Real-Time Communication

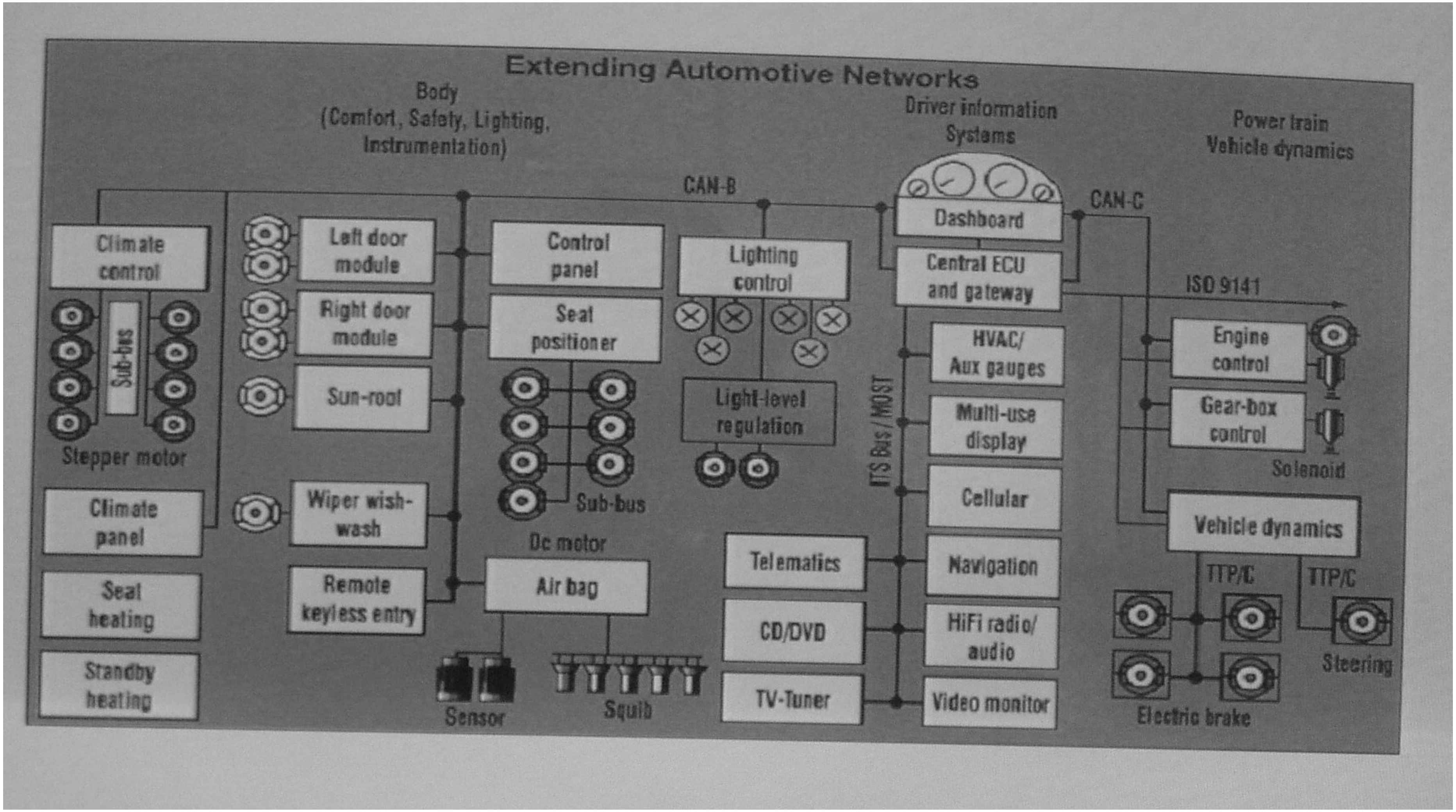
Importance of Real-Time Communication

For the following reasons, *distributed systems* are the dominant architectural choice for many real-time applications:

- *Composability*: the construction of new applications out of existing pre-validated components
- *Intelligent Instrumentation--integration* of sensor/actuator, local processing and communication on a single die
- *Reduction of wiring harness*
- *Avoidance of a single point of failure* in safety critical applications.

