IST-2001-37652

Hard Real-Time CORBA

Title	**D2.1 RT-Protocols for Real-Time Control** **Overview**
Authors	Thomas Losert (TUWien)
Reference	IST37652/036
Date	2002-12-20
Release	1.0
Status	Final
Clearance	Consortium
Partners	*Universidad Politécnica de Madrid* *Lunds Tekniska Högskola* *Technische Universität Wien* *SCILabs Ingenieros*

Summary Sheet

IST Project 2001-37652
HRTC
Hard Real-time CORBA

D2.1

RT-Protocols for Real-Time Control

Overview

Abstract:

The present document contains an overview of the available network protocols and investigates their suitability for hard real-time communication.

Copyright:

HRTC Partners:

Universidad Politécnica de Madrid
Lunds Tekniska Högskola
Technische Universität Wien
SCILabs Ingenieros.

Release Sheet (1)

Release:	0.1 Draft
Date:	2002-10-28
Scope	Initial version
Sheets	All

Release:	0.2 Draft
Date:	2002-11-28
Scope	Revised version
Sheets	All

Release:	1.0 Final
Date:	2002-12-20
Scope	Improved chapter 2 and 3
Sheets	All

Table of Contents

1 Introduction

One problem of CORBA is that – like in other models of computation – the necessary time for executing a method is not regarded. Thus the OMG interface definition language (IDL) lacks the capability of modeling the concept of physical time in interfaces.

Physical time is needed if we are to reason about timely failure detection (in particular, of autonomous component systems), performance, and other real-time properties. This point of view is also taken by Edward A. Lee in an excellent recent survey on embedded computer systems: *"Time has been systematically removed from theories of computation, since it is an annoying property that computations take time. 'Pure' computation does not take time, and has nothing to do with time. It is hard to overemphasize how deeply rooted this is in our culture. So called "real-time" operating systems have so little to go on that they often reduce the characterization of a component (a process) to a single number, its priority."* [Lee99].

According to page 8 of the Technical Annex it belongs to the goals of this project to "identify hard real-time requirements for distributed control systems" and to "implement a CORBA pluggable protocol over a hard real-time transport". Since hard real-time can not be established on top of a communication layer with unknown temporal behavior this document identifies the necessary properties of the transportation layer and evaluates the available protocols regarding their real-time capabilities. Further some possibilities of making available the special features of hard real-time protocols are outlined ordered by the intrusiveness to existing implementations or the current specification (CORBA 3).

The remainder of this document is organized as follows: Section 2 identifies the requirements that characterize protocols especially suited for our purpose. Section 3 gives an overview how real-time protocols can be included in nowadays CORBA implementations and if this is enough for making the real-time properties available. Section 4 gives an overview of the available protocols with emphasize on the properties that have been identified in the previous sections. Section 5 compares the properties according to the results of the previous section. Section 6 concludes this document with a recommendation for a protocol that serves best our purposes.

2 Requirements

2.1 Notion of Component

According to [Kru98] a component is a non-trivial, nearly independent, and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces.

A component is substitutable for any other component which realizes the same interfaces. Logical and physical cohesiveness of a component denotes a meaningful structural and/or behavioral part of a larger system. Furthermore a component represents a fundamental building block upon which systems can be designed and composed. Conformity of a component to a given interface means that it satisfies the "contract" specified by that interface and may be substituted in any context wherein that interface applies.

An **interface** is referred to as a common boundary between two subsystems. Since architecture design is primarily interface design, the most important phase in the design of large system architecture is the layout and placement of the interfaces ([Kop97], p. 77). The interface of a component is the point of interaction between a system and its environment (see [JKK+01]). A correctly designed interface provides understandable abstractions, which capture the essential properties of the interfacing subsystems and hide irrelevant details (control, temporal, functional, and data properties).

In order to disentangle unrelated functions of components it is advantageous to specify a distinct interface for every separable service [Kop00]. We have identified three unique functions that occur in many scenarios and should normally be serviced across independent interfaces: The Real-Time Service (RS) Interface, the Diagnostic and Management (DM) Interface, and the Configuration Planning (CP) Interface. Besides from these three interfaces that expose well defined services to other components there is a Local Interface that supports services depending from the particular application (e.g. the communication with a particular temperature sensor). This allows e.g. a component that encapsulates the services of the local interface and provides these services in a generalized way to other components.
The following sections further describe the special properties of these interfaces:

2.1.1 Real-Time Service (RS) Interface

This is the interface that provides the intended service in a temporal predictable way to the environment, namely the systems with which it interacts. The real-time service interface is the most important interface for the user of the service. To keep the service interface small and understandable, only those objects and functions that are required for the intended emerging service should be visible at the service interface. It is counterproductive for all internal objects of a component system to be visible at the service interface.
In the CORBA world [Sie00], the (syntax of the) services that are provided by an object are defined by the interface definition in a special interface definition language (IDL) that can be mapped into a number of different programming languages. The interface definition specifies the operations that can be performed by the object, the input and output parameters, possible exceptions that may by raised by the object during execution, and possibly, the declared state of the component.
In real-time systems, the purpose of the RS interface is the timely exchange of observations among the component subsystems. An observation states that the state variable possessed the stated value at the indicated instant or an event occurred at the instant. In control applications, the temporal access pattern of information at the RS interface is typically periodic, and a small delay and minimal jitter are important for the quality of control. These temporal parameters must be stable in order to support the composability at the RS interface. The user of the observations at the RS interface must know only about the meaning of these observations but does not need any knowledge about the internal structure or operation of the component system that delivers the observation.

