information of user nodes to the network via RS-485 bus, and the network also replies the user nodes by using network gate and the RS-485. The third kind of node is GPS timer node which periodically sends time information to the network, in order to adjust the time of each node.

The main advantage of using Three-layer architecture is, the top layer and the second layer PC can interact each other and improve the ability of fault tolerance. For example, when the second layer PC has broken, the first layer PC sends out alarm signal, and temporarily performs the work of second layer PC. In the similar way, the second layer

PC also can act as the first layer PC, and share the resource with other PC over LAN. We implemented the system successfully in the intelligent buildings in Peking, China.

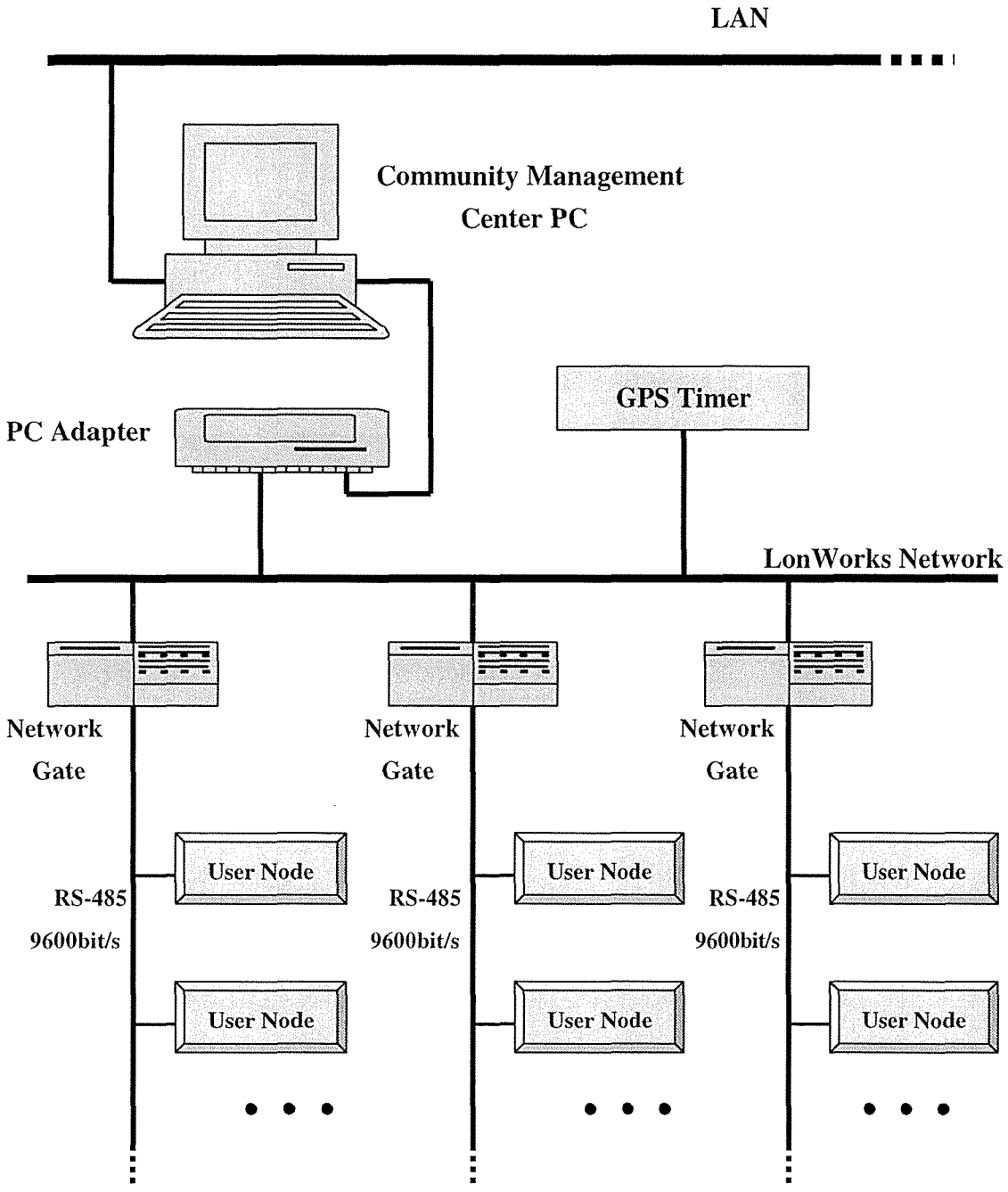


Figure 2-7. Intelligent buildings network configuration

2.9 Limitations of the Fieldbus control systems

As mentioned above, Fieldbus control systems were introduced in the industrial digital control systems to the change analog point-to-point wiring to a digital bus wiring where many devices can be connected to one wire. This brings about reduced cost for cabling, modularization of systems, and flexibility in system setup ***comparing*** *to* the analog systems.

However, fieldbus control systems is centrally organized automation designs which typically consist of a number of different networks, based on a number of protocols. Information is spread around in the organization; from sensor and actuator data at the field level, through production and support, and to logistics and orders at a more administrative level. The many different protocols that are used make it a difficult task to access data freely throughout the organization, especially across organizational "layers". Rather, information is structured hierarchically in the organization.

Moreover, fieldbus control system is a bus-based communication system which do not scale well. As more nodes are added to the network, collisions will occur more often and network performance will get worse. Minimizing collisions on the network is critical in keeping fieldbus control system performing well. To solve this issue, Ethernet has employed some methods to minimize collisions on the network. One of the simplest approaches is to just add bandwidth. The faster the transmit speed the quicker the network is free to transmit. However, most of the established, existing field devices should be changed in order to meet with the transmit speed of the network. Another way is to apply two separate networks for redundancy of the system. Both of the methods will also bring to cost problem and make the system more complex.

Therefore, the centralized design, special-purposed lots of protocols and high costs are the main limiting factor for these fieldbus control systems to meet with the strict time-limited hard real-time requirements.

On the other hand, current Ethernet standards promise network segmentation and full

duplex that greatly enhance Ethernet's deterministic performance. Also, Ethernet provides more bandwidth --- transfer speeds of 100Mbit/s (Fast Ethernet) and more (Gbit Ethernet). These together can minimize or completely eliminate collisions on the network. Therefore fast Ethernet also seems to be a good choice for connecting industrial devices with real-time requirements.

Next chapter will give the discussion about the research on hard real-time switched Ethernet communication.

3. Research on hard real-time switched Ethernet communication

3.1 Real-Time Systems

Real-time systems are computer systems, which must produce a result within a specified time. The overall correctness of the result is therefore dependent on the logical correctness of the result and on the time at which the result is produced. A logically correct result that comes too late is a wrong result [8].

Real-time systems are always surrounded by an environment, whose dynamic determines the needed behavior concerning time limitations.

An example of real-time systems is a robot used to pick up an object from a conveyor belt. The object is moving, and the robot must pick up the object within a given time slot. If the robot operates too slowly, it will miss the object even though it moves to the right place. On the other hand, if the robot operates too quickly, the object will not be there yet, and the robot may block it.

A real-time system is usually required to satisfy certain timing constrains [9-11]. The most used timing constraint is *deadline* i.e. the time point before which the expected result must be computed and delivered. In order to grade the importance of deadline, real-time tasks are often classified as being critical (hard), essential (soft) or non-essential (non real-time) as shown in Figure 3-1, where $y(t)$ is a cost measure associated with a task as a function of its completion time.

If a critical task misses its deadline, the consequences can be catastrophic and in most cases the system activity is terminated. Therefore, when critical tasks are present in the system, resources are often kept in reserve since the analysis of the required service for critical tasks is made on worst-case values rather than average behavior. Essential tasks will not cause a catastrophe or a system halt if they miss their deadlines, but will lead to a system malfunction for a short or long period of time. Most real-time tasks fall into this category. Finally, non-essential tasks often have no deadline at all (non-real-time

tasks) or, if they do have deadlines, nothing critical or essential will happen if these are missed. Maintenance tasks are typical examples of non-essential tasks.

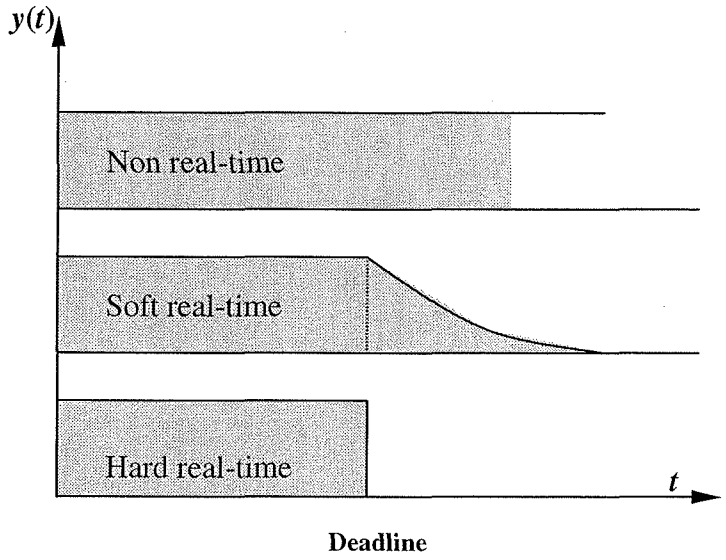


Figure 3-1. Illustration of different types of real-time systems

The classification into critical (hard), essential (soft) and non-essential (non real-time) tasks is an attempt made by the application or the user of the system to convey the importance of different tasks in a system and their deadlines [12]. The reason for doing this is that the available resources in the system often are limited and hence we need to use them as best as we can.

This is why scheduling becomes important [13-16]. Consider a processing unit, which is a limited resource that typically has several tasks of different importance to run. We then need to find a suitable procedure for determining the order in which these tasks should be executed. If all the tasks in the system are scheduled such that they all meet their respective deadlines, the corresponding schedule is called feasible. A scheduling

algorithm is said to be optimal if it can always find a feasible schedule whenever any other scheduling algorithm is able to do so [17-19].

There are a number of scheduling algorithms available for real-time scheduling [20-24] which we will illustrate in section 4.2, including *static* and *dynamic* scheduling algorithms.

3.2 Real-Time Communications

Communication is an important part of most real-time systems and can take place both between systems and within systems. For example, communication is needed in and between most industrial embedded and distributed systems so that sensors can report their readings and actuators be given directions. In multiprocessor systems the different processors need to communicate with each other.

The characteristics of real-time communication differ from the characteristics of non real- time communication. For example, the traditional measure of throughput is of less importance. Instead we are interested in a message loss rate or a probability of a message arriving before its deadline. A lost message will result in an infinite delay. Traditional critical hard real-time communication systems often require an upper bound on the maximum end-to-end message delay [25]. The real-time literature discusses two types of traffic in this context; guaranteed traffic and statistical traffic. When the traffic stream is guaranteed, it means that every frame in the stream is guaranteed to arrive before its deadline. Statistical traffic, on the other hand, refers to that no more than a certain percentage of the frames in the stream may miss their respective deadline [26].

Alternatively, some real-time literature classifies guarantees as being deterministic or probabilistic [27]. If the offered service is deterministic it is said to be predictable and suitable for hard real-time systems, since normally both a "guaranteed" minimum throughput and a bounded end-to-end delay is offered. A probabilistic guarantee is said to "only guarantee" to meet a specified QoS with a certain *probability*. This probability

may however be equal to one and hence result in performance that is comparable to a deterministic system [28].

Probabilistic guarantees are often connected to average behavior, whereas deterministic guarantees are made on a worst-case analysis and hence having a probabilistic guarantee is said to yield a higher utilization of the system resources.

The communication medium can be quite different depending on the application. It could be, e.g., optical wire line channels, copper wire local loops, a ring or a bus, wireless, or the Internet which is in a sense a mixture of all of the above. A local area network (LAN) is a relatively small network that shares a common medium that usually employs the same medium access control (MAC) method. The role of the MAC protocol is to ensure that all nodes connected to the LAN get access to the medium. The specific MAC method used generally depends on the medium in question, e.g., a ring networks may use Token ring, the Ethernet uses carrier sense multiple access with collision detection (CSMA/CD), and the GSM network uses time division multiple access (TDMA) [29].

Scheduling of tasks to different processors has many similarities with accessing the shared communication medium. The communication medium is a limited shared resource and the communication itself can be seen as a task that has a release time and a deadline. Consequently, processor scheduling algorithms can often be used to do communications scheduling as well. There are however a few differences: It is difficult to completely centralize the scheduling of communication tasks. Further, a large diverse network makes the scheduling problem significantly more complex. Specifically, if there are more than one MAC technique in use in the network and when routing is necessary.

Tasks are the real-time system's building blocks [30]. Each real-time task has certain temporal quantities (see Figure 3-2) associated with them:

The ***Release Time*** of a job is when the job becomes available to the system. The

Execution Time is the time it takes for a job to be completely processed. The ***Response Time*** is the interval between the release time and the completion of the execution. The ***Ready Time*** is the earliest time the job can start executing (always greater or equal to the Release Time). The ***Deadline*** is the time by which execution must be finished. If execution is not complete by the deadline, the job is late. A job's deadline can be either hard or soft, indicating the job's temporal dependence. As mentioned earlier, a missed hard deadline can have serious consequences for correct system operation. All real time systems have a certain level of jitter. ***Jitter*** is a variance on the actual timing of the above times. In a real-time system, jitter should be measurable within a +/– interval so that the system performance can still be guaranteed.

1. Release Time,
2. Ready Time,
3. Execution Time,
4. Response Time,
5. Deadline.

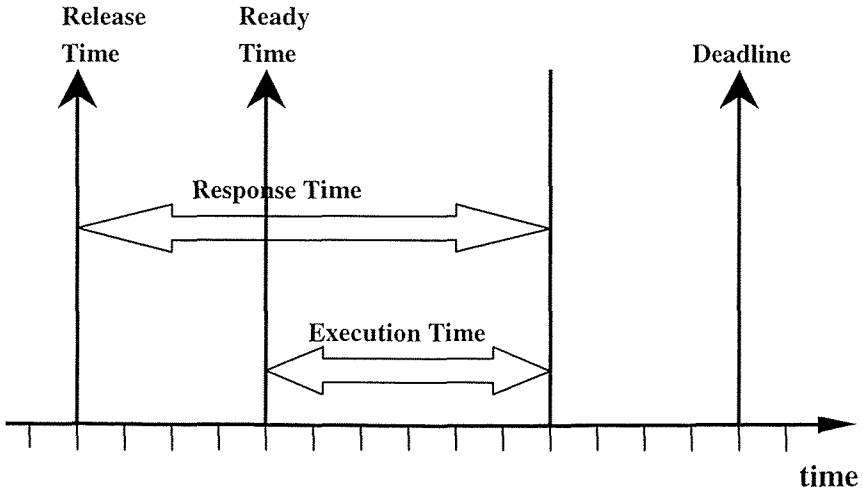


Figure 3-2. A real-time task

To develop a real-time distributed system, where computers are interconnected, it is vital to employ a network that can provide communication between the various distributed computers in a reliable and timely fashion. Distributed processors running real-time applications must be able to inter-communicate via a real-time protocol,

otherwise the temporal quality of their work is lost. Real-time communication networks are like any real-time system [31-32]. They can be critical, essential or non-essential, depending on the system requirements and their 'jobs' include message transmission, propagation, and reception. There are a number of real-time control networks available and employed in industry, but none have the popularity or bandwidth capabilities of Ethernet. In the following section, we will discuss the demand for a real-time Ethernet application.

3.3 Real-Time Communications with Ethernet Network

At the beginning of this section, we will give a short introduction for the Ethernet technologies, including the background of the Ethernet, a brief history, network elements, the Ethernet network topologies and structures, and the Ethernet data frame format. Then, we will focus on the real-time communication demand for Ethernet applications.

3.3.1 Ethernet Technologies

Background

The term *Ethernet* refers to the family of local-area network (LAN) products covered by the IEEE 802.3 standard that defines what is commonly known as the CSMA/CD protocol. Three data rates are currently defined for operation over optical fiber and twisted-pair cables:

- 10 Mbps—10Base-T Ethernet
- 100 Mbps—Fast Ethernet
- 1000 Mbps and more—Gigabit Ethernet

Ethernet has survived as the major LAN technology (it is currently used for approximately 85 percent of the world's LAN-connected PCs and workstations) because

its protocol has the following characteristics:

- Is easy to understand, implement, manage, and maintain
- Allows low-cost network implementations
- Provides extensive topological flexibility for network installation
- Guarantees successful interconnection and operation of standards-compliant products, regardless of manufacturer

Ethernet—A Brief History

The original Ethernet was developed as an experimental coaxial cable network in the 1970s by Xerox Corporation to operate with a data rate of 3 Mbps using a carrier sense multiple access collision detect (CSMA/CD) protocol for LANs with sporadic but occasionally heavy traffic requirements. Success with that project attracted early attention and led to the 1980 joint development of the 10-Mbps Ethernet Version 1.0 specification by the three-company consortium: Digital Equipment Corporation, Intel Corporation, and Xerox Corporation.