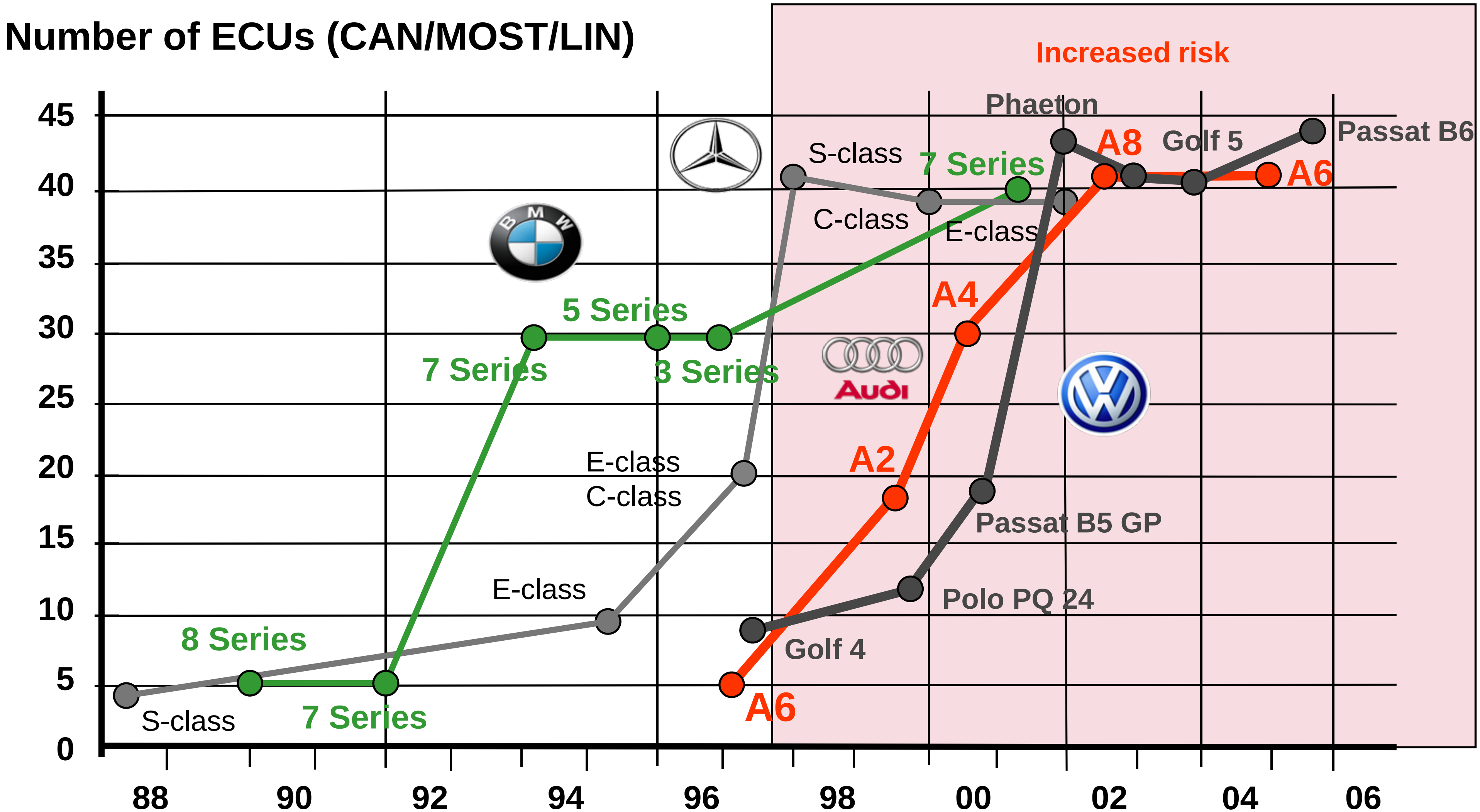
In any distributed real-time system, a proper real-time communication service is of central importance

Example of the Networks onboar a Car



From R. Basserone, R. Marculescu, Communication/Component Based Design, June 2002

We are Getting Overwhelmed by Boxes and Wires



Source: Prof. Dr. J□ürgen Leohold, TU Vienna Summer School 2004 on *Architectural Paradigms for Dependable Embedded Systems*

History of Real-Time Protocols

Over the past decades, many domain-specific real-time protocols have appeared on the market, such as:

CAN

Profibus

AFDX

TTP

FlexRay

Real-Time Ethernet

etc.

None of these protocols has penetrated the market in manner that is comparable to standard Ethernet in the non-real-time world.

A Protocol Consolidation is Expected

Technological and economic developments will enforce a protocol consolidation in the near future:

- According to the highly respected IRTS 2007, the cost of production per million bits of DRAM will fall from 0.96 cent in 2007 to 0.06 cent in 2015. The cost of design and mask generation of an SoC is more 10 Million Dollars that must be amortized over each hardware protocol implementation.
- Interoperability requirements require protocol compatibility.
- Every unique protocol requires a unique set of software modules and development tools that must be developed and maintained.
- Substantial human resources must be deployed to learn about and gain experience with a new protocol.