2.1.2 Diagnostic and Management (DM) Interface

The DM interface provides a communication channel to the internals of the component system for the purpose of diagnosis and management.
A maintenance engineer who accesses the internals of a component system via the DM interface must have detailed knowledge about the internal structure, the internal objects

and the precise behavior of the system. The end-points of communication are the internals of a component system on one side and some maintenance system or engineer, possibly sitting at a remote terminal on the Internet, on the other side. The communication pattern is, thus, point-to-point and the messages between the maintained component system and the maintenance system or engineer must be routed transparently through a set of networks. The DM interface should be independent from the service interface, since these two interfaces are directed towards two different user groups and require different knowledge.
In a real-time system, there is usually a need to support on-line maintenance and management while a system is operational. To achieve this objective, any sporadic maintenance and management traffic must coexist with the time-critical real-time traffic without disturbing the latter. The traffic pattern across the DM interface is normally sporadic and not time-critical, although precise knowledge about the instant when a particular value was observed or modified can be important.

2.1.3 Configuration Planning (CP) Interface

The CP interface is used during the integration or reconfiguration phase to connect a component system to other component systems of a system of systems.
The CP interface is typically point-to-point and not time-critical.

2.1.4 Local Interfaces

This interface realizes the connection between the component and some sensors or actuators located nearby the component. This interface depends from the particular application.

2.2 Communication Requirements for Transport

The interfaces denoted in the previous section are characterized by different requirements regarding the transport layer:

2.2.1 Real-Time Service (RS) Interface

The RS Interface requires the communication of real-time data with known latency and bounded jitter. The jitter can be translated in a measurement error if the gradient of the observed variable is known. Thus it should be reasonable low.
Since the data is transferred in a periodic schema this interfaces requires an a priori known bandwidth on the communication channel. Because of the nature of state information where the new state obsoletes the previous state the loss of individual data frames may be tolerated without degradation of service. For use in high risk environments an architecture is required that is able to tolerate a single failure in an arbitrary component in order to guarantee that no information is lost.

2.2.2 Event-Service (ES) Interface

Since the CP and the DM interface require similar temporal properties, they can be merged on the transport layer to an Event-Service (ES) Interface. Nevertheless for the application these interfaces should be available separately.
The CP Interface is mainly used in the initialization- and shutdown-phase and for switching to another configuration. Although it is not time critical for a seamless transition it is very important that the performance of the RS interface is not affected.

The DM Interface is used for diagnostic purposes in order to check the internal state of the node if the system does not operate in the intended way or for setting parameters (e.g. calibration values). This interface allows full access to the internals of the node and usually is not time critical.

2.2.3 Local Interfaces

Since this interface is used for realizing the connection between the component and some sensors or actuators located nearby the component, using CORBA is not viable for this interface. Thus the requirements of this interface are disregarded in the remainder of this document.

2.3 Classification of Requirements

A candidate protocol must provide possibilities of integrating real-time traffic with its special demands together with asynchronous non real-time communication without disturbing the real-time communication. This requires guaranteed bandwidth with known latency and minimal jitter to serve the purposes of the RS Interface. In addition it must allow interleaving real-time traffic with non real-time traffic (e.g. asynchronous requests for calibration).

In high risk environments highly dependable systems are required that must be able to tolerate the presence of a single failed component. Further, the protocol must allow a global notion of time and composability which means that the real-time properties of a system are preserved even in the case of adding other components.

To be able to run some demanding tasks the protocol must be available on a platform with sufficient CPU and memory resources. With respect to RT-CORBA it should be available on a widely available platform.

3 Including RT-Protocols in CORBA

Since the HRTC project targets especially on identifying approaches of creating CORBA based hard real-time applications this section identifies two approaches for adding hard real-time protocols to nowadays CORBA by using the current specification (CORBA 3).

3.1 Replacing the Protocol-Stack

Since the IIOP-protocol is mandatory for a CORBA-ORB communication between two ORBs is usually performed as an IIOP communication on top of the TCP/IP protocol which is often layered on top of the Ethernet protocol (see Figure 1: Commonly used Protocol-Layers of CORBA).

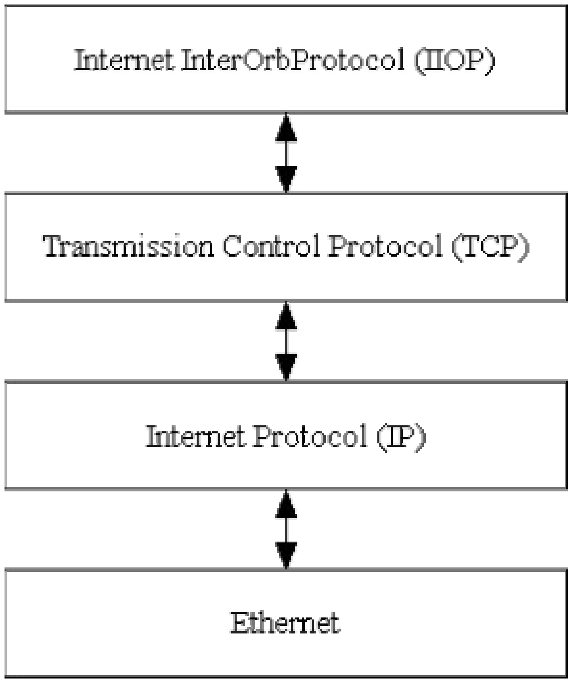
Internet InterOrbProtocol (IIOP)
Transmission Control Protocol (TCP)
Internet Protocol (IP)
Ethernet