The original IEEE 802.3 standard was based on (and was very similar to) the Ethernet Version 1.0 specification. The draft standard was approved by the 802.3 working group in 1983 and was subsequently published as an official standard in 1985 (ANSI/IEEE Std. 802.3-1985). Since then, a number of supplements to the standard have been defined to take advantage of improvements in the technologies and to support additional network media and higher data rate capabilities, plus several new optional network access control features.

Ethernet Network Elements

Ethernet LANs consist of network nodes and interconnecting media. The network nodes fall into two major classes:

• Data terminal equipment (DTE)—Devices that are either the source or the destination of data frames. DTEs are typically devices such as PCs, workstations, file servers, or print servers that, as a group, are all often referred to as end-nodes.

• Data communication equipment (DCE)—Intermediate network devices that receive and forward frames across the network. DCEs may be either standalone devices such as repeaters, network switches, and routers, or communications interface units such as interface cards and modems.

The current Ethernet media options include two general types of copper cable: unshielded twisted-pair (UTP) and shielded twisted-pair (STP), plus several types of optical fiber cable.

Ethernet Network Topologies and Structures

LANs take on many topological configurations. However, regardless of their size or complexity, all will be a combination of only three basic interconnection structures or network building blocks.

The simplest structure is the point-to-point interconnection, shown in Figure 3-3. Only two network units are involved, and the connection may be DTE-to-DTE, DTE-to-DCE, or DCE-to-DCE. The cable in point-to-point interconnections is known as a network link. The maximum allowable length of the link depends on the type of cable and the transmission method that is used.

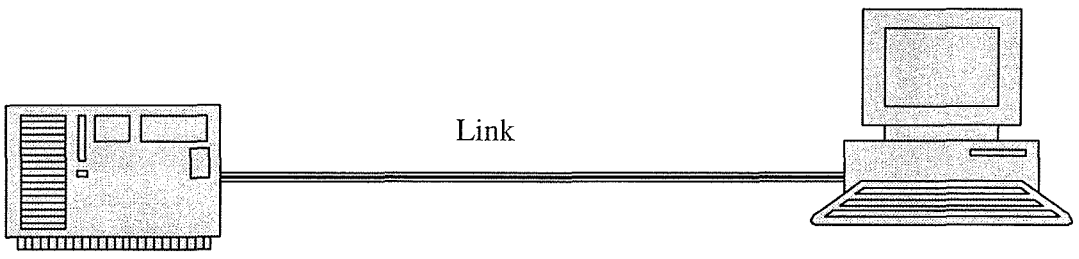


Figure 3-3. Example for Point-to-Point Interconnection

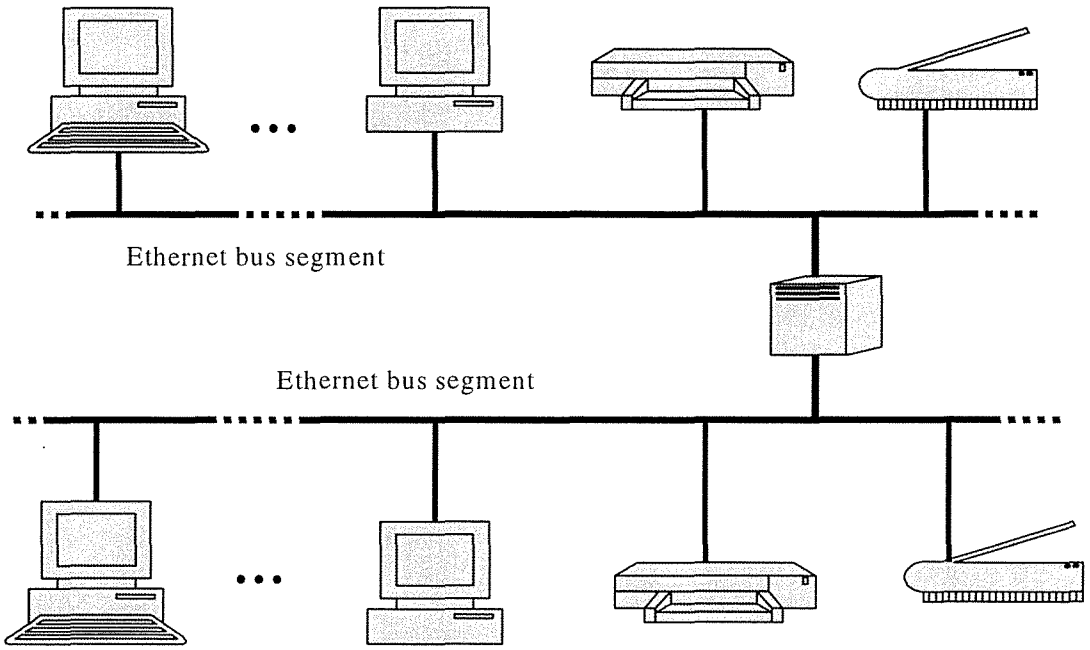


Figure 3-4. Example for Coaxial Bus Topology

The original Ethernet networks were implemented with a coaxial bus structure, as shown in Figure 3-4. Segment lengths were limited to 500 meters, and up to 100 stations could be connected to a single segment. Individual segments could be interconnected with repeaters, as long as multiple paths did not exist between any two stations on the network and the number of DTEs did not exceed 1024. The total path distance between the most-distant pair of stations was also not allowed to exceed a maximum prescribed value.

Although new networks are no longer connected in a bus configuration, some older bus-connected networks do still exist and are still useful.

Since the early 1990s, the network configuration of choice has been the star-connected topology, shown in Figure 3-5. The central network unit is either a multi-port repeater (also known as a hub) or a network switch. All connections in a star network are

point-to-point links implemented with either twisted-pair or optical fiber cable.

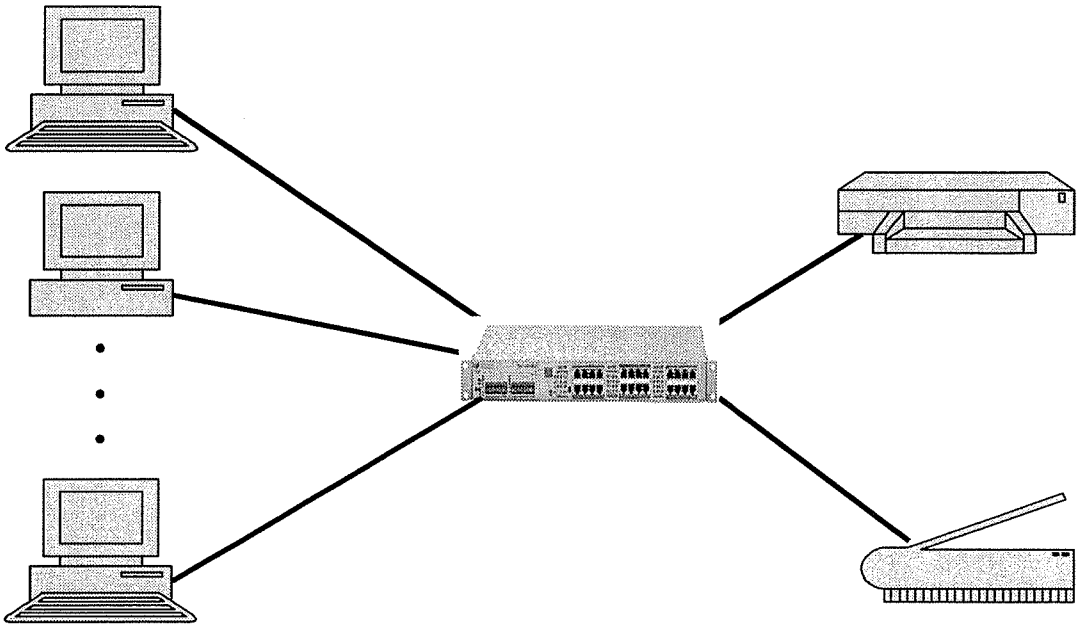


Figure 3-5. Example for Star-Connected Topology

The Basic IEEE 802.3 Ethernet Data Frame Format

The IEEE 802.3 standard defines a basic data frame format that is required for all MAC implementations, plus several additional optional formats that are used to extend the protocol's basic capability. The basic data frame format contains the eight fields shown in Figure 3-6.

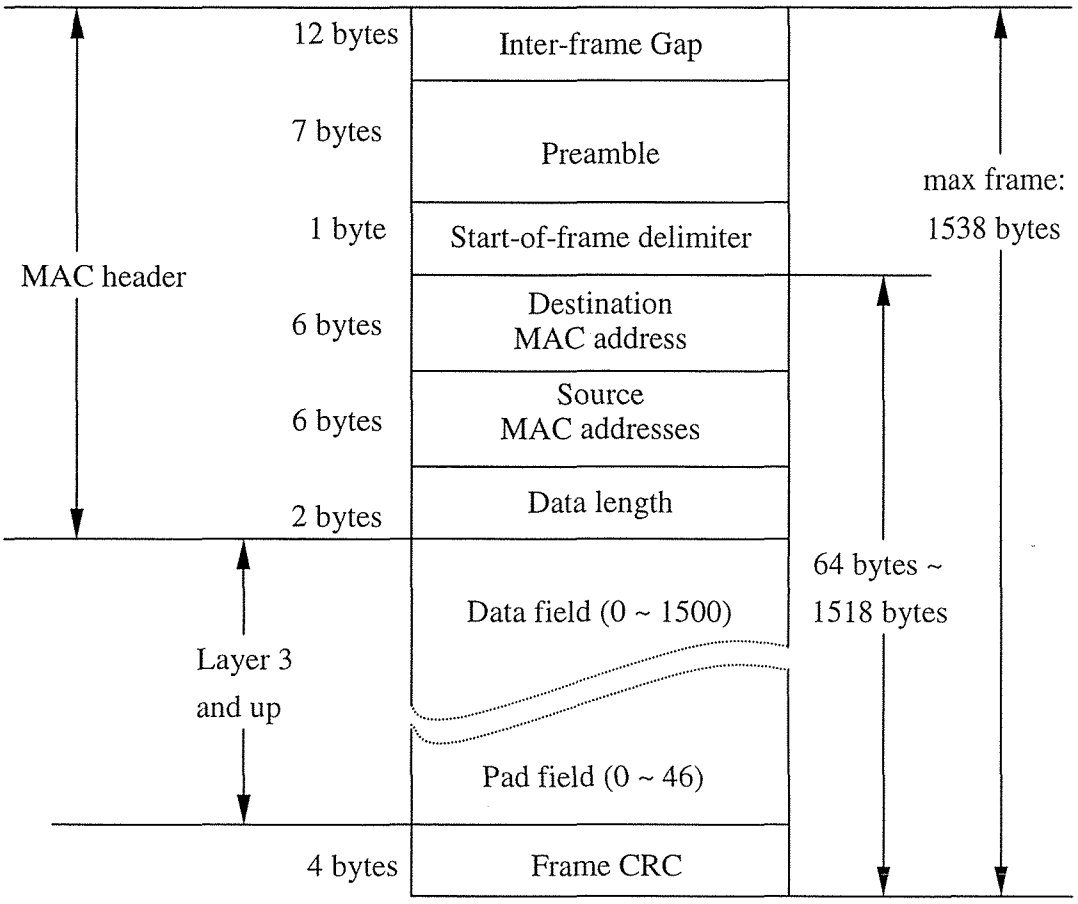


Figure 3-6. The Basic IEEE 802.3 Ethernet Data Frame Format

• **The Inter-frame Gap**—Defined as 96 bit times (12 bytes, 0.96 μ s at 100 Mbps), is an alternating pattern of ones and zeros. The inter-frame gap is the interval between successive Ethernet frames for the MAC. Depending on traffic conditions, the measurement reference for the inter-frame gap changes. If a frame is successfully transmitted without a collision, the inter-frame gap measurement starts from the deassertion of the Transmit Enable (TXEN) signal. However, if a frame suffers a collision, the inter-frame gap measurement starts from the deassertion of the Carrier Sense (CRS) signal.

• **Preamble**—Consists of 7 bytes. The preamble is an alternating pattern of ones and zeros that tells receiving stations that a frame is coming, and that provides a means to synchronize the frame-reception portions of receiving physical layers with the incoming bit stream.

• **Start-of-frame delimiter**—Consists of 1 byte. The start-of-frame delimiter is an alternating pattern of ones and zeros, ending with two consecutive 1-bits indicating that the next bit is the left-most bit in the left-most byte of the destination address.

• **Destination MAC address**—Consists of 6 bytes. The destination MAC address field identifies which end node(s) should receive the frame. The left-most bit in the destination MAC address field indicates whether the address is an individual address (indicated by a 0) or a group address (indicated by a 1). The second bit from the left indicates whether the destination MAC address is globally administered (indicated by a 0) or locally administered (indicated by a 1). The remaining 46 bits are a uniquely assigned value that identifies a single end node, a defined group of end nodes, or all end nodes on the network.

• **Source MAC addresses**—Consists of 6 bytes. The source MAC address field identifies the sending node. The source MAC address is always an individual address and the left-most bit in the source MAC address field is always 0.

• **Length/Type**—Consists of 2 bytes. This field indicates either the number of MAC-client data bytes that are contained in the data field of the frame, or the frame type ID if the frame is assembled using an optional format. If the Length/Type field value is less than or equal to 1500, the number of Logical Link Control (LLC) bytes in the Data field is equal to the Length/Type field value. If the Length/Type field value is greater than 1536, the frame is an optional type frame, and the Length/Type field value identifies the particular type of frame being sent or received.

• **Data**—Is a sequence of *n* bytes of any value, where *n* is less than or equal to 1500. There is a nonzero minimum data size requirement because the standard states that valid

frames must be at least 64 bytes long from destination MAC address to frame cyclic redundancy check (84 bytes, including inter-frame gap, preamble and start-of-frame delimiter).

If the data portion of a frame is less than 46 bytes, the pad field is used to fill out the frame to the minimum size. There are two reasons for this minimum size limitation. First, it makes it easier to distinguish valid frames from "garbage." When a transceiver detects a collision, it truncates the current frame, which means that stray bits and pieces of frames frequently appear on the cable. Second, it prevents a node from completing the transmission of a short frame before the first bit has reached the far end of cable, where it may collide with another frame.

• **Frame cyclic redundancy check**—Consists of 4 bytes. This sequence contains a 32-bit cyclic redundancy check (CRC) value, which is created by the sending MAC and is recalculated by the receiving MAC to check for damaged frames. The frame cyclic redundancy check is generated over the destination MAC address, the source MAC address, Length/Type, and Data fields.

3.3.2 The Demand for Real-Time Ethernet

The demand for Ethernet as a real-time control network is increasing as manufacturers realize the benefits of employing a single network technology from the boardroom to the plant floor. Decreased product costs coupled with the possibility of overlapping training and maintenance costs for information, field level, control and possibly device networks would greatly reduce the expense to the manufacturers [33].

Ethernet offers many benefits at the real-time control level over existing solutions. As a control network, fast Ethernet and gigabit Ethernet offers bandwidth that is much more faster than today's comparable fieldbus networks (such as the 1 Mbps and 2.5 Mbps of commonly used FCS) and can also support real-time communication. Distributed applications in control environments require tight synchronization so that

the delivery of control messages can be guaranteed within defined message cycle times [34]. Typical cycle times for control applications are listed in Table 3-1. Traditional Ethernet and fieldbus systems are not capable of meeting cycle time requirements below a few milliseconds, however the emerging real-time Industrial Ethernet solutions allow cycle times as low as a few microseconds.

Table 3-1

Typical Cycle Times for Control Applications	
Control Application	**Typical Cycle Time**
Low speed sensors (e.g. temp. pressure)	Tens of milliseconds
Drive control systems	Milliseconds
Motion control (e.g. robotics)	Hundreds of microseconds
Precision motion control	Tens of microseconds
High speed devices	Microseconds
Electronic ranging (e.g. fault detection)	Hundreds of nanoseconds

Along with the increased bandwidth and tight synchronization, real-time Ethernet gives manufacturers the security of using a physical and data-link layer technology that has been standardized by both the IEEE and the ISO. Ethernet can provide reduced complexity with all the attributes required of a field, control or device network - in operations having up to 30 different networks installed at this level [35]. Furthermore, Ethernet devices can also support TCP/IP stacks so that Ethernet can easily gate to the Internet. This feature is attractive to users since it allows remote diagnostics, control, and observation of their plant network from any Internet-connected device around the world with a license-free web browser. Although Ethernet does introduce overhead through its minimum message data size (46 bytes), which is large in comparison to

existing control network standards, its increased bandwidth, standardization and integration with existing plant technology should generate good reasons to consider Ethernet as a control network solution.

3.4 Making Real-Time Ethernet a reality

3.4.1 Ethernet and CSMA/CD

As introduced above, current Ethernet standards promise transfer speeds of 100Mbit/s (Fast Ethernet) and more (Gbit Ethernet). This is significantly faster than most of the established fieldbus systems and embedded networks. Therefore Ethernet is seems to be a good choice for connecting industrial devices with *hard* real-time requirements. However, deterministic behavior of data transfer is more crucial than the network bandwidth [36]. With Ethernet in particular, data communication can be delayed unpredictably. This is not acceptable for dynamic industrial control applications.