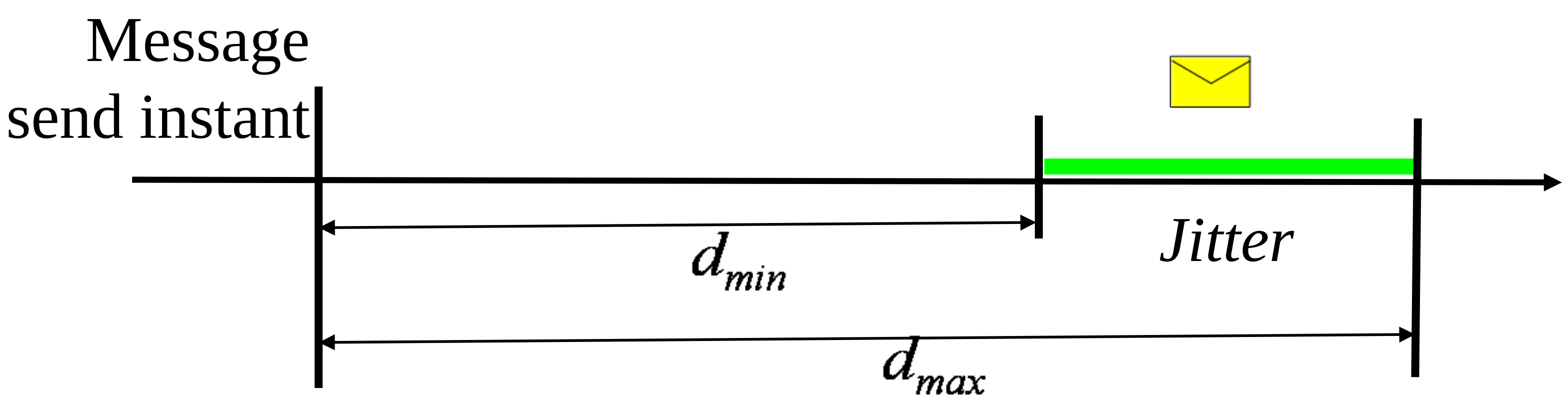
Properties of a Successful Protocol

A successful real-time protocol must have the following properties:

- ♦ Sound theoretical foundations w.r.t. time, determinism, security, and composability.
- ♦ Support for all types of real-time applications, from multimedia to safety-critical control systems.
- ♦ Support error containment of failing nodes
- ♦ Economically competitive--a hardware SoC protocol controller should cost less than 1 €.
- ♦ Compatibility with the Ethernet standard that is widely used in the non-real-time world will reduce the software and human effort.

Timeliness of a Communication Channel

Given that the *operating system* requests the transmission of a message at the *start instant* t_{start} then the *receive instants* $t_{receive}$ at all correct receivers of the (multicast) message will be in the interval $< t_{start} + d_{min}, t_{start} + d_{max} >$, where d_{min} is called the *minimum delay* and d_{max} is called the *maximum delay*. The difference $d_{max} - d_{min}$ is called the *jitter* of the transmission channel. d_{min} and d_{max} are *a priori* known characteristic parameters of the given transmission channel.



Open-World System

An *open-world system* is a system where an *(unknown) number* of *uncoordinated* clients *compete* for the services of a server. In such a system there is always the possiblity that *n senders* send a message to the *same receiving port* at the *same instant*. There two alternatives to resolve this conflict in the network:

- Exercise *back-pressure flow control* on the *n-1 senders* that did not succeed in getting the message through or
- Store the *n-1 messages* in the networks.

Both alternatives are unsuitable for real-time messages.

Closed-World System

- In a *closed-world system* the number of clients is *limited and known a priori*. The clients *cooperate* with each other such that the server is in the position to meet the requests of all clients within specified temporal bounds.
- **Temporal guarantees can only be given in a closed-world system.**

Flow Control

Flow Control deals with the control of the information flow between communicating partners, such that the sender does not outpace the receiver.

In any communication scenario, it is the receiver who should determine the speed of communication.

We distinguish between

- ♦ Explicit flow control and
- ♦ Implicit flow control

Implicit Flow Control

Sender and receiver establish a priori, i.e., before run time, that a maximum send rate will not be exceeded by the sender and that this rate will be accepted by the receiver.

- At run time, the communication channel can be unidirectional.
- Error detection is in the responsibility of the receiver.
- Implicit flow control is well suited for multicast communication services.
- Diffusion based flow control.