Figure 1: Commonly used Protocol-Layers of CORBA

By replacing some of these protocols from bottom to top by a protocol offering the same services with specified temporal properties it is possible to enhance the temporal behavior of CORBA:

Ethernet: The duration for the transmission of a message is highly dependent from the current load of the system and is not limited if other nodes use the network continuously. Replacing the Ethernet Layer with a packet service that provides guaranteed bandwidth and constant latencies the communication of two ORBs becomes independent from the communication of other nodes.

IP: The services of IP are Fragmentation of packets and Routing through the network. If real-time communication takes place in one single cluster only, routing is not needed anymore which simplifies a replacement IP-layer.

TCP: One source of indeterminism of the TCP layer is based on the retry mechanism for lost packages. If the underlying packet service is based on a highly dependable communication channel, the retry mechanism will not be used and this source of indeterminism is eliminated. If the packets are provided in the proper order this also eliminates a source of indeterminism.

This approach is compatible with every ORB since the IIOP protocol is mandatory for all CORBA 3.0 compliant implementations. It can be performed with no changes of existing implementations.

This approach is least intrusive since the existing CORBA specifications are not touched. This approach allows RT-communication and

3.2 Extensible Transport Framework

Some members of the OMG have recognized the importance of replacing the TCP/IP protocol with other protocols. Thus in 2000 an RFP (see [OMG00]) has been issued.

In response to this RFP a proposal for an Open Communications Interface (OCI) has been submitted (see [OMG01]). Since the standardization progressed (and is still not finished yet) today's most recent interim document is [OMG02a].

Although these submissions differ in details (e.g. the names of the interfaces) they both describe the same core concepts and allow replacing the transport layer by the use of plugins with a specified interface:
Information is stored in buffers as a sequence of octets. The ORB of the client requests a connection from the server which allows reading and writing data stored in a buffer. This connection is considered as stream of octets, that guarantees that data is received in the same order as it is sent and no information gets lost. Thus the plugin has to include mechanisms for retrieving lost packages for an unreliable, connectionless transport. If a connection is not used anymore there are methods for closing the connection.

3.3 Evaluation

Neither of these possibilities allows solving all problems. By providing dedicated virtual connections from one RT-ORB to the other these approaches allow communication of other nodes without disturbing the behavior of the RT-ORBs.
As soon as a RT-ORB is involved in communication with a non real-time ORB it is possible that the RT-ORB (or the task running on this machine) misses its deadline because it has been interrupted by the non real-time ORB. Further, both approaches lack mechanisms to adjust to deadlines or provide information how long the communication of the data will need or about the delay because of well filled buffers.
For establishing a hard real-time system the following points should be considered:

? Reliable communication channel with bounded jitter and bounded transmission delay that is accessed in an a priori known pattern.
? The worst case execution time (WCET) of the ORB has to be adjusted to this pattern.
? The worst case execution time (WCET) of all tasks has to be adjusted to this pattern.

Introducing these properties and the necessary hooks for configuring the transport requires changes in several parts of the CORBA specification.

4 Overview of Protocols

Since there are a lot of protocols available the following comparison cannot be exhausting but it emphasizes on the protocols that are widely used or provide special features that qualify them for this purpose.

Some protocols (like TCP/IP) are layered on top of other protocols. Since the behavior is heavily influenced by this protocol, statements about this protocol's behavior are based on assumptions of the underlying protocol as mentioned in the description.

Each Protocol is described according to a standard structure. It is judged how the protocol meets the requirements established in the previous section.

4.1 Ethernet

The DIX Ethernet (the abbreviation DIX is for the developing companies: Digital, Intel, Xerox) is based on a CSMA/CD project of Xerox that started in 1973.

IEEE 802.3 (see also [IEEE802.3]) is a standard of the Institute of Electrical and Electronics Engineers (IEEE) for a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Local Area Network (LAN) specifying several data rates from 1 MBit to 1 GBit. It supports the bus topology as well as the star topology. The first version of this standard has been published in 1983 and is based on the results of DIX Ethernet.

An IEEE 802.3 Frame consists of a header (30 octets) and a payload (42-1500 octets) resulting in a minimum packet length of 72 octets.

In half duplex mode (which is especially used for bus topology), a station waits before transmitting for a quiet period on the medium (that is, no other station is transmitting) and then sends the intended message in bit-serial form. If, after initiating a transmission, the message collides with that of another station, then each transmitting station intentionally transmits for an additional predefined period to ensure propagation of the collision throughout the system. Since each node must be able to detect the collision the length of cabling in bus topology is limited (depending on the data rate) and a minimum packet length is required.

The station remains silent for a random amount of time (backoff) before attempting to transmit again. Each aspect of this access method process is specified in detail in subsequent clauses of this standard.

Without any further assumptions this standard is not usable for real time applications. It is possible that the senders A and B send packages to the receiver C at a higher combined transmitting bandwidth than C has as receiving bandwidth. Either the central switch drops some packages or it is not possible to calculate an upper bound for the latency.