The reason for Ethernet's non-determinism is its stochastic media access mechanism CSMA/CD (Carrier Sense Multiple Access /Collision Detect). Using CSMA/CD, each node monitors the wire before sending data. If the node sees that the wire is idle, it begins transmitting. Otherwise, it waits until the wire is free. Most of the time, monitoring the wire ensures that there is no conflict (see Figure 3-7).

Ethernet is fast enough that the time the node waits for the wire to be free causes only a small bounded delay.

There is a chance, however, that two nodes will be waiting for the same transmission to end. In this case, they will both detect an idle wire and start transmitting at the same time (see Figure 3-8). This causes a collision, and the two nodes must arbitrate for access.

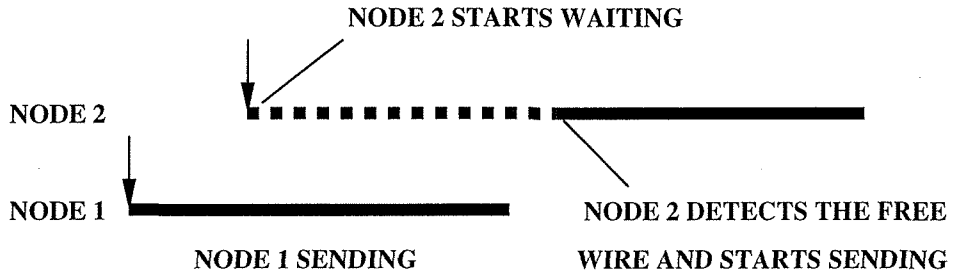


Figure 3-7. When a second node arrives while the wire is busy, it simply waits for its turn to send. The first packet's transmission time is the only delay.

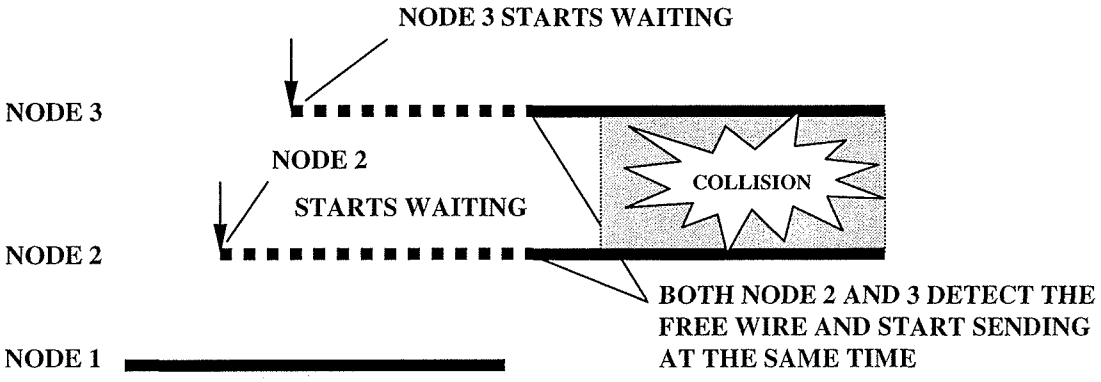


Figure 3-8. Most collisions occur when there are two nodes waiting to use the wire. When the first node completes transmission, two other nodes will start trying to send at the same time.

When a node notices a collision, it backs off and waits up to two time slots before retrying. If the retry fails, the maximum wait time is doubled, and the node waits a random period again. The algorithm continues until the network is acquired. On a 100

Mbps network, one slot is only 5.12 μs. But the algorithm cannot go forever, and it stops doubling after 10 attempts (1024 slot delays) and declares an error after 16 attempts. Thus, the exponential back-off algorithm causes unpredictable delays. It is obvious that this method is not suitable for real-time networking.

3.4.2 Ethernet switch

For obtaining deterministic characteristics, the concept of switched Ethernet is then proposed by utilizing one device, called the Ethernet switch, to eliminate the collision domain by adding only one point-to-point connection between one of the switch ports and the device on the network or to reduce the size of the collision domain by subdividing the whole network into several segments. The switch is a layer-2 device according to the ISO-OSI seven-layer model and can both receive and forward messages based on their MAC addresses and certain message dispatching algorithms [37]. Besides avoiding message collision, the switch can also act as a traffic flow controller by sending queue status back to the senders to pause message transmission. Flow control is a mechanism for limiting traffic load to a certain level. While remembering traffic smoothing, flow control may be crucial to avoid the degradation of real-time communication networks performance [38].

Flow control is an important switch feature that eliminates dropped packets on congested full duplex ports (on half-duplex, congestion is avoided through carrier-sense jamming). Flow control achieves this objective by "warning" stations that are overloading the switch.

An Ethernet switch in general consists of a certain number of input and output ports, memory or buffer, microprocessors, and the switching hardware [39]. A full-duplex Ethernet switch with N input/output ports is shown in Figure 3-9.

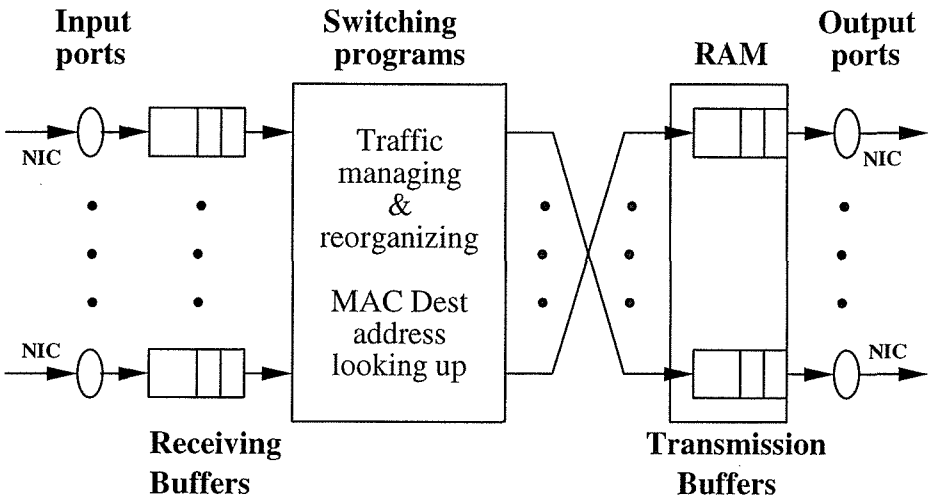


Figure 3-9. Model on Ethernet switch

The capability of an Ethernet switch is determined by the numbers of input/output ports, memory size, and the microprocessor speed. And its performance is then characterized by the switching latency, frame forwarding delay, and buffering delay etc. The numbers of input and output ports are the numbers of devices or segments of devices that the switch can connect to simultaneously. The memory or buffer area is used to store the packets to be forwarded and the network topology information near the switch.

Switches provide a flexible and scalable solution to the problems and limitations inherent to Shared Ethernet networks (bus or hub-based), through the use of new mechanisms such as micro-segmentation and full-duplex operation. These mechanisms may improve the timing determinism and performance of Ethernet to a great extent.

Ethernet networks have been based on hubs or buses where network nodes shared the same physical medium (broadcast network). This means that only one node could send data at a time. If the number of nodes increased (potentially increasing the network load), a bridge or a router could be used to segment the traffic, splitting the network into

different collision domains.

A switch is an intelligent hub that can read and process the destination address of the incoming data and sends it only to the required ports [40]. While in a hub/repeater, data sent by one node will be broadcast to all other nodes, in a switch, data is only sent to the destination node(s), which means that several nodes can transmit at the same time. A bridge also overcomes such problem (common traffic between segments), but lacks some of the technological features inherent to switches (VLANs, flow control, etc.). Moreover, a switch has several ports, while a bridge usually has only two.

Comparing to routers, switches make their forwarding decisions based on link-layer addresses, while routers must parse network layer (Layer 3) header information, and make changes in the packet, before forwarding it to the destination. Thus, a switch avoids the processing overhead inherent to a router.

3.4.3 Micro-segmentation with full-duplex switched Ethernet operation

If segmentation in a switch is taken to an extreme, each device is isolated in its own segment and has the entire port throughput for its own use. Micro-segmentation removes one of the primary causes of LAN congestion [41]. This means that there are no more problems with the aggregation of traffic from multiple stations, since there can be only one node offering load to the segment at any given time. In these conditions, the congestion burden is shifted to the central switch. This may not be a problem, as the switch can be built to handle the total aggregate load of all the attached devices (non-blocking feature). A possibility of congestion still remains due to traffic patterns causing load to converge on a given port. If many devices are all attempting to communicate with a single device, then it is possible to have a congestion problem, even though every device has its own LAN (flow control techniques must be used to overcome this problem).

Ethernet is normally a half-duplex communication system. While data can be transferred in both directions, a station is either transmitting or receiving at a given time. On the original physical media used with Ethernet (i.e., coaxial cable), this was the only way to communicate, since the same wire (and frequency) was used for transmission and reception.

A micro-segmented topology (where at most one device is connected to each port), assures that there is only one device wishing to use each wire pair. Only the attached end node ever speaks to the switch (using the transmit pair of the cable), and only the switch ever speaks to the attached end node (using the receive pair of the cable) – full-duplex operation. (The set-up of Figure 3-10 is for illustration purposes, and unlikely to be implemented in a practical form).

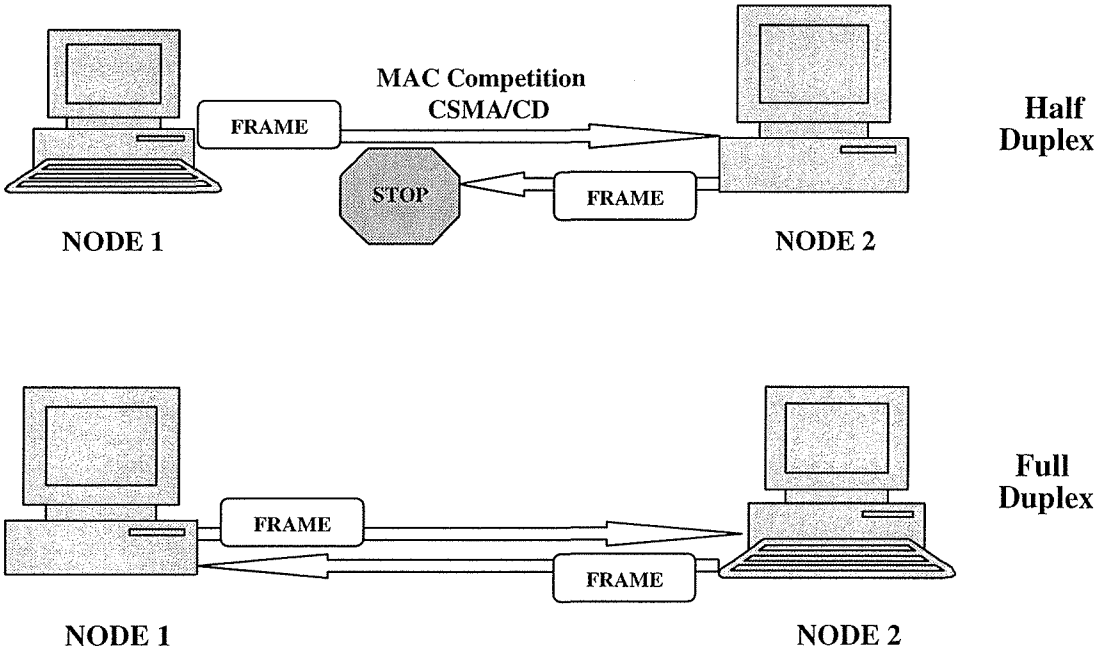


Figure 3-10 — Half Duplex and Full Duplex

There is never contention between end nodes for the use of the medium, oppositely to a Shared Ethernet topology.

With the possibility of contention removed, there is no longer any real need for the collision detection and back-off function. They can be eliminated and both the end node and the switch are allowed to transmit at will, in both directions simultaneously – full-duplex operation. Theoretically, this mode of operation doubles the data transfer rate.

3.4.4 Advances in switching

Another very important difference between a switch and a bridge is that the switch is able to behave like a massively parallel LAN bridge [42], allowing information on any port to pass to any other port, simultaneously.

LAN switches may have two basic switching modes – cut-through and store-and-forward. Cut-through provides *on-the-fly commutation*, i.e., a packet is forwarded (by the switch) as fast as possible. The switch inspects each incoming frame for the destination address of the target. It quickly determines the appropriate output port by consulting its internal address map (similar to a bridge). If the output port is available, the switch immediately forwards the frame to the destination, reducing the latency inherent to most bridge architectures (that have to receive the entire frame before making a forwarding decision). A store-and-forward switch, on the other hand, accepts and analyses the entire packet before forwarding it to the destination. It takes more time to examine the packet, but it allows the switch to catch certain packet errors and keep them from propagating bad packets through the network.

Today, the speed of store-and-forward switches has caught up with cut-through switches to the point where the difference between the two is minimal [43]. Also, there are a large number of hybrid switches available that mixes both architectures, at choice.

Intel [44] considers a third switching mode – fragment-free – that filters out most

error packets but does not necessarily prevent the propagation of errors throughout the network. It also considers an adaptive switching technology that chooses the optimal forwarding mode (for each port independently) based on real-time error monitoring.

Another important issue in switching is the blocking/non-blocking feature. Considering a switch specification and adding up all the ports at the theoretical maximum speed, the result will be the theoretical sum of the switch throughput. If the switching bus, or switching components cannot handle the theoretical total bandwidth, the switch is considered to be a *blocking switch;* otherwise it is a *non-blocking switch.* For most applications, a *blocking switch* that has an acceptable and reasonable throughput level will work well.

Consider an eight port 10/100 switches as shown in figure 3-11. Since each port can theoretically handle 200 Mbit/s (full duplex), there is a theoretical need for 1.6 Gbit/s. Though, considering the simultaneity coefficient, each port will not probably exceed 50% utilization, so an 800 Mbit/s switching bus might be adequate.

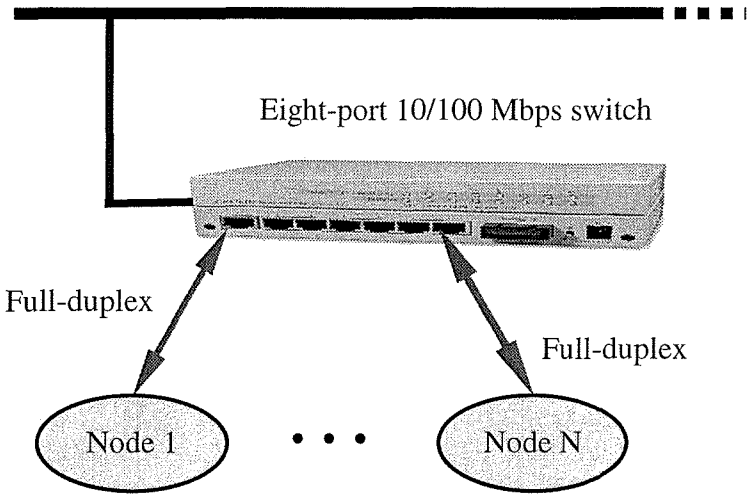


Figure 3-11. An eight port 10/100 Mbps switch with full duplex

When designing a real-time communication system over Switched Ethernet, the blocking/non-blocking characteristic and how low/high should be the data-rate (in a blocking solution), becomes an important issue that should be object of a thorough analysis.

3.5 Switched Ethernet for hard real-time communications

3.5.1 Conventional hard real-time communication protocols

With the increasingly demand for real-time industrial systems, the ability of computer networks to handle deadline guaranteed "hard" real-time communication is becoming more and more important [45-47]. High bandwidth and determinism are the critical necessary conditions for hard real-time applications. Unlike traditional, bus-based CSMA/CD Ethernet, the switched Ethernet is a star-based topology which can provide collision domain to each of the ports of a switch. Therefore, end-node cooperation is needed only for bandwidth control, not any more to avoid collisions. In addition, current Ethernet standards promise transfer speeds of 100Mbps (Fast Ethernet) and more (Gbit Ethernet). This will significantly improve Ethernet bandwidth available to real-time communication applications.

Several researches have been done to treat hard real-time communication. MIL-STD-1533 [48] is an early development of the industrial protocol. It is reliable standard interface for token ring LAN. However, bandwidth has become a limiting factor for it. RTCC [49] (Real-Time Communication Control) is centralized approach that has the disadvantage that failure in the controller will lead to the entire network useless, unless some sort of recovery protocol is implemented. Another research using switched Ethernet for hard real-time communication is studied in [50]. The main idea of the work is traffic shaping to every end-node on the switched Ethernet network. This enables the network shared with non real-time nodes; however, the traffic shaping

capabilities also bring about time delay. An Ethernet access protocol called EDF-CSMA [51] has been proposed, which can also deal with soft and non real-time communication. This method provides a solution to handle real-time messages with strict deadlines, making it suitable to a wider range of real-time applications. However, hardware modification is necessary to implement this protocol. Tutorial of hard real-time communication in packet switched networks is found in [52].