Explicit Flow Control

- The sender sends a message to the receiver and waits until the receiver has explicitly acknowledged the receipt of this message.
- The *sender must be in the sphere of control of the receiver*, i.e., the receiver has the authority to slow down the sender (sometimes called *back pressure* flow control).
- Error detection is in the responsibility of the sender.
- A message that is not acknowledged implies that either the message has been lost or the receiver is late or the receiver has failed.

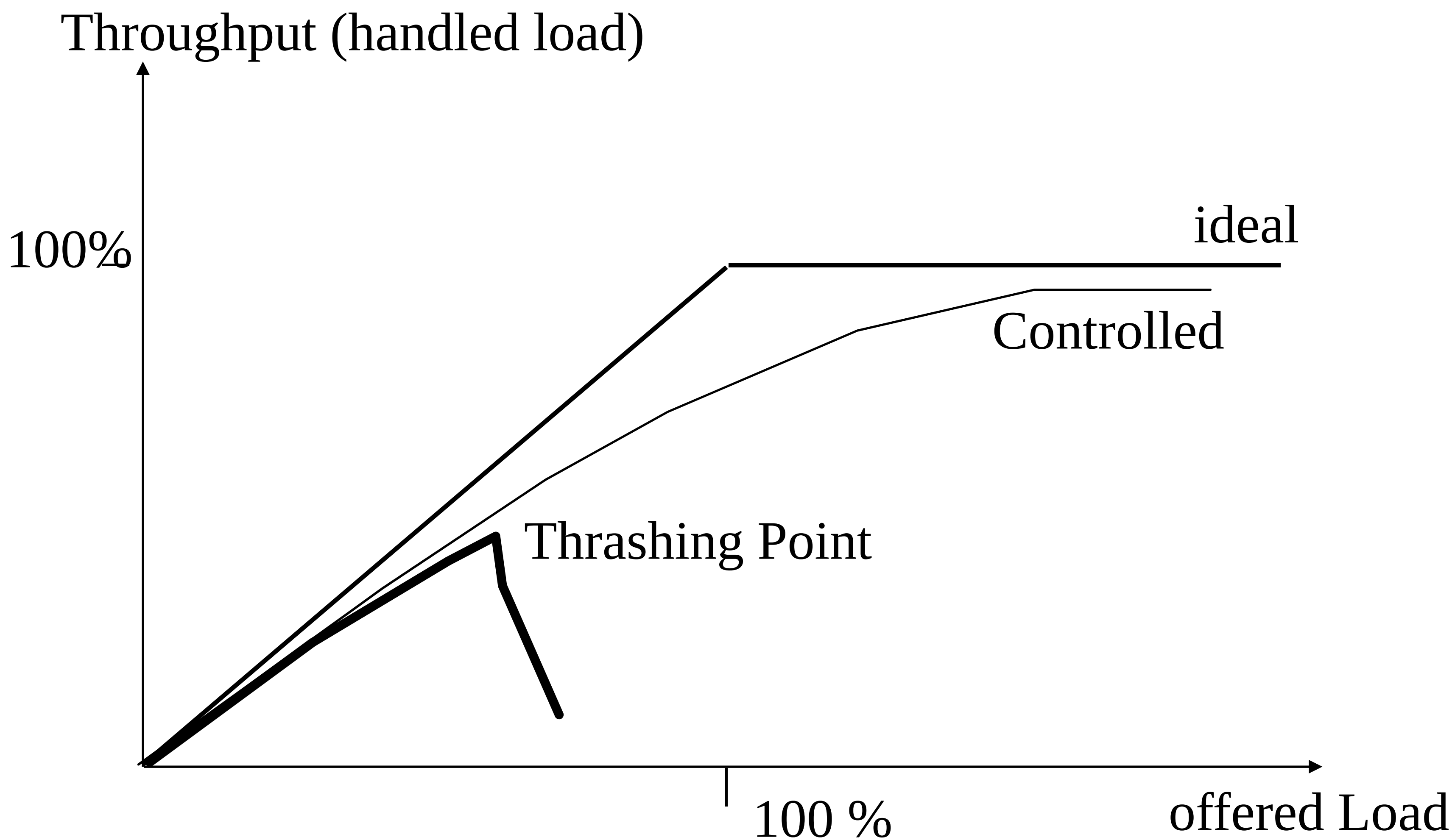
In an on-line transaction processing system, back pressure flow control can be exercised. If the computer system is overloaded it slows down and extends its response time, thus forcing the clerks to reduce the request rate.