Predictability can be established by using a bus access strategy that prevents collisions and limiting the bandwidth for each node. The following is an example for a real-time system based on Ethernet:

In the MARS approach (see [RSG89]) each node runs a set of statically scheduled real-time tasks and an identical copy of the MARS operating system. All operating system concepts that could lead to an unexpected delay or deadlock (e.g. dynamic resource allocation) have been kept out. Communication is performed by the broadcast of messages containing state variables on Ethernet. The bus is accessed in a TDMA scheme in order to prevent collisions. They have been able to establish a hard real-time system with a global time base, allowing a known synchronization accuracy of a few microseconds.

Another example of preventing collisions is subdividing the collision domain of an Ethernet network into several smaller collision domains by using an Ethernet Switch. Although in switched Ethernet all nodes remain in the same broadcast domain a node that is directly connected to a port of the switch is a collision domain of its own and therefore might utilize all available bandwidth for communication with the target node without disturbing communication between other nodes connected to the switch. Determinism can be reached if priorities are used (see [HS02]).

4.2 Wireless

According to [IEEE802.11] Wireless LAN (WLAN) is specified for two fundamentally different types of physical layers. The communication via diffuse infrared light is specified but only used rarely. Further, the communication via radio waves is specified. This medium is used by most network devices according to this standard.

Like Ethernet without any further assumptions this standard is not usable for real time applications.

By using an appropriate protocol on top of WLAN (e.g. as described in [NMG01]) it is possible to use it for real-time applications but in addition to the problems of Ethernet (see previous section) data loss can not be neglected in wireless communication which makes it only usable for real-time applications where sporadic data-loss can be tolerated.

In [LO01] is a discussion of other wireless communication mechanisms and a brief overview of their real-time capabilities. In order to prevent problems because of sporadic data loss and thus reduce complexity in debugging a wire bound communication mechanism should be preferred.

4.3 Internet Protocol (IP)

Networking in UNIX Systems is based on the Internet Protocol (IP) which has been developed in 1981 by the Information Sciences Institute (ISI) for the Defense Advanced Research Projects Agency (DARPA) which is a part of the Department of Defense (DoD) (see also [RFC791]).

The IP interfaces on one side to higher level protocols like TCP or UDP (see below) and on the other side to a local network protocol like ethernet (see Figure 2: Internet Protocol (IP) in relation to other protocols).

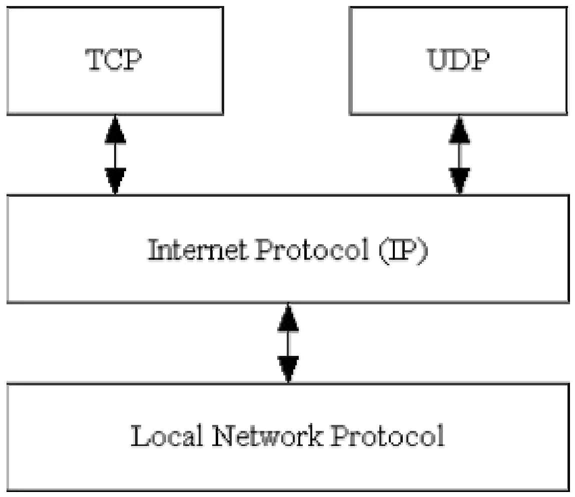

Figure 2: Internet Protocol (IP) in relation to other protocols

The IP serves mainly two purposes:

? Routing: Datagrams are transported from the source to the sink of the communication even if there is no direct path and thus gateways have to be used.

? Fragmentation: If an intermediate network supports small packet sizes only the datagrams have to be fragmented in smaller packages and reassembled at the receivers side.

Data is communicated through datagrams consisting of the IP header and the payload. The header length is at least 20 Bytes. For using fragmentation a packet must allow a payload of at least 8 octets thus resulting in a minimum packet size of 28 Bytes. If

TCP/IP is used with a packet size of just 28 Bytes even the header of the overlying protocol is likely to be fragmented which could be a reason for problems with some protocol implementations. According to [Cox96] IP requires at least 72 Bytes but for acceptable performance at least 200 Bytes should be provided.

The temporal behavior of IP depends on the temporal properties of the Local Network Protocol and on the load of the queues in the sender, the receiver, and intermediate nodes.

4.4 User Datagram Protocol (UDP/IP)

The User Datagram Protocol (UDP) assumes that IP is used as underlying protocol hence the name UDP/IP (see also [RFC768]).

It allows multiple processes to access the network independently by introducing the concept of ports. Each UDP Packet contains the source port and the destination port and the operating system provides a multiplexer and demultiplexer that assigns the datagram to the application owning the respective port.

In addition a 16 bit checksum is provided in order to protect the datagram against bit errors.

The UDP header adds to the IP header 8 bytes of header.

4.5 Transmission Control Protocol (TCP/IP)

The Transmission Control Protocol (TCP) assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols. The TCP fits into a layered protocol architecture just above the IP which provides a way for the TCP to send and receive variable-length segments of information enclosed in internet datagram (see also [RFC793]).

TCP is a connection oriented protocol introducing the following properties:

? Order: A stream of data is decomposed to packages. At the receivers side order has to be re-established thus composing the stream of data.