3.5.2 Advanced network technologies based on admission control

Recently, new schemes have been proposed based on call admission control (CAC) depending on quality of service (QoS) and choice of the packet service discipline. The common concept of the schemes is establishment of a *Real-time Channel*: a simplex, virtual connection between source nodes and destination nodes, with a priori guarantees for communication performance in switching networks [53-55].

To support the QoS demands of applications, both the ATM and the IP community have defined service classes that provided per-flow guarantees to applications [56-57]. In order to provide guaranteed services, resources need to be reserved for every accepted connection. ATM does this at the data-link layer. For every call it reserves a virtual channel over all links on the route/switch from the source to the destination. At the network layer, resource reservation can be done using Multi Protocol Label Switching (MPLS) or Differentiated Services (Diffserv) developed by the Internet Engineering Task Force (IETF) [58-59].

MPLS is a network paradigm that provides for efficient routing, forwarding, tunneling and switching of traffic through a network. Within an MPLS network, traffic is sent on label-switched paths (or LSPs). A LSP is a path that has been established between a source and destination and where every node contains a sequence of label identifiers. Depending on the traffic characteristics, different LSPs could be created for packets with different QoS or CoS requirements. As a result, high-speed data switching

is possible since hardware is able to switch packets quickly between links based on the MPLS label.

MPLS is capable of multiprotocol support since the Forward Equivalent Classes can combine any of the network layer protocols and their associated routing protocol information. In this way, MPLS can work with a large variety of commonly used link-layer mediums such as ATM, Ethernet, Gigabit Ethernet, and even Token Ring. Additionally, MPLS supports all types of forwarding: unicast, unicast with type of service, and multicast packets. MPLS offers enhanced routing capabilities such as traffic engineering, QoS-based forwarding, and VPNs (virtual private networks). However, an end-to-end solution is only possible if an entire network is MPLS-enabled. In other words, MPLS is domain-specific. Because of this specificity, on the one hand, MPLS naturally finds its greatest strength in core networks. On the other hand, a large infrastructure overhaul to existing IP carrier equipment (routers, switches, carrier networks) is needed to implement.

Furthermore, applications at this level are classified as best-effort, rate sensitive or delay sensitive, therefore, MPLS has no guarantee for strict deadline-sorted hard real-time communication.

Diffserv is an architecture for providing different types or levels of service for network traffic. One key characteristic of Diffserv is that flows are aggregated in the network, so that core routers only need to distinguish a comparably small number of aggregated flows, even if those flows contain thousands or millions of individual flows. DiffServ stands for differentiated services and refers to the concept of providing a different priority for different classes of traffic. DiffServ is implemented by using an available IP header field to indicate the priority that a packet should receive.

The advantage of DiffServ is that all the policing and classifying is done at the boundaries between DiffServ clouds. This means that in the core of the Internet, routers can get on with doing the job of routing, and not care about the complexities of

collecting payment or enforcing agreements.

However, one of the disadvantage is that the details of how individual routers deal with the type of service field is somewhat arbitrary, and it is difficult to predict end-to-end behaviour. This is complicated further if a packet crosses two or more DiffServ clouds before reaching its destination. Also, DiffServ only enables prioritization of one packet type over another, with no guarantee that even high-priority packets will not be affected by congestion. Therefore, it will also not suitable for hard real-time communication.

Furthermore, to provide different QoS commitments, the IETF developed the *integrated services model* that requires resources such as bandwidth and buffers to be explicitly reserved for a given data flow to ensure that the application receives its requested QoS. The Resource Reservation Protocol (RSVP) is used by the integrated services model to provide the reservation messages required to set up a flow with a requested QoS across the network. In RSVP, the admission control and resource allocation policies are based on different levels of QoS specifications --- deterministic or guaranteed, statistical and best-effort. RSVP requests resources for simplex flows, i.e., it requests resources in only one direction from a sender to a receiver. Therefore, RSVP treats a sender as logically distinct from a receiver, although the same application process may act as both a sender and a receiver at the same time. It is receiver-oriented in that the receiver of the data flow is responsible for the initiation and maintenance of the resource reservation and quality of service (QoS) specification.

The main feature of RSVP is the maintenance of *soft-state* at each intermediate router or switch: a resource reservation at an intermediate station is maintained for a limited time only, therefore, the sender must periodically refresh its reservation. This approach allows RSVP to adapt to network load and outages. RSVP is designed to deliver resource reservation requests to related switches. Therefore, RSVP is more appropriately a *state establishment protocol.* The resource reservation is based upon the

link's ability to satisfy the user specified QoS requirements mapped to its domain. This function is done by its two local control modules: Admission Control and Policy Control at each intermediate station in the connection.

Figure 3-12 illustrates the resource reservation process of RSVP. To make a resource reservation at a node, the RSVP daemon communicates with admission control and policy control. Admission control determines whether the node has sufficient available resources to supply the requested QoS. Policy control determines whether the user has administrative permission to make the reservation. If either check fails, the RSVP program returns an error notification to the application process that originated the request. If both checks succeed, the RSVP daemon sets parameters in a *packet classifier* and *packet scheduler* to obtain the desired QoS. The packet classifier determines the QoS class for each packet and the scheduler orders packet transmission to achieve the promised QoS for each stream.

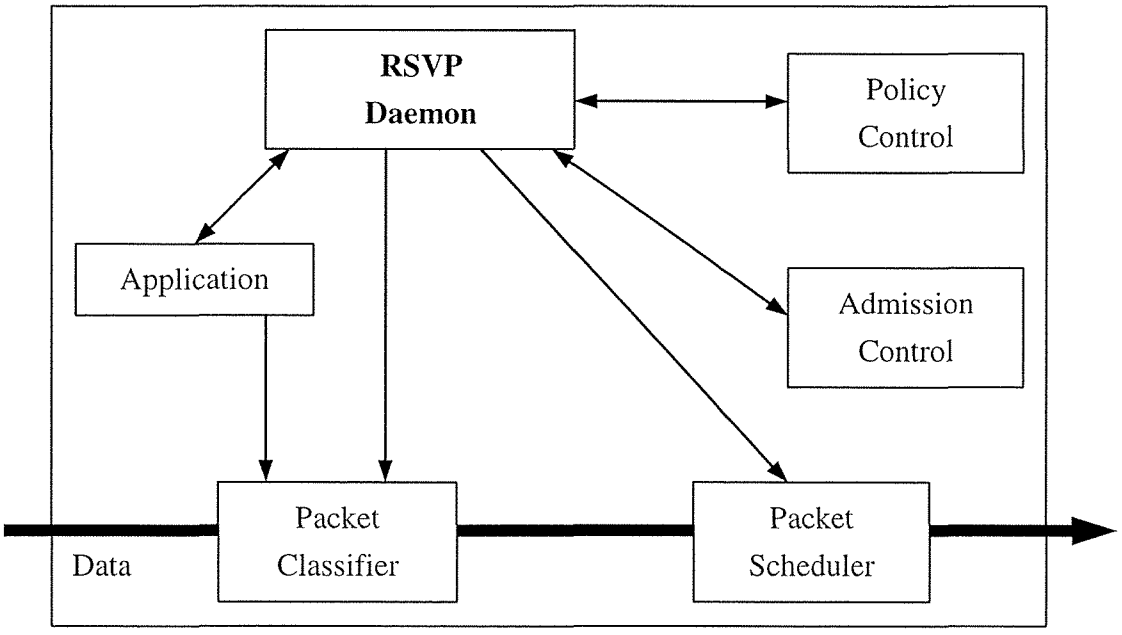


Figure3-12. Resource reservation process

RSVP makes resource reservations for both unicast and many-to-many multicast applications. Therefore, it is more appropriate for a large group communications. To the best of our knowledge, RSVP is mostly used for customized multimedia and online meetings such as videoconferencing. RSVP is IP based protocol, and there is no guarantee of deadline in application service lifetime, and have large runtime overhead, therefore, there are few applications in industrial real-time communication control systems.

However, one of the goals in our work is to use cheap existing hardware and software whenever possible. This requires that the network used should at least be based on existing hardware. Customized hardware would drive up the price of devices considerably and take up a lot of time to design and implement.

3.5.3 TCP/UDP/IP for hard real-time Ethernet ?

With industrial Ethernet, there is a tendency to define an application layer environment along with the TCP/IP protocol, to realize an industrial automation networking application. Although some real-time Ethernet applications (e.g. EtherNet/IP) perform all their communication (including real-time communication) through the TCP/UDP/IP stack and provide compatibility with TCP/IP, most solutions do not employ this protocol for hard real-time communication. In a system like EtherNet/IP, TCP is used for initialization and configuration (explicit) messages while UDP, with its reduced overhead, is used for real-time I/O (implicit messaging).

For hard real-time Industrial Ethernet applications, the proposed protocol compatible with TCP/IP. The ability of the proposed protocol for hard real-time Ethernet application to intercommunicate with an office-based system is paramount to achieve the Ethernet-technology plant.

3.5.4 Proposed hard real-time communication protocol

Based on the above-mentioned QoS architectures and protocols, the proposed protocol in our work also establishes a way (virtual link) between source nodes and destination nodes by applying admission control based upon the requested QoS. However, in our work, a key strategy to realize hard real-time communication is "bypassing" of TCP/IP suites. The proposed protocol manages hard real-time traffic to bypass the TCP/IP stacks. This makes considerably reduce the dwell time in the nodes, and increase the achievable data frame rate by evasion of the non-deterministic behavior inherent in the TCP and IP stacks.

In our star-like network architecture, every end-node is connected with a private virtual link to a switch, so that there is a private traffic link for each direction. Then collisions can no longer occur on any network cable.

Present-day switches employ a technique called store-and-forward to transfer packets from one port to another, using per-port buffers for packets waiting to be sent on that port. But congestion may occur when one node is suddenly receiving a lot of packets from the other nodes. Current switches do not provide any guarantees as to which packets will be sent first. We solved this by providing a switch with bandwidth reservation capabilities inside the switch, and we used Earliest Deadline First (EDF) scheduling to make decisions as to which packets are forwarded first. This provides guarantees for both bit rates and strict delivery deadlines.

In our work, a software called real-time layer (RT layer) is added between the Ethernet protocols and the TCP/IP suite in each end nodes and switch to support both bit rate and maximum latency guarantees for hard real-time traffic. The guarantee is uphold by real-time channels (RT channel), which can be created and closed dynamically. An RT channel can be considered as a virtual wired connection between two nodes connected to the switch. The nodes, which are connected to the switch, can either be real-time nodes or non real-time nodes (nodes without RT layer added)

depending on which level QoS is required.

The added RT layer in the nodes enables applications to reserve real-time channels on the network subject to a feasibility analysis. In this way we guarantee bandwidth for real-time traffic. RT channel is established between source nodes and destination nodes according to the RT layer, not defines a way of encapsulating other layer 2 and layer 3 protocols that the other protocols does.

The proposed protocol does not need any modifications in the Ethernet hardware on the network interface cards, and coexists with TCP/IP suites. Which means that non-real-time nodes can coexist in the network, and the proposed protocol support for co-existing standard TCP/IP and UDP/IP Internet traffic without disturbing the real-time traffic, in order to support existing network systems. Therefore, the LAN with the proposed protocol can be connected to the existing Internet networks. It can be adopted in hard real-time applications such as embedded systems, distributed industrial systems, parallel signal processing and robotics.

4. Hard Real-time communication support

In this chapter, design and implementation of a switched Ethernet protocol for hard real-time communication are discussed. In section 4.1, outline of the network architecture is introduced. In section 4.2, we give a brief description of the real-time scheduling algorithms. In section 4.3, we describe the RT channel establishment by applying the admission control. Section 4.4 illustrates the management of hard real-time traffic and best-effort traffic. Lastly, in section 4.5, we elaborate feasibility analysis for RT channel establishment and focus on scheduling of real-time frames.

4.1. Network architecture

We applied a full-duplex switched Ethernet which is connectable to existing Internet. Switches enable flexible network configuration of multiple and simultaneous links between various ports. A switched Ethernet enables some key benefits over traditional Ethernet, such as full duplex and flow control. Therefore, switches enable flexible network configuration of multiple and simultaneous links between various ports. In addition to this, it directs network traffic in an efficient way, establishing a kind of direct descent of communication between two ports of each frame structure.

In our work, a key strategy to realize hard real-time communication is *bypassing* of TCP/IP suites. In order to manage this bypass, both the switch and the nodes have software --- real-time layer (RT layer) added between the Ethernet protocols and the TCP/IP suite in the OSI reference model. All nodes are connected to the switch and nodes can communicate mutually on the logical real-time channels (RT channels), it is a virtual connection between two nodes of the system respectively (see Figure 4-1).

A node can either be real-time node or non real-time node depending on which level QoS is required. Non-real-time node can coexist in the network without disturbing the real-time traffic. MAC function, frame buffering and the concentrated transmission

arbitration is included in the switch. Therefore, switch has the overall responsibility both for set-up of RT channels and for online control of packets passing through the switch.

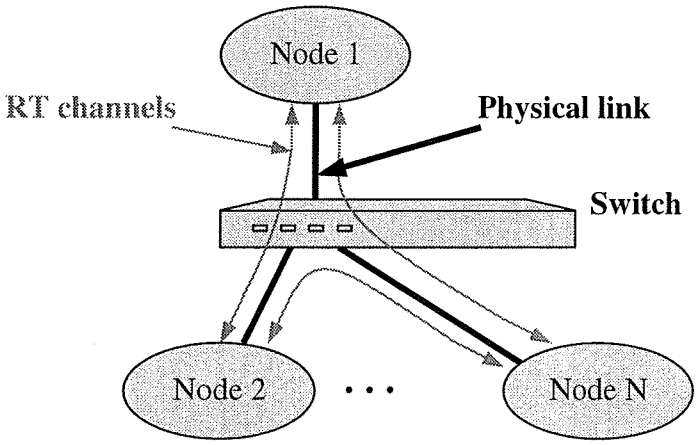


Figure 4-1. Network architecture with RT channels

Hard real-time communication is realized by dynamic priority scheduling of real-time data frames based on the earliest deadline first (EDF) algorithm which will described in the following section. The RT layer do-nothing to non real-time frames and makes them go through the ordinary circuit with TCP/IP suites. Namely, the RT layer is compatible with existing TCP/IP and handles both real-time and non real-time traffic depending on QoS.

4.2 Real-time scheduling algorithm

Real-time scheduling algorithms fall into two categories: *static* and *dynamic* scheduling [17-20] (as shown in the Figure 4-2).

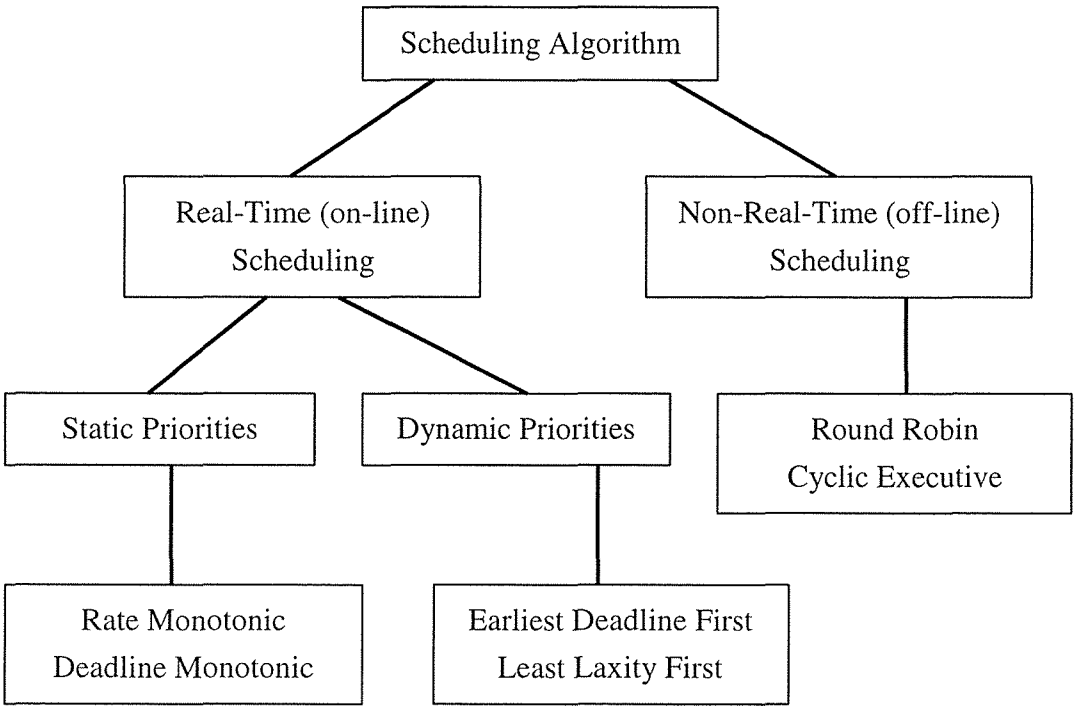


Figure 4-2. Classification of scheduling algorithms

In static scheduling, the scheduling algorithm has complete knowledge of the task set and its constraints, such as deadlines, computation times, precedence constraints, and future release times. The Rate Monotonic (RM) algorithm and its extensions are static scheduling algorithms and represent one major paradigm for real-time scheduling. In dynamic scheduling, however, the scheduling algorithm does not have the complete knowledge of the task set or its timing constraints. For example, new task activations, not known to the algorithm when it is scheduling the current task set, may arrive at a future unknown time. Dynamic scheduling can be further divided into two categories: scheduling algorithms that work in *resource sufficient* environments and those that work in *resource insufficient* environments. Resource sufficient environments are systems

where the system resources are sufficient to *a priori* guarantee that, even though tasks arrive dynamically, at any given time all the tasks are schedulable. Under certain conditions, Earliest Deadline First (EDF) [17] is an optimal dynamic scheduling algorithm in resource sufficient environments because the task with the nearest deadline for its current request has the highest priority. At any instant, the task with the highest priority and an unfulfilled request gets hold of the CPU. This contrasts with the RM algorithm in which priorities do not change with time.

The necessary and sufficient condition for the feasibility of a task set with EDF is given below:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \quad , \tag{1}$$

where, n is the total number of tasks.

The ratio of *Ci/Ti* is the CPU utilization by the *ith* task. The above condition is necessary because the tasks should not overload the CPU if they have to meet all the deadlines.

EDF is optimum in the sense that if a task set can be scheduled by any algorithm, it can also be scheduled by EDF. Jeffay and Stone [21] gave an equivalent feasibility condition:

A set of periodic tasks scheduled using EDF will be feasible if and only if for all L, $L \geq 0$ where,

$$L \geq \sum_{i=1}^{n} \left\lfloor \frac{L}{T_i} \right\rfloor C_i \ . \tag{2}$$

Dynamic priority scheduling with the EDF algorithm has a distinct advantage over fixed priority scheduling: the schedulable bound for EDF is 100% for all task sets. This means that we can fully utilize the computing power of the CPU. Embedded systems are in general fully loaded, as they attempt to get as close to 100% utilization as possible while still maintaining the necessary predictability.

EDF is another major paradigm for real-time scheduling. The EDF algorithm has the higher scheduling overhead comparing to the RM algorithm, as it has to dynamically compute priorities of tasks (run-time processing). However, this should not be a major difficulty as the computing speed of the CPU getting faster and the computing power of the CPU can be fully utilized.

A major problem with the EDF algorithm is that there is no way to guarantee which tasks will fail first if the *task overload* occur (with RM, low priority tasks are always the first to fail. However, no such priority assignment exists with EDF). Which means that the EDF algorithm can guarantee for the real-time scheduling only under the *resource sufficient* (task under loaded) environments.

In our work, before the real-time traffic is transmitted, we should apply sufficient resources in order to establish the real-time channel, by applying admission control based upon the provided QoS (which will be described in the next section). Under this circumstance, there should be no overload exist in the real-time channel, and EDF is an optimal dynamic scheduling algorithm under such condition because of its higher schedulable bound. Therefore, we used EDF scheduling algorithm in our work.

4.3 RT channel establishment

Before the real-time traffic is transmitted, the real-time channel (RT channel) should be established. The establishment of RT channel is including request and recognition communication after the source nodes, destination nodes and switch have agreement with channel establishment. As new real-time requests (channel establishment) are made, they must go through an admission control module that determines if there is sufficient network bandwidth available to satisfy the request of the node.

Admission control is the problem of deciding which requests to accept and which to reject based upon the supported QoS, with the goal of maximizing the total profit accrued by the accepted requests. In other words, admission control is the problem of

finding a feasible solution with maximum profit.

Admission control deals with the question as to whether a switch can accept a new connection or not based upon the constraint to meet the negotiated QoS requirements of all existing connections as well as of the new connection. Typically, the decision to accept or reject a new connection is based on the following two criteria:

1. Resource Availability: Are there enough resources to support the new connection without affecting the guarantees of existing connections?
2. Policy Restrictions: Does this connection satisfy administrative restrictions, maximum number of open connections in its traffic class, source node/destination node addresses etc.?

The decision on resource availability is made by keeping a running total of resources already committed to existing connections. The running total is queried to determine if the resources required by the new connection would violate a guarantee given to some other connection. The connection parameters are then checked for compliance with policy restrictions. If the above criteria are both satisfied, the connection is accepted.

If a new connection is accepted, bandwidth and/or buffer space in the switch is allocated for this connection. The allocated resources are released when the connection is completed.

If the system includes critical tasks, the worst-case values are used in the analysis, otherwise average values are most common. When determining the worst-case completion time for a task, not only consider the execution time of the task, but also consider execution interference from other higher priority tasks and any potential preemption. If a task is accepted in an admission control system, it is typically guaranteed a certain quality of service (QoS). Usually, there are different levels of QoS available in the system. Using different priorities is one way of ensuring different levels of quality. One reason for having several QoS levels is that most systems need to be able to support both real-time and non-real-time tasks concurrently. The relative

deadline of a task is defined as the difference between its deadline and its release time.

As for the establishment of an RT channel, each connection request is specified by two distinct nodes, the sending node and the receiving node in the network that want to communicate. Each transmission request has a certain bandwidth requirement and some time specification given by its starting time and its duration. If the network establishes a transmission request, it first decides on a path from the sending node to the receiving node of that transmission, through which the transmission is being routed. Then it allocates the requested amount of bandwidth on all links along that path during the time period in which the transmission is active.

Figure 4-3 describes the establishment of an RT channel. The RT channel should be established between a source node and a destination node according to the RT layer that is added between the Ethernet protocols and the TCP/IP suites. Each node is connectable to multiple sending and receiving RT channels. When a node wants to send hard real-time frames, it directly accesses the RT layer. The RT layer then sends "RT channel establishment request" to the RT traffic management in the switch (see Figure 4-4). The switch then evaluates the feasibility of traffic schedule of a path from the sending node to the receiving node of that transmission, by applying the admission control. If the schedule is feasible, the switch responses with the network schedule parameters (see Figure 4-5) to the sending node. Otherwise, the switch sends out a set of recommended control parameters to the sending node. These control parameters are suggested based on the status of switch queue and the active queue control law.

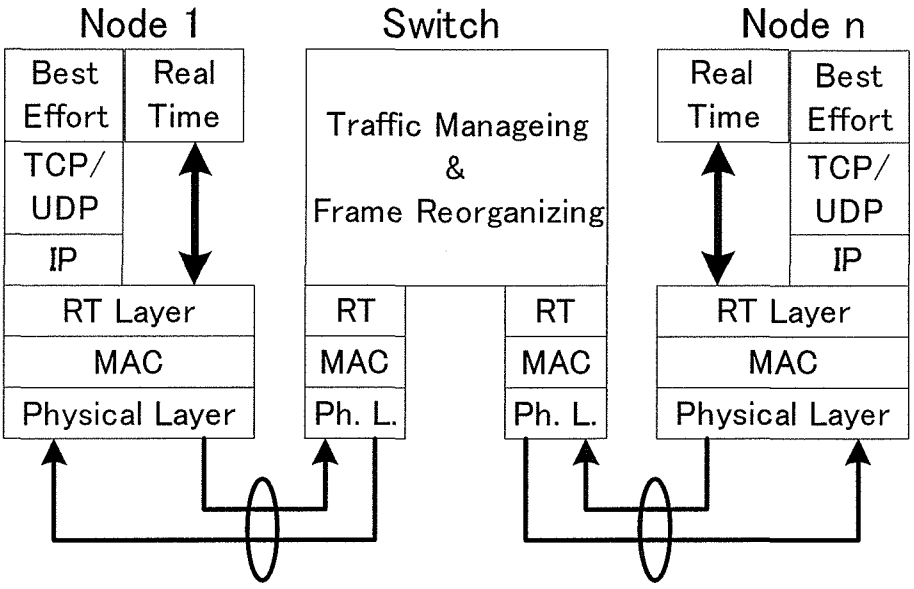


Figure 4-3. Real-Time channel establishment

Data field in Ethernet-frame containing details about the connection request

Dest. MAC addr. = switch addr. 6 bytes	Type: Connect packet 1 byte	MAC source address 6 bytes	MAC dest. address 6 bytes	IP source address 4 bytes	IP dest. address 4 bytes	T_p 4 bytes	C 4 bytes	T_d 4 bytes	Connect. request ID 1 byte	RT channel ID 2 bytes

Figure 4-4. Request Frame sent by source node trying to establish an RT channel

Source MAC addr. 6 bytes	Type: Response packet 1 byte	Response 0:Reject 1:Accept	Connect request ID 1 byte	RT channel ID 2 byte

Figure 4-5. Response parameter

4.4 Traffic management

After RT channel is established, only real-time data traffic from the end-node bypasses the TCP/IP stacks and thus considerably reduces the dwell time in the nodes, and increases the achievable data frame rate by evasion of the non-deterministic behavior inherent in the TCP and IP stacks. Here dwell time of a node refers to one of the substantial influence factors for the real-time performance. An RT channel should cross two physical links: one from the source node to the switch, and the other from the switch to the destination node (uploads and downloads, respectively). The RT channel is required to provide real-time guarantees for both the uplink and the downlink.

Besides hard real-time traffic, our Ethernet network protocol should allow for best-effort traffic which must not affect the transmission of hard real-time packets. Namely, best-effort traffic (non real-time or soft real-time traffic) come from best-effort protocols (HTTP, SMTP, FTP, etc.) uses the services of the TCP/IP protocol suites and put in an FCFS-sorted (First Come First Serve) queue in the RT layer. In order to achieve this, best-effort traffic is allowed when no hard real-time packets want to transmit. If a best-effort sender needs higher bandwidth to send a large amount of data (for example, a long packet), it tries to make an additional reservation and transmit its data immediately after reservation. If the time is over and the long packet did not finished yet, it tries to make a reservation again.

When a hard real-time packet becomes ready to transmit again, the RT channel management should immediately interrupt the best-effort traffic and go to the corresponding node, so that the hard real-time traffic may start. According to the RT layer, the last node visited for best-effort traffic should be remembered, so the next round of best-effort traffic packet can start off at that node.

Because there are two different output queues for each port on the switch, "frame recognizing" is necessary. On that account, the switch has two MAC addresses: one is for control traffic (e.g., RT channel request frames); and the other is for real-time traffic

over RT channels. And then, the switch will be able to recognize the different kinds of frames: control frames, real-time data frames and non real-time data frames that come from TCP/IP stacks. The end-nodes recognize control frames by reading MAC source address that is set to the switch address. Non real-time frame carries the final destination MAC address in the Ethernet header when it leaves from the source node.

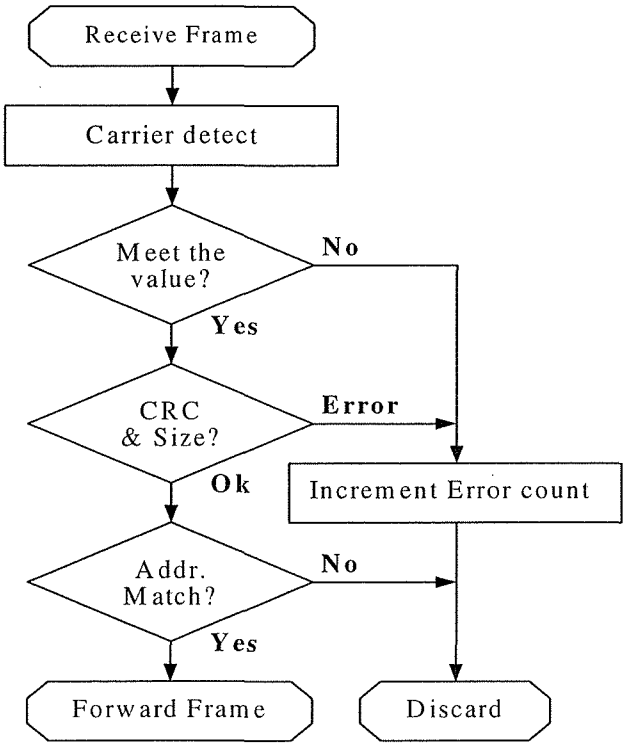


Figure 4-6. Real-time data frame processing

Another important role of the RT layer is to handle "RT data frame processing". The RT layer makes the switch recalculate the Ethernet cyclic redundancy check (CRC) of an incoming real-time frame before putting it to the correct deadline-sorted output queue (see Figure 4-6). This will also be helpful to increase the reliability of the hard

real-time data frames, since for a hard real-time traffic, the reliability also is a very important factor as a deadline. The checksums of non real-time frames do not need to be recalculated.

4.5 Scheduling of real-time frames

In this subsection, we elaborate feasibility analysis for RT channel establishment and scheduling for traffic management.

We assume that Node 1 (source node) wants to send real-time traffic to Node 2 (destination node). The real-time guarantee is supported by RT channel, and the scheduling of real-time frames in the switch and end-node is made according to EDF theory. First, we check the feasibility of the real-time traffic according to calculate the total utilization of all the request frames. RT channel of the i-th task is characterized by $\{T_{pd,i}, C_i, T_{d,i}\}$; where $T_{pd,i}$ is period of the data, C_i is time required to complete the execution of the task per the period, $T_{d,i}$ is the relative deadline used for the end-to-end EDF scheduling.

The task period $T_{pd,i}$ can be described as:

$$T_{pd,i} = T_{n1,i} + T_{n2,i} + T_{ct}\,, \tag{3}$$

where $T_{n1,i}$ and $T_{n2,i}$ are the deadlines of each real-time frame for upload and download, respectively; and T_{ct} is the corner-turn delay introduced by the switch. In a fully switched Ethernet there is only one equipment (end-node) per each switch port. In case that wire-speed full-duplex switches are used, the end-to-end delay can be minimized by decreasing the message buffering. A frame traveling through the switch in its path without any buffering has the minimum delay. The total corner-turn delay introduced by a switch including:

. The switching latency (including traffic classification and switch fabric set-up time),

. The frame forwarding latency which depends on the forwarding mode and eventually

on the frame length if the "store & forward" mode is running,

. The buffering delay when the frame is queued.

The switching latency is a fixed value which depends on the switch performance and often provided by the switch vendor; the frame forwarding delay can be obtained knowing in which mode the switch is running; buffering delay exists in a switch whatever the switch is with full wire-speed or not. In fact message buffering occurs whenever the output port cannot forward all input messages on time. However, for the input traffic, scheduling analysis can give the worst-case buffering delay, providing thus the hard real-time guarantee.

Because the system is full duplex, for each link we consider two independent tasks; one is about the download parts through the link, and the other is the upload parts. The task of the upload link and download link is then executed with the set of instruction queue in the order decided by the switch.

According to EDF theory, the total utilization of all the request frames is then calculated as:

$$U = \sum_{i} \frac{C_i}{T_{pd,i}} . \tag{4}$$

Suppose $T_{pd,i} \leq T_{d,i}$ for simplicity, it is well known that EDF scheduling is feasible if and only if $U \leq 1$.

If the test for task *i* succeed and real-time channel is established, real-time data frame bypasses the TCP/IP stacks and put in a deadline-sorted queue scheduled by RT layer in the switch and end-nodes according to the EDF theory.

The processes of hard real-time traffic and best-effort traffic transmission (see Figure 4-7) are summarized as follows:

1. When the switch received a packet from an end-host, it recognizes which application the packet is come from (real-time or best-effort).

2. If the packet come from the real-time application, then the RT layer interrupt transmitting best-effort traffic immediately so that the hard real-time traffic may start. And the information for last transmitted queue of the best-effort traffic should be stored so that the next round of best-effort traffic can start off at that queue.

3-A. The switch will be able to recognize the different kinds of frames: control frames (e.g., RT channel request frames) and real-time data frames. If the received packet is the control frame, then the RT layer must go through an admission control module that determines if there is sufficient network resource (bandwidth and guaranteed time limit) available to satisfy the request of the node.

4-A. If a request is admitted, the switch responses with the network schedule parameters to the sending node, and make RT channel virtual connection. And then allocates the requested amount of bandwidth and/or buffer space on this connection link.

4'-A. Otherwise, the switch sends out a set of recommended control parameters to the sending node.

5-B. After RT channel is established, hard real-time data is delivered through the circuit and bypassing the TCP/IP stacks by reading MAC addresses in response parameter.

6-B. The RT layer makes the switch recalculate the Ethernet cyclic redundancy check (CRC) of an incoming hard real-time frame before putting it to the correct deadline-sorted output queue. This will also be useful to increase the reliability of the hard real-time data frames.

7-B. Real-time data passed the above check is then put in a deadline-sorted queue

scheduled by RT layer in the switch and end-hosts according to the EDF theory, and then,

8-B. Forward the deadline-sorted data to the destination node. The allocated resources are released when the connection is completed.

3-C. On the other hand, by carrying the final destination MAC address in the Ethernet header when leaving from the source node, non or soft real-time data is delivered through the circuit including the TCP/IP stacks in an FCFS-sorted queue, and transmit the traffics at the idle time of the schedule.

4-C. If a best-effort sender needs to send a large amount of data (for example, a long packet), it tries to make an additional cycling time reservation and transmit its data immediately after reservation. If the time is over and the long packet did not finished yet, it tries to make a reservation again.