? Reliability: TCP tolerates errors on the underlying datagram service (damaged, duplicate or lost packages) by using a Positive Acknowledgement or Retransmission (PAR) protocol with sequence numbers.

? Flow Control: TCP communicates information about the number of packages the receiver may receive.

? Multiplexing: TCP allows multiple processes to access the network independently by introducing the concept of ports. Each TCP Packet contains the source port and the destination port thus allowing each packet to be assigned to the application owning the respective port.

? Connections: TCP allows establishing and clearing of connections.

The temporal behavior of TCP relies on the underlying protocols (IP, Ethernet). Besides from this the buffering of TCP introduces additional delays and the multiplexing mechanism leads to unpredictable temporal behavior.

4.6 General/Internet Inter-Orb Protocol (GIOP/IIOP)

In 1996 the Object Management Group (OMG) released version 2.0 of their Common Object Request Broker Architecture (CORBA) specification. Among several other extensions the specification included “out of the box” interoperability by introducing the General Inter-Orb Protocol (GIOP) and its mapping to TCP/IP called Internet Inter-Orb Protocol (IIOP). The latest version of GIOP includes the following features (see also [OMG02b]):

? Common Data Representation (CDR): a representation of all data types of the IDL is defined, that is independent from the computer’s byte order.

? Request Multiplexing: allows sharing one transmission channel between several CORBA objects.

? Fragmentation: It is anticipated that value types may be rather large. Hence breaking up the serialization into an arbitrary number of chunks is supported in order to facilitate incremental processing.

? Connection Management: allows establishing and clearing of connections.

Since IIOP is a mapping of GIOP to a TCP/IP transport the temporal behavior depends besides from the current workload of the ORB on the client’s side as well as on the server’s side also on the temporal characteristic of TCP/IP (see previous section).

4.7 LIN

LIN is a single-wire serial communication protocol based on the UART interfaces which are available as low cost silicon module on almost all micro-controllers and can also be implemented as equivalent in software or as pure state machine for ASICs. The medium access in a LIN network is controlled by a master node so that no arbitration or collision management in the slave nodes is required, thus giving a guarantee of the worst-case latency times for signal transmission.
The maximum transmission speed is 20 kbit/s which results from the requirements of electromagnetic compatibility (EMC) and clock synchronization.
A node in LIN networks does not make use of any information about the system configuration, except from the denomination of the master node which allows adding nodes to the LIN network without requiring hardware or software changes in other slave nodes. A LIN network comprises one master node and one or more slave nodes. All nodes include a slave communication task that is split into a transmit task and a receive task, while the master node includes an additional master transmit task. The communication in an active LIN network is always initiated by the master task: the master sends out a message header which comprises the synchronization break, the synchronization byte, and the message identifier.
Exactly one slave task is activated upon reception and filtering of the identifier and starts the transmission of the message response. The response comprises two, four, or eight data bytes and one checksum byte. A message frame consists of the header and the response part.
The size of a LIN network is typically under 12 nodes (though not restricted to this), resulting from the small number of 64 identifiers and the relatively low transmission speed. The clock synchronization, the simplicity of UART communication, and the single-wire medium are the major factors for the cost efficiency of LIN.
Further details can be found in the LIN specification (see [Wen00]).

4.8 Control Area Network (CAN) and TTCAN

The Control Area Network (CAN) is a serial communications protocol. It supports prioritization of messages. This is achieved by a bit arbitration mechanism on the identifier (it is assumed that there is a dominant and a recessive state on the bus and the identifier consisting of dominant bits only has highest priority). Data rates up to 1 MBit are supported.

CAN uses 4 different types of frames: data frames, remote frames, error frames, and overload frames.

A data frame consists of the Start-of-Frame marker (a single dominant bit), arbitration field, control field, data field, CRC field, ACK field, and End-of-Frame (a sequence of seven recessive bits). The arbitration field contains the ID of the sender and the Remote-Transmission-Request (RTR) Bit. The control field contains a Data-Length-Code of 4 Bits. The data field contains from 0 to 8 Bytes of payload. Thus a data frame consists of 44 bit for the header plus 0-8 data bytes for payload.

A remote frame is similar to a data frame but contains no data field and is used by the receiver in order to request data from the sender.

Error frames and overload frames are used for signaling error conditions or delaying the transmission of further messages.

The bus access strategy is a Carrier Sense Multiple Access Collision Avoidance (CSMA/CA) and assumes that the bus provides a dominant state. The identifier of the node with highest priority consists of dominant bits only.

Further information is available in [Bos91].

TTCAN is a higher-layer protocol built on top of the unchanged standard CAN protocol that synchronizes the communication schedules of all CAN nodes in a network that provides a global system time. When the nodes are synchronized, any message can be transmitted at a specific time slot, without competing with other messages for the bus. Thus the loss of arbitration is avoided and the latency becomes predictable (see also [Har00]).

4.9 Time Triggered Protocol Class A (TTP/A)

TTP/A is a member of the TTP protocol family that is intended for low cost networks of sensors and actuators. TTP/A is ideally suited for integration with TTP/C into an overall architecture where each and every sensor's reading and actuator's command is completely predictable with respect to its temporal properties. This allows carrying out all sensing and actuating action within hard real-time deadlines and minimal jitter.