The protocol designed with above procedure meets the deadline for the periodic and aperiodic data frames, and also realizes faster execution of the real-time tasks.

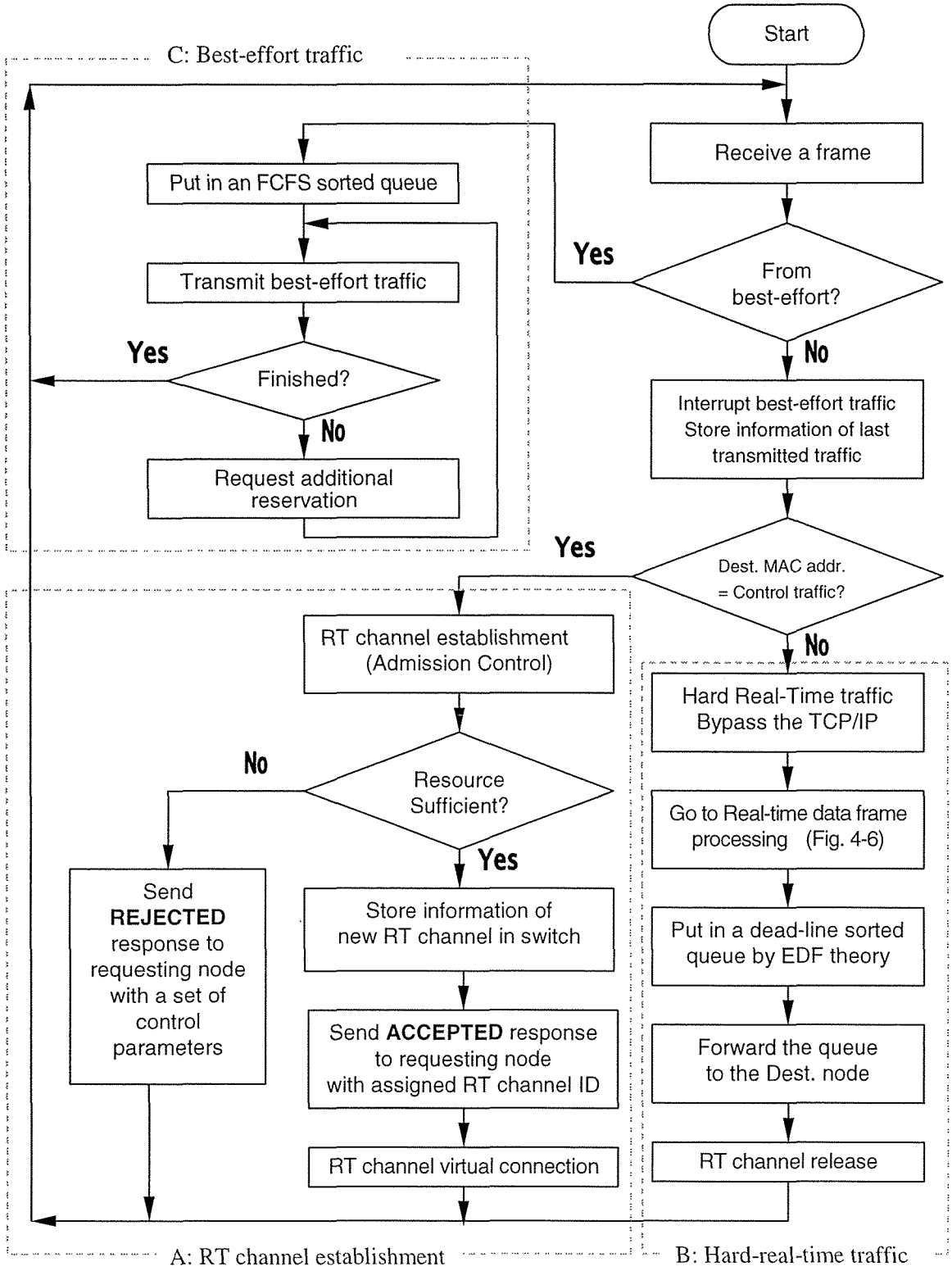


Figure 4-7. Processes of traffic transmission.

Pseudocode for the Process of traffic transmission:

#include <ctype.h>

#include <errno.h>

#include <stdio.h>

#include <stdlib.h>

ArrivalFrame[] = F; Process[] = F;

/* ArrivalFrame[] and Process[] have source MAC address, destination MAC address,

$\{T_{pd,i}, C_i, T_{d,i}\}$, remaining_time, and connection ID*/

if ArrivalFrame has only final destination MAC address **then**

TerminationTime = ∞; CurrentProcess = NULL;

/*Process[] treat as best-effort traffic*/

else

CurrentProcess ≠ NULL;

stop best-effort traffic and store last transmitted traffic address;

if MAC source address is set to the switch address **then**

Call Admission Control;

else

/*parse_input(RTin, RT_layer)*/

int parse_input(ArrivalFrame_t, task_buf)

ArrivalFrame[] = RTin_task;

Input: RTin_task to RT_layer

Goto RT data frame processing;

Pseudocode for EDF algorithm:

#include <ctype.h>

#include <errno.h>

#include <stdio.h>

#include <stdlib.h>

Input: s: packet; $T_{pd,i}$ is period of the data; q: processor; l: expected workload;

Input: T: packet set; P: processor set; W = (period(T_i)): allocation time window;

for each packet $i = 1$ to $|T|$ in increasing order of packet relative deadlines **do**

for each period $j = 1$ to $\lceil W / period(T_i) \rceil$ **do**

dl(S) = ComputeDeadline(s,l);

/* compute the deadline of s */

for processor q $\in PR$ **do**

ResponseTime(*q*) = *EDF_Response_Time_Analysis* (*s, p, q, l*);

/* determine the shortest response time */

MinResponse = *min*{*ResponseTime*(*x*)|*x* $\in PR$};

Determine p_{min} that has the shortest response time;

if *MinResponse* > *dl*(*s*) × (1-*Margen*) **then**

Return FAILURE;

else

pr(*s*) = p_{min};

Return SUCCESS;

EDF_Response_Time_Analysis (*s, p, q, l*):

#include <ctype.h>

#include <errno.h>

#include <stdio.h>

#include <stdlib.h>

Input: s: subtask; p: period relative to expected load change time; q: processor; l: expected workload;

ArrivalEvents[] = Φ; *Process*[] = Φ;

/* *ArrivalEvents*[] and *Process*[] have four fields: time, subtask, remaining_time, and resume_time */

x = *ParentTaskNumber*(s); y = *SubtaskNumber*(s, x);

/* y is the id of s within its parent task x */

StopTime = *ArrivalTime*(T_x) + $dl(T_x)$; /* time window until the absolute deadline of T_x) */

ArrivalEvents = *ArrivalEvents* $\cup[st_y^x,$ *ArrivalTime*(s), $eex(s, l / replicas(st_k^i)),$ $-\infty]$;

for each task i = 1 to (x-1) /* consider tasks with shorter relative deadlines */ **do**

for each period j = 1 to [*StopTime* / period(T_i)] **do**

for each subtask k = 1 to [$ST(T_i)$] **do**

if st_k^i has a replica executing on processor q during period j **then**

/* insert the arrival event at its arrival time position */

ArrivalEvents = *ArrivalEvents* $\cup[st_k^i,$ *ArrivalTime*(st_k^i), $eex(s, l / replicas(st_k^i)),$ $-\infty]$;

TerminationTime = ∞; *CurrentProcess* = NULL;

/* construct the schedule to determine subtask response time */

for each arrival event *k* from *ArrivalEvents* in increasing order of arrival times **do**

if *ArrivalTime*(*k*).time < *TerminationTime* **then**

t = *ArrivalEvents*[*k*].time;

if *CurrentProcess* ≠ *NULL* /* CPU is not idle */ **then**

Process[*CurrentProcess*].remaining_time = (*t* -*Process*[*CurrentProcess*].resume_time);

Process = *Process* + *ArrivalEvents*[*k*];

ArrivalEvents = *ArrivalEvents* - *ArrivalEvents*[*k*];

CurrentProcess = *LocalScheduler*(*Process*[]);

CurrentProcess.resume_time = *t*;

TerminationTime = *t* + *Process*[*CurrentProcess*].remaining_time;

else

t = *TerminationTime*;

if *Process*[*CurrentProcess*].subtask = *s* **then**

ResponseTime = *t* – *Arrivaltime*(*s*);

/* one of the subtasks of shorter deadline tasks missed its deadline */

if *t* > *Process*[*CurrentProcess*].time + *dl*(*Process*[*CurrentProcess*].subtask) **then**

Return ∞;

Return *ResponseTime*;

Pseudocode for admission control:

Reservation Estimation

on packet *p* arrival:

b <- state(*p*);

L = L + b;

on time-out T_W:

/* estimated reservation */

R_est = L / T_W;

R_bound = min(R_bound,

R_est/(1 - f) + Ra);

Ra = 0;

Admission Control

on reservation request r:

/* admission ctrl. test */

if (R_bound + r <= C)

Ra = Ra + r;

R_bound = R_bound + r;

accept(r);

else

reject(r);

4.6 Performance evaluation

4.6.1 LAN with full-duplex switched Ethernet

We made a LAN with a full-duplex switched Ethernet and end-nodes, by using desktop computer and several embedded Ethernet development boards which is produced by YDK Technologies Inc. that provides a hardware platform based on Altera® ACEX™ devices (see Figure 4-8). The Ethernet Development board includes hardware and software components that provide network connectivity for the embedded systems.

The components including:

- A network-interface card (NIC) that plugged directly into the development board.
- An SOPC Builder library component.
- A C language library that provides a Network-protocol stack.

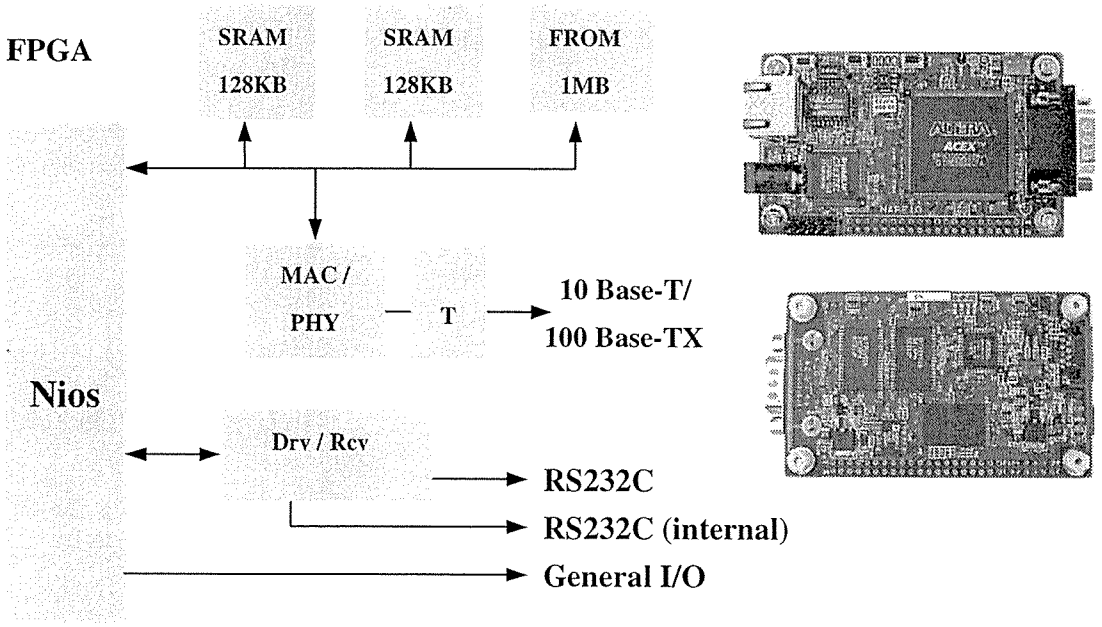


Figure 4-8. YDK Ethernet development board

The performance of the development boards is listed in table 4-1.

Ethernet development board is linked with PC via the Ethernet switch. For the Debug-Port Connector and Serial Interface Connector, used Debug-exclusive and Cross-link RS232C cable to link with COM1 and COM2 of PC. The board is working with the factory default settings.

In order to establish interaction and communication with the Ethernet development board, we downloaded the system design (RT layer) to Flash Memory on the board. We can also erase the user section of the flash memory device easily, and download the software again.

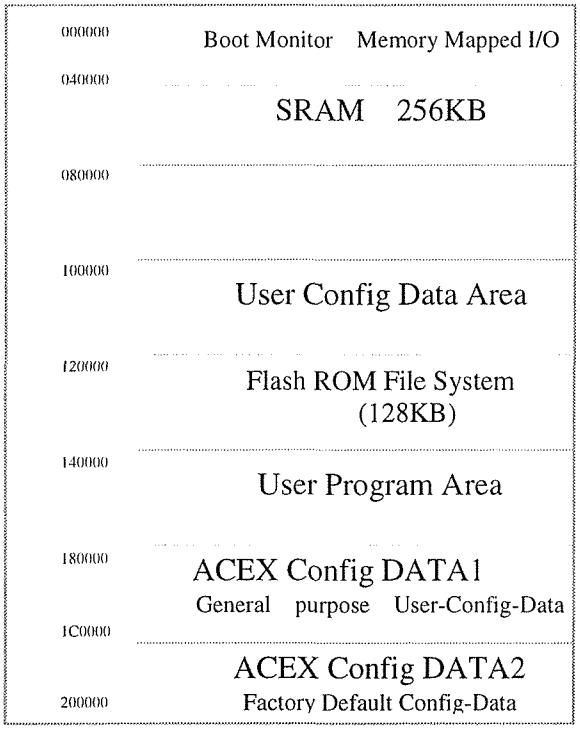
When download the software to the flash memory device, the system module pins are logically connected to pins on the ACEX device. The external physical pins on the ACEX device are in turn connected to other hardware components on the board, allowing the Nios embedded processor to interface with SRAM, FROM, LEDs, LCDs, switches, and buttons.

Table 4-1. Performance of Ethernet development board

Specifications of the board

Items	Contents
CPU	Nios (32bit RISC)
SRAM	256K byte
Flash ROM	1M byte
PLD	EP20K160EQC240
Ethernet	10 Base-T/100 Base-TX
RS232C	115.2K~ 9600bps
SIO (internal)	115.2Kbps
Extended connector	Common I/O (32bit)

Memory and I/O map



We use a 100 Mbit/s Ethernet switch with Ethernet frames that has the data field maximized (1538 bytes in IEEE 802.3 standard). Figure 4-9 illustrates the configuration of the LAN with a full-duplex switched Ethernet and end-nodes. We use the existing desktop computer, with AMD-K7(tm) Processor at 700MHz, 5-ports Ethernet switch with full-duplex links at 100Mbps. The size of data packet is from 64 bytes to 1538 bytes. There are no modifications in the Ethernet hardware on the NIC.

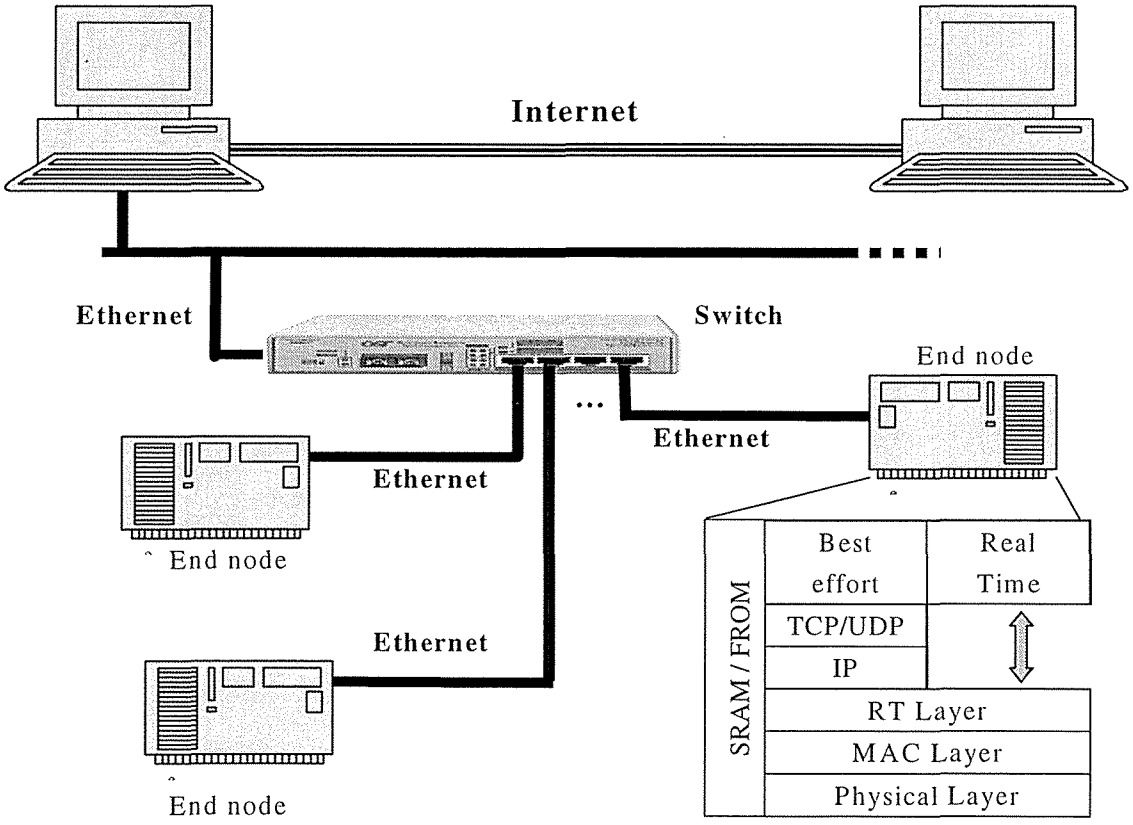


Figure 4-9. LAN with full-duplex switched Ethernet

4.6.2 Performance comparisons and analyzing

Below we discuss about the transmission latency of the real-time frame in worst-case situation. When all RT-channel starts simultaneously, or all the messages that use all the capability allowances of the RT channel, RT channel equipped with the longest deadline will be scheduled at last so that it may have the worst-case latency. Here for all RT channels, the maximum latency is characterized by:

$$T_{m\_lat} = \max_{i}\{T_{n1,i} + T_{n2,i} + T_t\}, \tag{5}$$

where $T_{n1,i}$ is the latency from source nodes to switch, $T_{n2,i}$ is the latency from switch to destination nodes, and T_t is the total delay inside the switch.