TTP/A is a time-triggered protocol based on the TDMA bus access scheme. However, in contrast to TTP/C it is a master/slave protocol, without redundant channels and with limited fault-tolerance capabilities. The master is responsible for the clock-synchronization and thus initiates the TDMA rounds. Since TTP/A is intended for low cost systems, a standard UART and an 8 bit micro controller are sufficient for the protocol. The communication protocol is integrated with the application software running on an operating system.

Communication is performed by using Master-Slave (MS) rounds or Multi-Partner (MP) rounds. While the MS rounds are used for the DM and CP interface the MP rounds allow temporally predictable communication of data via the RS interface.

The maximum transmission speed depends on the physical layer. There are several implementations based on UART frames at a data rate of 19.2 kbit/s. Data rates of up to 1 Mbit/s are specified on an RS485 physical layer or CAN physical layer. Since this

protocol is not restricted to UART frames other data rates are possible (e.g. on a fiber optic physical layer).
Due to the higher data efficiency the TTP/A protocol provides less jitter and better real-time response compared to the LIN protocol on a physical layer with the same speed (see [KEM00]).
Further details about this protocol can be found in [EEE+01].

4.10 Time Triggered Protocol Class C (TTP/C)

TTP/C is a member of the TTP protocol family that is intended as high speed network for high dependability applications. TTP/C is ideally suited for integration with TTP/A into an overall architecture where several fieldbus clusters have to be interconnected in a temporally predictable manner. This allows carrying out all sensing and actuating action within hard real-time deadlines and minimal jitter.
TTP/C is a time-triggered protocol based on the TDMA bus access scheme. However, in contrast to TTP/A it is no master/slave protocol and provides fault tolerance by the capability of tolerating one arbitrary faulty node or one faulty communication channel in the system. This is reached by utilizing redundant channels and two central bus guardians. Since TTP/C is intended for high dependable systems, the major part of the protocol is available in hardware (the C1 communication controller and its successor, the C2 communication controller). This controller provides to the microcontroller the communication network interface (CNI), which is implemented as dual-ported RAM and mapped into the address space of the microcontroller. This configuration acts as a temporal firewall (see [KN97]) where only state data but no flow of control crosses the boundary of the interface.
Communication is performed in a predefined TDMA schedule, called message descriptor list (MEDL), which is stored in the flash memory of the communication controller of each node. This protocol also provides a fault tolerant global notion of time.
The current implementation of the TTP/C protocol supports data rates up to 25 Mbit/s. Since the communication controller has to perform the protocol code in the inter frame gaps (IFG) there is a minimum duration for the IFG which depends on the clock speed of the controller and other parameters.
Further details of the TTP/C protocol can be found in [Kop02].

4.11 FlexRay

FlexRay is based on the TTP/C protocol, for this reason the fundamental concepts of the time-triggered architecture can also be found in FlexRay. Thus it is also based on a TDMA schedule for avoiding collisions, uses redundant communication channels which are protected from babbling nodes by a bus guardian, and provides a fault tolerant global notion of time.
Communication is done in a communication cycle consisting of a static and a dynamic segment, where each of the segments may be empty. The sending slots are used deterministically (in a pre-defined TDMA strategy) in the static segment. In the dynamic segment there can be differences in the phase on the two channels. It is possible to send different data in the same sending slot on different channels. Nodes that are connected to both channels send their frames in the static segment simultaneously on both channels. Two nodes, that are connected to one channel only, but not the same channel, may share a slot in the static segment.

To guarantee the consistency of the clock synchronization only nodes which are received by all other nodes are allowed to participate. All nodes execute the clock synchronization algorithm, but only the frames of the static segment are considered.
As described in [Kop01] the FlexRay protocol differs from the TTP/C protocol (see previous section) by the point of view of the tradeoff between contradicting requirements. While in TTP/C safety, composability, and flexibility (in this order) are of outmost importance, the FlexRay protocol tilts away from safety and composability towards flexibility.
Further details of the FlexRay protocol can be found in [BBE+02].

4.12 PROFIBUS

The history of the PROcess FIeld BUS (PROFIBUS) started in 1987 in Germany where 21 companies and institutes joined forces and aimed at realization and establishing of a bit-serial fieldbus. The communication layer is specified in the standards IEC 61158 and IEC 61784 for speeds of 9600 kBit/s – 12 MBit/s.
From the technological standpoint the lower level (communications) of the system structure of PROFIBUS is based on the ISO/OSI reference model: It has a modular design and offers several different communication technologies, application and system profiles, as well as device management tools.
As bus access protocol, PROFIBUS allows the master-slave procedure, supplemented by the token passing procedure for coordination of several masters on the bus. This guarantees that each master may access the bus within an a priori known interval.
Further information about Profibus can be found at [PI02].
In Profibus the presence of stations with highly differing dynamics can cause high-priority messages to miss their deadline if the workload increases (see [BM00]).

4.13 WordFIP / FIP

The WorldFIP protocol is based on a white paper defining the Flux Information Process (FIP) concept that has been published in 1984 and is designed to satisfy the following two communication requirements of process control systems: Operational communication between sensors and actuators and automated equipment. Communication of information between equipment in the control room and automated devices, as well as communication between configuration and maintenance tools and equipment in the system, including sensors and actuators. The entire WorldFIP protocol is covered by the standards EN 50170-3 and EN 50254.
The Fieldbus Internet Protocol (FIP) combines the advantages of the Internet Protocol and WorldFIP with supported data rates from 31.25 kb/s to 25 Mb/s. In addition to compatibility with the installed base, it conserves all WorldFIP advantages such as synchronization and availability. It also provides services offered by the Internet, including true access to multimedia upon request.
Further information is available in [Wor01].

4.14 EN 50170, EN 50254, EN 50325

It has been tried to standardize a fieldbus protocol that included the best of the available national standards. Since the favorites, PROFIBUS and FIP, have been completely different it has been impossible to integrate both protocols in one single protocol. After more than 10 years this standard included both protocols and some other protocols that have been developed in the meanwhile.

In order to make the standards easier to handle the available standards have been bundled according to their primary application areas (see [FS02]): “General purpose field communication systems” (EN 50170), “High efficiency communication subsystems for small data packages” (EN 50254), and CAN based standards (EN 50325).

Among these standards are PROFIBUS, WorldFIP, Foundation Fieldbus, P-NET, ControlNet, INTERBUS, DeviceNet, SDS, and CANOpen.

4.15 SERCOS

The SErial Realtime COmmunications System (SERCOS) defines an open, digital standardized interface for communication between digital controls, drives, sensors and actuators for numerically controlled machines and systems.
Development began in 1988 and the physical layer and communications protocol were accepted as international standard IEC 61491 in 1995.
The SERCOS interface uses fiber optics in ring topology as the communications medium for high noise immunity and electrical isolation. It operates at standardized data rates between 2 and 16 Mbit/s and allows up to 254 nodes per ring.
There are three types of data telegrams that are received simultaneously by all nodes (the media is accessed in an a priori known TDMA schedule): Master-Sync-Telegram (for distributing time information), Master-Data-Telegram, and Slave-Data-Telegram (for communication between master and slave). Each Data-Telegram contains a field for predefined cyclic data and a service-field for asynchronous requests.
It has been designed to include mechanisms to ensure the high level of determinism required when synchronizing multiple axes of digital drives and directly supports position, velocity and torque commands which can be set with a precision of about 1 µs. The cycle time may be set to some specified values down to 62 µs.
For further details see [Ser99] or the standards mentioned above.

5 Comparative Evaluation

The temporal behavior of Ethernet is deterministic only when no collisions occur on the media. If this can be guaranteed Ethernet provides low latency, constant delay, and minimal jitter.

Wireless communication has similar restrictions as Ethernet but in addition the data rate depends on external disruption as well as collisions because of other devices on the same frequency spectrum.

The temporal properties of IP and UDP/IP are inherited from the underlying layer.

Since TCP/IP introduces retries in the case of lost packages, the temporal behavior is unpredictable. If it can be assumed that the underlying network provides a high dependability no retries are necessary and the temporal behavior of TCP/IP is similar to the behavior of the IP layer.

GIOP/IIOP relies on TCP/IP or another stream oriented transport and inherits its temporal properties from the underlying layer.

LIN is specified for 20 kbit/s only and suffers from its high jitter.

The worst case delay in a CAN network is too high because of the possibility of collisions. If TTCAN is used the mandated temporal properties could be established.

TTP/A is well suited from the point of view of the temporal properties but usually its implementations lack the resources for running RT-CORBA.

TTP/C provides the necessary temporal properties on protocol level and is designed to tolerate one arbitrary fault. In addition a global timebase is provided that allows synchronization of tasks. At TTTech (see [TTT02]) there are several Motorola 68k based computation nodes available as well as an implementation with more memory based on the Power PC that provides enough resources for running some demanding real-time tasks with CORBA.

In Profibus the presence of stations with highly differing dynamics can cause high-priority messages to miss their deadline if the workload increases.

FlexRay provides similar properties than TTP/C but lacks composability due to its event-triggered part.

The SERCOS is based on similar mechanisms as the TTP/A protocol. It is designed for controlling digital drives and includes a lot of commands that are specific for this domain. Transmission of CORBA messages is should be possible if a packet service is established on top of SERCOS.

6 Conclusion

Apart from a predictable communication protocol with constant delay and minimal jitter the TTP/C protocol also provides a common notion of time and provides a dependable communication because of its capability of tolerating one arbitrary fault. Thus it is best suited for implementing a predictable and dependable CORBA system. The implementation based on the PowerPC could be used as a test bed for some demanding tasks based on CORBA.
If the requirements for a precise global timebase and dependability are relaxed other systems (e.g. flexray, switched Ethernet, etc.) become usable.

7 References

[BBE+02] Ralf Belschner, Josef Berwanger, Christian Ebner et al.; FlexRay: Requirements Specification; BMW, DaimlerChrysler, Robert Bosch, General Motors, and Opel, Germany, 2002.

[BM00] L. Lo Bello, O. Mirabella; Multi-Ring Scheduling Strategies for Profibus Networks; Università di Catania, Facolta di Ingegneria, Italy, 2000.

[Bos91] Robert Bosch GmbH; CAN Specification Version 2.0; Robert Bosch GmbH, Stuttgart, Germany, 1991.

[Cox96] Alan Cox; Network Buffers And Memory Management; Linux Journal, issue 30, October 1996, available at http://www.linuxjournal.com/.

[EEE+01] Stephan Eberle, Christian Ebner, Wilfried Elmenreich et al.; Specification of the TTP/A Protocol; Research Report No 61/2001; Institut für Technische Informatik, Technische Universität Wien, Austria, 2001.