Besides utilization and worst-case latency, another important performance is a runtime overhead: R_i defined as:

$$R_i = \frac{T_{pd,i} - L_i \times 8 / B}{T_{pd,i}}, \tag{6}$$

where L_i is the length of data in a request frame, $L_i \times 8$ is the number of bits in the frame; $T_{pd,\,i}$ represents the period duration from the startup to the end of the frame, and B represents the Ethernet bandwidth.

Utilization, Data frame transmission latency and the frame runtime overhead can be obtained by implementing the proposed protocol to the LAN. Figure 4-10 illustrates the utilization on real-time data frame. The maximum size of data frame is 1538 bytes.

From the figure we can learn that the trend of utilization is increasing while the traffic increases, until arriving at the peak value that is more than 90%. Sudden decreases happen on the curve sometimes, which is caused by some short frames having pad field whose utilization are lower than longer frames. The utilization curve is always smooth, because we assume the sufficient resources (bandwidth and time specification) have been obtained in our work, when the RT channel is established. Under this circumstance, there should be no overload exists in the real-time channel. The result shows that the

deadlines have been met for all data because utilization of all data frames is less than 100% using EDF scheduling. Dynamic priority scheduling with the EDF algorithm has a distinct advantage over fixed priority scheduling: the schedulable bound for EDF is 100% for all task sets. This means that we can fully utilize the computing power of the CPU. Embedded systems are in general fully loaded, as they attempt to get as close to 100% utilization as possible while still maintaining the necessary predictability.

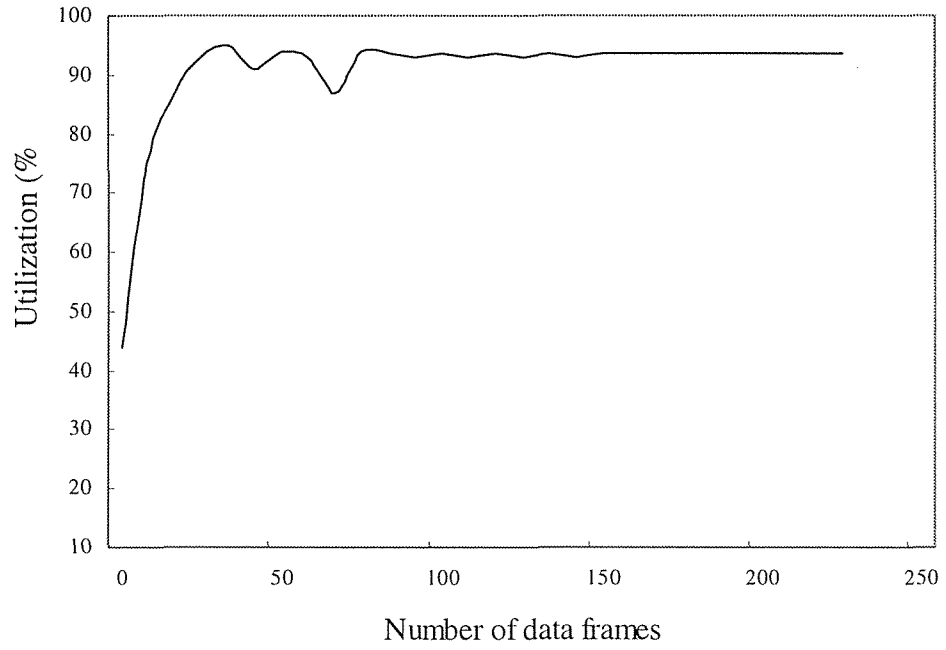


Figure 4-10. Utilization on real-time data frame

To the best of our knowledge, except a few implementations of hard real-time communication protocols on Ethernet, most of the protocols are generally soft real-time, which means that there are few protocols provides guarantees for both bit rate and strict delivery deadlines, so that it is difficult to compare them with the proposed hard real-time protocol. Therefore, we made performance comparison of the proposed

protocol only with the hard real-time communication protocols: MIL-STD-1553B protocol and RTCC protocol. Figure 4-11 shows the comparison of the data frame transmission latency of these three kinds of hard real-time communication protocols.

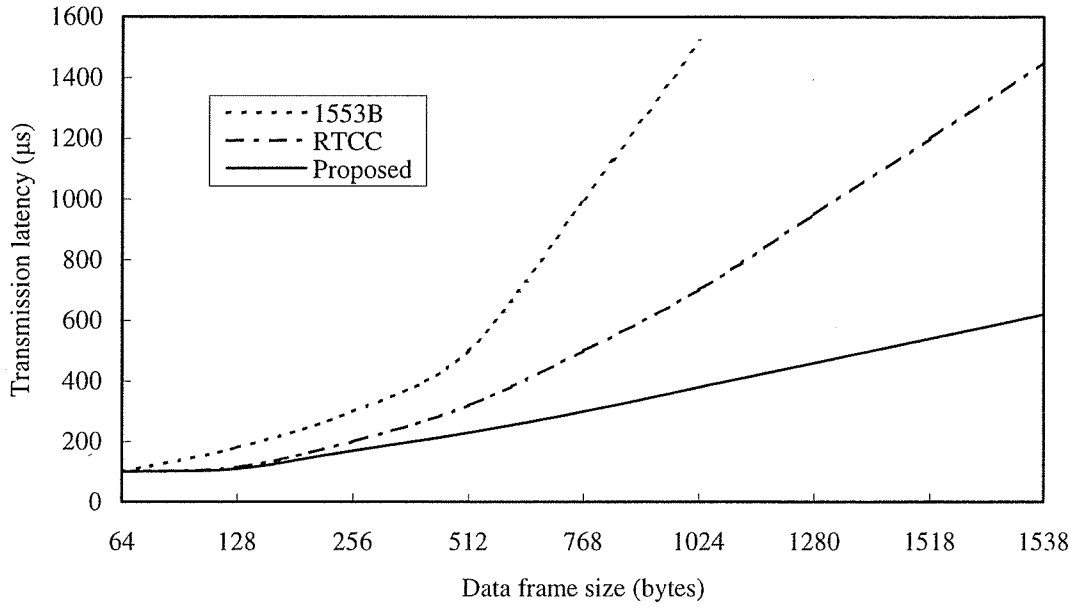


Figure 4-11. Transmission latency of data frame

Even for the Ethernet frames that have the data field maximized (1538 bytes in IEEE 802.3 standard), the latency of the proposed protocol is about 620 microseconds. This latency is quite short in a LAN with a full-duplex switched Ethernet at 100 Mbps, and should be meet the demands of hard real-time communication for industrial distributed control systems.

Runtime overhead of data frames are demonstrated in Figure 4-12. The figure shows that the runtime overhead of the proposed protocol is higher than the other hard real-time supported protocols at the small-sized data frame. However, as the data frame size become larger (from about 900 bytes), the proposed protocol has more better

runtime overhead than the other protocols.

In both experiments, only MIL-STD-1553B protocol used 1Mbps Ethernet because it is the nominal speed of the protocol.

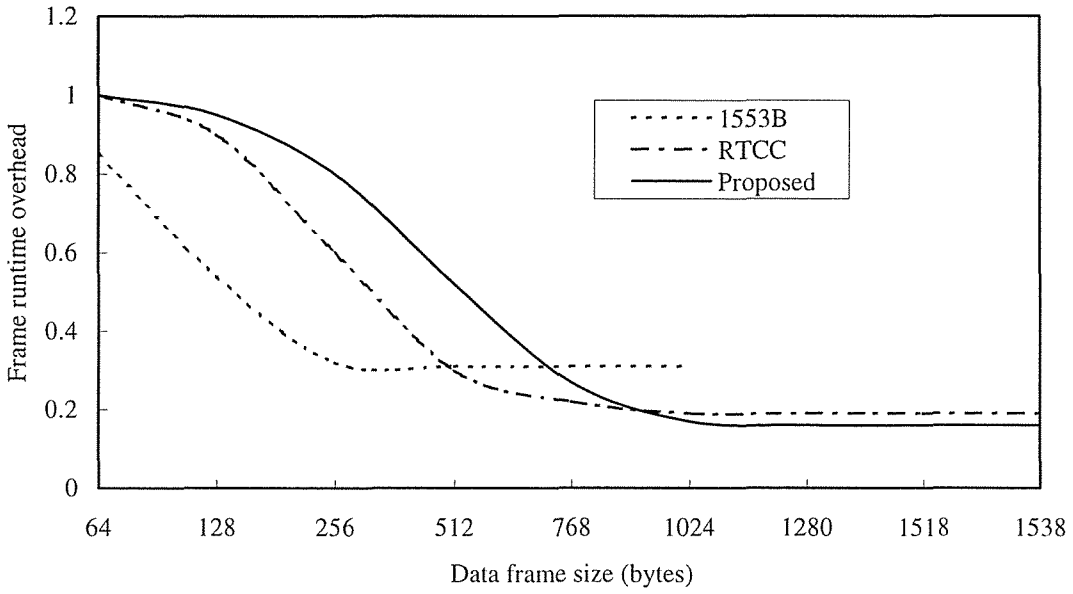


Figure 4-12. Runtime overhead of data frame

Furthermore, in order to evaluate the time effectiveness of hard real-time bypassing the TCP/IP stack, we made experiments with ordinary UDP/IP and TCP/IP protocols comparing with the proposed protocol (see Figure 4-13). The result shows that by using the proposed protocol to bypass the TCP/IP stacks can reduce 32% of the time comparing with UDP/IP protocol, and more than 50% of the time comparing with TCP/IP protocol even if the proposed protocol needs RT channel establishment and EDF scheduling.

From these figures it is noticed that the proposed protocol can get better real-time performance comparing with the other conventional hard real-time communication protocols and UDP/TCP/IP protocols.

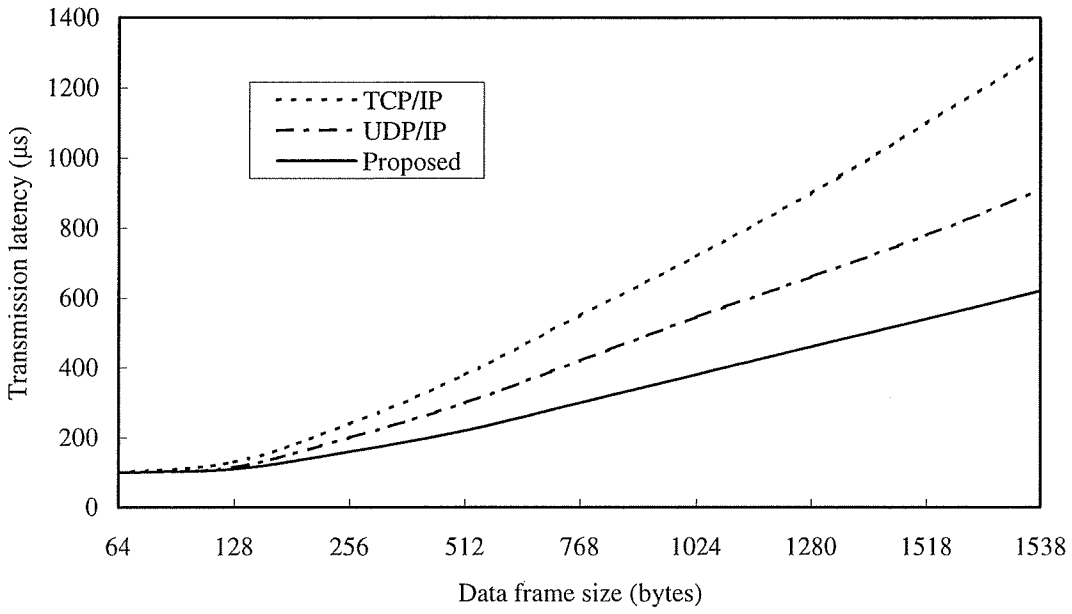


Figure 4-13. Transmission latency comparison

5. Conclusions

Ethernet is a dominating local area networking technology and is positioned to play a major role in future high-speed system area networks. The purpose of the work has been to explore switched Ethernet networks especially in the context of the hard real-time communication for industrial distributed control system network and make a model of the network components.

This chapter emphasizes the achieved objectives, main points of the work, and future research works.

5.1 Objectives achieved

In this paper, we have presented a protocol to support hard real-time communication over non real-time LAN technology, with a switched Ethernet based network concept. For the network configuration, based on establishment of the RT channel strategy, both switch and end-nodes have an RT layer added to support hard real-time traffic based on EDF algorithm.

Hard real-time traffic from the end-node bypasses the TCP/IP stacks and thus considerably speed up real-time communication. For the time effectiveness of hard real-time traffic bypassing the TCP/IP stack, experimental results shows that by using the proposed protocol can get much shorter transmission latency than using ordinary UDP/IP and TCP/IP protocols. The real-time traffic schedule is performed according to dynamic-priority EDF algorithm without modifications in the Ethernet hardware. Therefore, it is flexible and efficient.

For the process of best-effort (non or soft real-time) traffic, by carrying the final destination MAC address in the Ethernet header when leaving from the source node, it is delivered through the circuit including the TCP/IP stacks in an FCFS-sorted queue, and transmit the traffics at the idle time of the schedule.

In the proposed work, there are no modifications in the Ethernet hardware on the NIC. This allows connecting the Ethernet LAN to existing Internet networks.

We have constructed a simple Ethernet LAN with the proposed hard real-time communication protocol and made several experiments to evaluate the protocol. Through the comparison with some conventional hard real-time network protocols, we have shown that the proposed Ethernet protocol has better real-time performances, and meets the requirements of reliability for hard real-time systems. In addition to this, we made experiments with ordinary UDP/IP and TCP/IP protocols comparing with the proposed protocol, in order to evaluate the time effectiveness of hard real-time bypassing the TCP/IP stack. The result shows that by using the proposed protocol to bypass the TCP/IP stacks can reduce 32% of the time comparing with UDP/IP protocol, and more than 50% of the time comparing with TCP/IP protocol, even if the proposed protocol needs RT channel establishment and EDF scheduling.

To summarize the achieved objectives mentioned above, the presented switched Ethernet protocol can be adopted in an industrial hard real-time LAN applications with a limited number of end-nodes such as factory-floor embedded system communications, distributed industrial control system communications, parallel signal processing and robotics.

5.2 Important points

Several conclusions are made above, here is a summary of the important points:

1. ***Switched Ethernet networks can provide a reasonably good service for system area networks with real time requirements.***

Switches provide a flexible and scalable solution to the problems and limitations inherent to shared Ethernet networks (bus or hub-based), through the use of new mechanisms such as micro-segmentation, flow control and full-duplex operation. These

mechanisms may improve the timing determinism and performance of Ethernet to a great extent.

2. ***Proposed Ethernet protocol has better real-time performances.***

This discussion has two aspects: transmission latency and runtime overhead.

The latency of the proposed protocol is small comparing with the other hard real-time supported protocols, and comparing with the ordinary UDP/TCP/IP protocols. Even for the Ethernet frames that have the data field maximized (1538 bytes in IEEE 802.3 standard), the latency is about 620 microseconds. This latency is quite short in a LAN with a full-duplex switched Ethernet at 100 Mbps, and should be meet the demands of hard real-time communication for industrial distributed control systems.

Although the runtime overhead of the proposed protocol is higher than the other hard real-time supported protocols at the small-sized data frame, as the data frame size become larger (from about 900 bytes), the proposed protocol have more better runtime overhead than the other protocols.

3. ***Full utilization can be obtained by applying EDF algorithm in the proposed protocol.***

Embedded systems are in general fully loaded, as they attempt to get as close to 100% utilization as possible while still maintaining the necessary predictability. EDF is an optimal dynamic-priority scheduling algorithm in resource sufficient environments and the schedulable bound for EDF is 100% for all task sets.