[Har00] Florian Hartwich, Bernd Müller, Thomas Führer et al.; CAN Network with Time Triggered Communication; Robert Bosch GmbH, Germany, 2000, available at http://www.can.bosch.com/.

[HS02] Øyvind Holmeide and Tom Skeie; VoIP Drives Real-Time EtherNet; Industrial Ethernet, Issue 5, 2002, available at http://ethernet.industrial-networking.com/.

[IEEE802.3] IEEE; Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications; Institute of Electrical and Electronics Engineers, New York, NY, U.S.A., 2000.

[IEEE802.11] IEEE; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications; Institute of Electrical and Electronics Engineers, New York, NY, U.S.A., 1999.

[JKK+01] Cliff Jones, Mark-Olivier Killijian, Hermann Kopetz et al.; Revised Version of DSoS Conceptual Model; University of Newcastle upon Tyne, Technical University of Vienna, LAAS-CNRS, and QinetiQ, 2001.

[KEM00] Hermann Kopetz, Wilfried Elmenreich, Christoph Mack; A Comparison of LIN and TTP/A; 3rd IEEE International Workshop on Factory Communication Systems (WFCS), 2000.

[KN97] Hermann Kopetz and Roman Nossal; Temporal Firewalls in Large Distributed Real-Time Systems, 6th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems, Tunis, Tunesia, 1997.

[Kop97] Hermann Kopetz; Real-Time Systems, Design Principles for Distributed Embedded Applications; Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[Kop00] Hermann Kopetz; Software Engineering for Real-Time: A Roadmap; Proceedings of the 22nd International Conference on Future of Software Engineering (FoSE) at ICSE 2000, 4th - 11th June 2000, Limerick, Ireland, 2000.

[Kop01] Hermann Kopetz; A Comparison of TTP/C and Flexray; Research Report 10/2001; Institut für Technische Informatik, Technische Universität Wien, Austria, 2001.

[Kop02] Hermann Kopetz; TTP/C Protocol – Version 1.0.0; TTTech Computertechnik AG, Vienna, Austria, 2002, available at http://www.ttpforum.org/.

[Kru98] Philippe Kruchten; Modeling Component Systems with the Unified Modeling Language; in Proceedings of the International Workshop on Component-Based Software Engineering, 1998.

[Lee99] Edward A. Lee, Embedded Software – An Agenda for Research, University of California, Berkely, CA, U.S.A., 1999.

[LO01] Thomas Losert, Roman Obermaisser; Wireless Real-Time Communication Technologies: A Comparative Study; Proceedings of the IEEE Workshop on Real-Time Embedded Systems, London, United Kingdom, 2001.

[NMG01] Edgar Nett, Michael Mock, Martin Gergeleit; Das drahtlose Ethernet; Der IEEE 802.11 Standard: Grundlagen und Anwendung; Addison-Wesley, München, Germany, 2001.

[OMG00] OMG; Extensible Transport Framework for Real-Time CORBA Request For Proposal; document number orbos/2000-09-12, Object Management Group, Needham, MA, U.S.A., 2000, available at http://www.omg.org/.

[OMG01] OMG; The Open Communications Interface (OCI); Iona, U.S.A, Object Oriented Concepts, Australia, 2001, available at http://www.omg.org/.

[OMG02a] OMG; Extensible Transport Framework Joint Revised Submission; document number mars/2002-09-06, Borland, OIS, Vertel, U.S.A., 2002, available at http://www.omg.org/.

[OMG02b] OMG; CORBA 3.0 Specification, chapter 15, document number formal/2002-06-51, Object Management Group, Needham, MA, U.S.A., 2002, available at http://www.omg.org/.

[PI02] Profibus International; Profibus: Technical Overview; Profibus User Organization (PNO), Germany, 2002, available at http://www.profibus.com/.

[RSG89] Johannes Reisinger, Wolfgang Schwabl, Günter Grünsteidl; A Survey of MARS; Research Report 16/1989; Department of Computer Science, Vienna University of Technology, Austria, 1989.

[RFC768] Jon Postel, editor; User Datagram Protocol: DARPA Internet Program Protocol Specification; Information Sciences Institute, Marina del Rey, California, U.S.A., Aug. 1980.

[RFC791] Jon Postel, editor; Internet Protocol: DARPA Internet Program Protocol Specification; Information Sciences Institute, Marina del Rey, California, U.S.A., Sep. 1981.

[RFC793] Jon Postel, editor; Transmission Control Protocol: DARPA Internet Program Protocol Specification; Information Sciences Institute, Marina del Rey, California, U.S.A., Sep. 1981.

[Ser99] Sercos; IEC 61491, EN 61491 SERCOS interface: Technische Kurzbeschreibung; available at http://www.sercos.com/, 1999.

[Sie00] Jon Siegel, CORBA 3: Fundamentals and Programming, John Wiley & Sons, New York, NY, U.S.A., 2000.

[TTT02] TTTech Webpage, available at http://www.tttech.com/, 2002.

[Wen00] H.-Chr. v. d. Wense; editor; LIN Specification Package: Revision 1.2; Motorola, Munich, Germany, 2000.

[Wor01] WorldFIP Association; WorldFIP Summary; WorldFIP Association, available at http://www.worldfip.org/, 2001.