4. ***Ethernet devices can also support TCP/IP stacks so that Ethernet can easily gate to the Internet.***

This feature is attractive to users since it allows remote diagnostics, control, and observation of their plant network from any Internet-connected device around the world with a license-free web browser.

5.3 Future work

The work of this thesis has been verified by experimental studies, on a theoretical level only. The complexity of industrial distributed and embedded hard real-time applications is growing in both size and diversity. For the proposed work to be useful for such applications, system configuration and architecture must be able to express the requirements of functionality as well as the possibilities and limitations of the run-time systems, as accurately as possible.

In order to validate the claims of the industrial application of this thesis, the presented hard real-time Ethernet protocol should be practically implemented in commercially available real, full-scale development projects.

Therefore, practical implementation of the proposed protocol in the factory-floor distributed hard real-time communication control systems is one of our main motivations.

Furthermore, today's switched Ethernet offers large bandwidths (Gbps) for factory communication, and the proposed Ethernet protocol should be suitable for them without modifications of the hardware and/or the software. Therefore, the proposed work should have much better performance if using the wider bandwidth Ethernet and high-performance switches. Also, if it is possible for implementing the proposed protocol in the larger networks by interconnecting several switches, there should be the large number of end-nodes added to the network.

Therefore, our further research subjects also including to consider the way of implementing and experiment by using the proposed protocol to the multi-layer, many switches real-time communication systems.

References

1. Husayin and Alimujiang Yiming, The communication between STD industrial control computer systems and PC, Journal of Electric Drive, ISSN1001-2095, Vol.25, No.5, p45-46, May 1995.
2. Husayin, Bin Yu and Alimujiang Yiming, Accomplishing the AC digital speed-controller using industrial control computer, Journal of Electric Drive, ISSN1001-2095, Vol.25, No.3, p41-46, March 1995.
3. Fieldbus Standard for Use in Industrial Control Systems, Part 2: Physical Layer Specifications and Service Definition, ISA S50.02-1992.
4. International Standard for Use in Industrial Control Systems, Part 2: Physical Layer Specifications and Service Definition, IEC 61158-2 (1993).
5. Foundation Fieldbus Specifications, Revision 1.3, Fieldbus Foundation, 1998.
6. Benefits Observed During Field Trials of an Interoperable Fieldbus, Kurt A. Zech, Fieldbus Foundation, Paper #94-504 – ISA, 1994.
7. Alimujiang Yiming, Applications of Fieldbus Control Technology in Intelligence Building, Journal of Xinjiang Institute of Technology, ISSN1008-9942, Vol.15, No.12, p8-11, December 2001.
8. C. Krishna and K.G. Shin, Real-Time Systems, McGraw-Hill International edition, (1997)
9. M. Joseph, Real-Time Systems, Prentice Hall, (1996)
10. Phillip A. Laptane, Real-Time System Design and Analysis, IEEE Press, third edition, (2004)
11. S. Baruah, A.K. Mok, and L.E. Rosier, Preemptively scheduling hard real-time sporadic tasks on one processor, December 1990.
12. A. Burns, N.C. Audsley, M.F. Richardson, and A.J.Wellings, Hard real-time scheduling: the deadline monotonic approach, Proceedings of the IFAC/IFIP Workshop, UK, 1992.

13. Luca Abeni and Giorgio C. Buttazzo, Integrating multimedia applications in hard real-time systems, In Proceedings of the 19th IEEE Real-Time Systems Symposium, RTSS'98, Madrid, Spain, December 1998.
14. G. Fohler, Flexibility in Statically Scheduled Hard Real-Time Systems, PhD thesis, Technische Universit¨at Wien, Austria, Apr. 1994.
15. G. Fohler. Joint scheduling of distributed complex periodic and hard aperiodic tasks in statically scheduled systems, In Proceedings of 16th Real-time Systems Symposium, Pisa, Italy, 1995.
16. G. Fohler and C. Koza. Heuristic scheduling for distributed real-time systems, Technical Report 6/98, Institut f¨ur Technische Informatik, Technische Universit¨at Wien, April 1989.
17. Gerhard Fohler, Dynamic timing constraints — relaxing over constraining specifications of real-time systems, In Proceedings of the IEEE Real-Time Systems Symposium – Work-in-Progress Session, December 1997.
18. D. Isovic and G. Fohler, Online handling of hard aperiodic tasks in time triggered systems, In Proceedings of the 11th Euromicro Conference on Real-Time Systems, Stockholm, Sweden, June 1999.
19. J. Stankovic and K. Ramamritham, Hard Real-Time Systems, IEEE Computer Society Press, (1988)
20. C. L. Liu and J. W. Layland, Scheduling algorithms for multi-programming in hard real-time traffic environment, Journal of the Association for Computing Machinery, 20-1, pp.46-61, (1973)
21. G. Buttazzo and J. Stankovic, Adding Robustness in Dynamic Preemptive Scheduling, Kluwer Academic Publishers, 1995.
22. Jan Carlson, Tomas Lennvall, and Gerhard Fohler, Enhancing time triggered scheduling with value based overload handling and task migration, In Proceedings of 6th IEEE International Symposium on Object-oriented Real-time distributed

Computing ISORC'2003, May 2003.

23. H. Chetto, M. Silly, and T. Bouchentouf, Dynamic scheduling of real-time tasks under precedence constraints, Real-Time Systems Journal, 2(3):181–194, Sept. 1990.
24. Jeffay, K., and Stone, D. L. 1993, Accounting for interrupt handling costs in dynamic priority task systems, in Proceedings of the 14th Real-Time Systems Symposium, pp. 212–221.
25. H. Kopetz. Sparse time versus dense time in distributed real time systems, Proceedings of the Second International. Workshop on Responsice Computer Systems, Saitama, Japan, Oct. 1992.
26. G. Buttazzo. Hard Real-time Computing Systems, Predictable Scheduling Algorithms and Applications - Real-Time Systems Series 2ND Edition, Springer Verlag, (2004)
27. Liu, J. Real-time Systems, Prentice Hall Inc., 2000; ISBN: 0130996513.
28. Crockett, N. Industrial Ethernet is ready to revolutionize the factory — Connecting the factory floor, Manufacturing Engineer, Volume: 82, Issue: 3, June 2003, pages 41–44.
29. ISO/IEC 15802-3: 1998, ANSI/IEEE Std. 802.1D, 1998 Edition, Information technology — telecommunications and information exchange between systems — local and metropolitan area networks — common specifications, Part 3: Media Access Control (MAC) bridges.
30. H. Kopetz, REAL-TIME SYSTEMS --- Design Principles for Distributed Embedded Applications, Kluwer Academic Publishers, (1997)
31. C. Venkatramani and T. Chiueh, Supporting real-time traffic on the Ethernet, in Proc. IEEE Real-Time Systems Symposium, San Juan, Puerto Rico, December 1994, pp. 282-286.
32. H. Hoang, M. Jonsson, U. Hagström, and A. Kallerdahl, Switched real-time

Ethernet with earliest deadline first scheduling - protocols and traffic handling, in Proc. Workshop on Parallel and Distributed Real-Time Systems, Fort Lauderdale, FL, April 2002.

33. Jan Axelson, Embedded Ethernet and Internet Complete, Lakeview Research, September 2003
34. A. Leon-Garcia and I. Widjaja, Communication Networks - Fundamental Concepts and Key Architectures, McGraw-Hill Osborne, (2001)
35. M. Schwartz & T. E. Stern, Routing Protocols, in Computer Network Architectures and Protocols (Second Ed.), Ed. C.A. Sunshine, Plenum Press, New York / London (1989)
36. IEEE Std. 802.3, 2000 Edition, IEEE Standard for Information technology -- Telecommunications and information exchange between systems -- Local and metropolitan area networks -- Common specifications -- Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.
37. SIXNET, The sixnet industrial ethernet switch, http://www.sixnetio.com/.
38. J.-F. Hermant and G. Le Lann, A protocol and correctness proofs for real-time high performance broadcast networks, in Proceedings of the 18th IEEE International Conference on Distributed Computing Systems, 1998, pp. 360–369.
39. S. Varadarajan and T. Chiueh, Ethereal: A host-transparent real-time fast Ethernet switch, in Proceedings of IEEE International Conference on Network Protocols, October 1998
40. IEEE 802.1Q, 1998 IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridge Local Area Networks.
41. R. Seifert, Issues in LAN Switching and Migration from a Shared LAN Environment, Technical Report, Networks and Communications Consulting, November 1995.

42. R. Seifert, Issues in LAN Switching and Migration from a Shared LAN Environment, Technical Report, Networks and Communications Consulting, November 1995.
43. Lantronix, Ethernet Tutorial: Network Switching, Technology Tutorials, http://www.lantronix.com/technology/tutorials/switching.html, 1999.
44. Intel Adaptive Technology – Optimizing Network Performance, http://www.intel..com/network
45. Georges, J.-P.; Rondeau, E.; Divoux, T, Evaluation of switched Ethernet in an industrial context by using the Network Calculus, Factory Communication Systems, 2002. 4th IEEE International Workshop on 28–30 Aug. 2002, pages 19–26.
46. Rockwell Int. Corp., Ethernet for Industrial Control, An Ethernet White Paper, Rockwell International Corporation, April 21, 1998.
47. (ISO/IEC) ANSI/IEEE Std 802.1D, 1998 Edition, Information technology --Telecommunications and information exchange between systems -- Local and metropolitan area networks - Common Specification - Media access control (MAC) bridges.
48. http://www.condoreng.com/support/downloads/tutorials/ MIL-STD-1553Tutorial.PDF
49. Z. P. Wang, G. Z. Xiong, J. Luo, M. Z. Lai and W. Zhou: A hard real-time communication control protocol based on the Ethernet, Proceedings of 7th Australasian Conference on Parallel and Real-Time Systems (PART 2000), pp.161-170, (2000)
50. J. Loeser and H. Haertig, Low-latency hard real-time communication over switched Ethernet, Proceedings of 16th Euromicro Conference on Real-Time Systems (ECRTS'04), pp. 13-22, (2004)
51. S. Ouni and F. Kamoun, Hard and soft real time message scheduling on Ethernet networks, Proceedings of the 2nd IEEE International Conference on Systems, Man

and Cybernetics of the twenty-first century, 6, (2002)

52. H. Zhang, Service disciplines for guaranteed performance service in packed switching network, Proc. of the IEEE, vol.83, no.10, (1995)
53. D. Ferrari and D. Verma, A scheme for real-time channel establishment in wade-area networks, IEEE Jornal of Selected Areas in Communications, vol.8, no.3, pp.368-379, (1990)
54. L. Zhang, Designing a new architecture for packet switching communication networks, IEEE Communications Magazine, vol. 25, n. 9, pp. 5-12, (1987).
55. G. Agrawal, B. Chen, W. Zhao, and S. Davari, Guaranteeing Synchronous Message Deadlines with Time Token Medium Access Control Protocol, IEEE Transactions on Computers, Vol. 43, No. 3, pp 327-339, (1994)
56. S. Chakrabarti and A. Mishra, QoS issues in ad hoc wireless networks, IEEE Communications Magazine, vol. 39, no. 2, pp. 142-148, February 2001.
57. J. M. Wozencraft and M. Horstein, Digitalised communication over two-way channels, in Proc. Fourth London Symposium on Information Theory, London, U.K., September 1960.
58. Reinder J. Bril, Maria Gabrani, Christian Hentschel, G. C. van Loo, and E. F. M. Steffens, Qos for consumer terminals and its support for product families, In Proceedings of the International Conference on Media Futures, Florence, Italy, May 2001.
59. G.C. Buttazzo, G. Lipari, and L. Abeni, A bandwidth reservation algorithm for multi-application systems, Proceedings of the International Conference on Real-time Computing Systems and Applications, Japan, 1998.

Acknowledgement

This work is the result of three years research and study as a Ph.D. student in the control system laboratory of Kitami Institute of Technology. I have been very fortunate to be accepted as a Ph.D. student in this lab and do my research under the supervision of excellent professors, professor Toshio Eisaka, professor Masakyo Suzuki and Professor Yuji Kamiya.

I am very grateful to my professor Toshio Eisaka who has supported and collaborated in all of my research activities. His insight, suggestion and unending patience contributed immensely to the fulfillment of this thesis.

I would like to thank professor Masakyo Suzuki, for his advice on improving this work. I also thank to assistant, Mr. Akira Kikuta, technical staff, Mr. Shukuin and my fellows in the control system lab for their help during my study and research.

I should thank to professor Junichi Kamemaru, to the staff of Kitami Institute of Technology, especially in student affairs and international center, for their kind manner and help during my stay here in KIT.

I was also very fortunate to get the Yoneyama scholarship of international Rotary club in Japan. It was the main financial support during my study and research in KIT.

Lastly, I address my special thanks to my family, my wife Rizwan and my daughter Aymire, for their abundant love, patience, warm encouragement and emotional support.

Thank you all.

Study History

Journal papers:

Authors (all), title, Journal, Vol., No., pp. - Month, Year

1. Alimujiang Yiming and Toshio Eisaka, Hard Real-Time Communication over Switched Ethernet, International Journal of High Performance Computing and Networking (IJHPCN), to appear.

2. Alimujiang Yiming and Toshio Eisaka, A Switched Ethernet Protocol for Hard Real-Time Communication, Journal of the Society of Instrument and Control Engineers (the SICE), submitting.

3. Alimujiang Yiming and Toshio Eisaka, A switched Ethernet protocol for hard Real-Time embedded system applications, special issue of the Journal of Interconnection Networks (JOIN), September 2005.

4. Alimujiang Yiming, Applications of Fieldbus Control Technology in Intelligence Building, Journal of Xinjiang Institute of Technology, ISSN1008-9942, Vol.15, No.12, p8-11, December 2001.

5. Alimujiang Yiming, Dynamic neuro-fuzzy control of the nonlinear process, Journal of Xinjiang Institute of Technology, ISSN1008-9942, Vol.15, No.8, p13-16, August 2001.

6. Alimujiang Yiming, Solving the N-bit parity problem using neural networks, Journal of Xinjiang Institute of Technology, ISSN1008-9942, Vol.15, No.2, p21-24, February 2001.

7. Alimujiang Yiming, Analyzing of functions of the optimal controller, Journal of Xinjiang Institute of Technology, ISSN1008-9942, Vol.14, No.2, p31-35, February 2000.

8. Abdurexit and Alimujiang Yiming, Application of neural intelligence PID control algorithm, Journal of Xinjiang Institute of Technology, ISSN1008-9942, Vol.14, No.2, p1-4, February, 2000.

9. Alimujiang Yiming, Applications of Artificial neural networks in industrial control, Journal of Xinjiang Institute of Technology, ISSN1008-9942, Vol.14, No.1, p29-34, January 2000.

10. Husayin and Alimujiang Yiming, The communication between STD industrial control computer systems and PC, Journal of Electric Drive, ISSN1001-2095, Vol.25, No.5, p45-46, May 1995.

11. Husayin, Bin Yu and Alimujiang Yiming, Accomplishing the AC digital speed-controller using industrial control computer, Journal of Electric Drive, ISSN1001-2095, Vol.25, No.3, p41-46, March 1995.

International Conferences:

1. Alimujiang Yiming and Toshio Eisaka, Industrial Hard Real-Time Traffic Protocol based on Switched Ethernet, International Symposium on Communications and Information Technologies 2005 (ISCIT 2005), October 12-14, 2005, Fragrant Hill Hotel, Beijing, China.

2. Alimujiang Yiming and Toshio Eisaka, A switched Ethernet protocol for hard Real-Time embedded system applications, The IEEE 19th International Conference on Advanced Information Networking and Applications (AINA-2005), March 28 -

March 30, 2005, Tamkang university, Taiwan, Republic of China, ISBN: 0-7695-2249-1, vol.2, p41-44.

3. Alimujiang Yiming and Toshio Eisaka, Development of an Ethernet protocol for industrial real-time communications, International Symposium on Communications and Information Technologies 2004 (ISCIT 2004), October 26-29, 2004, Sapporo Convention Center, Sapporo, Japan, ISBN: 0-7803-8594-2, p1122-1125.

4. Alimujiang Yiming and Toshio Eisaka, A Multi-point Ethernet Communication Protocol in Real-Time Embedded applications, International Workshop on Modern Science and Technology 2004 (IWMST 2004), September 2-3, 2004, Kitami Institute of Technology, Kitami, Japan, p55-58

5. Alimujiang Yiming and Toshio Eisaka, An Ethernet protocol for real-time communications, Proceedings of the Society of Instrument and Control Engineers (SICE) 43rd Annual Conference in Sapporo, August 4-6, 2004, Hokkaido Institute of Technology, Japan, p1905-1908

Local Conferences:

1. Alimujiang Yiming and Toshio Eisaka, A switched Ethernet protocol for hard real-time communications, SICE Hokkaido Branch 37th Annual Conference, January 27-28, 2005, Hokkaido university, Sapporo, Japan, p83-86

2. Alimujiang Yiming and Toshio Eisaka, Communications between the embedded system devices and PC using Nios EDK software, SICE Hokkaido Branch 36th Annual Conference, January 22-23, 2004, Sapporo Convention Center, Sapporo, Japan, p1-2