Explicit Flow Control can Lead to Thrashing

Thrashing can be caused be explicit flow control.

If, under high load conditions, a message is not acknowledged within the specified timeout, then the sender resends the message causing an increase in the message rate at the worst possible time.

Throughput Load Characteristic



Causes for Thrashing:

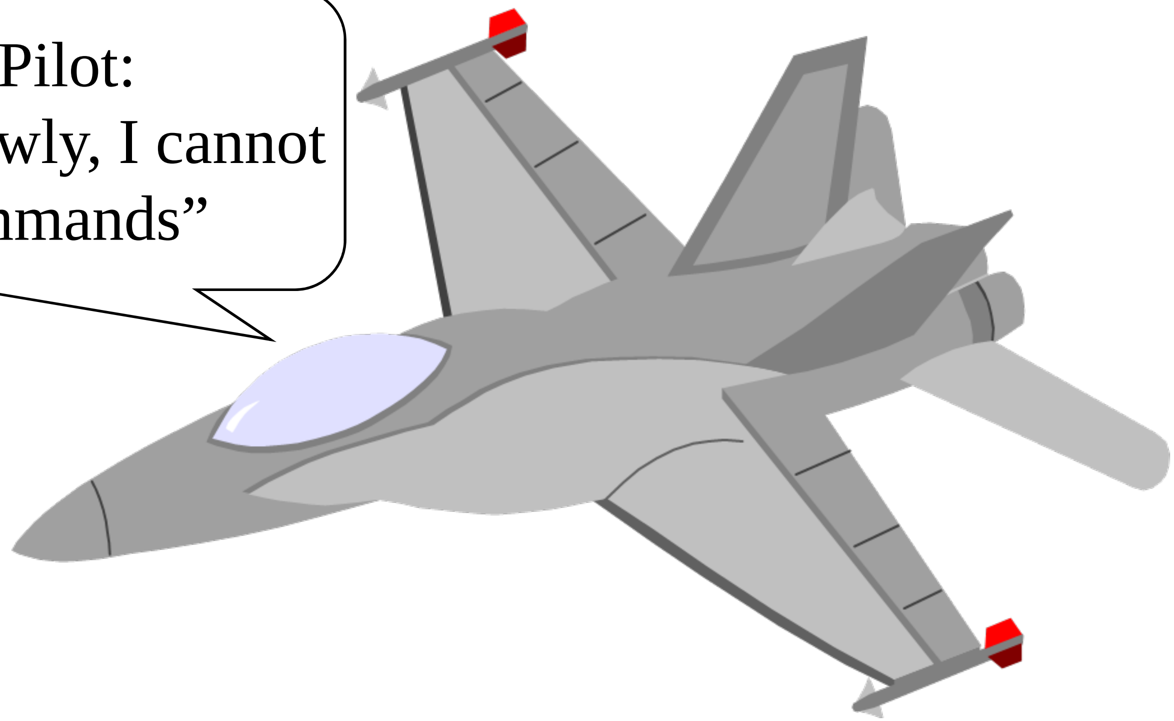
Mechanism that generate more service requests or consume more resources as the load increases, e.g.:

- Retry mechanism in communication protocol
- Buffer managementThrashing can be avoided by flow control and congestion control.

Consider the multicast communication in an ET system!

Example: Sphere of Control (SOC)

Computer to Pilot:
"Please fly more slowly, I cannot follow your commands"

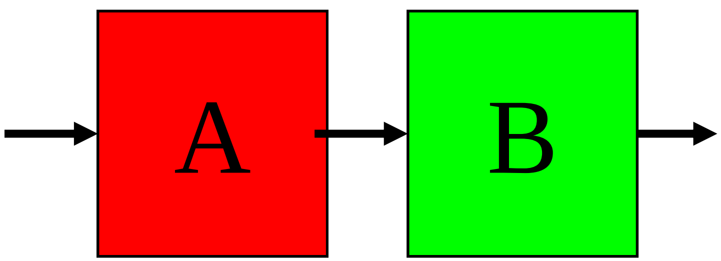


RT Communication System must provide

- Predictable Communication Service for the transmission of real-time data --*deterministic multicast unidirectional message*
  - *Determinism*
    Timeliness
    Complexity Reduction
    Testing
    Active Redundancy (e.g., TMR)
    Certification
  - *Multicast* --independent non-intrusive observation, TMR
  - *Uni-directionality* --separate *communication* from *computation*
- Flexible best-effort Communication Service for the transmission of non-real-time data coming from an open environment
- Support for Streaming Data
- Dependability

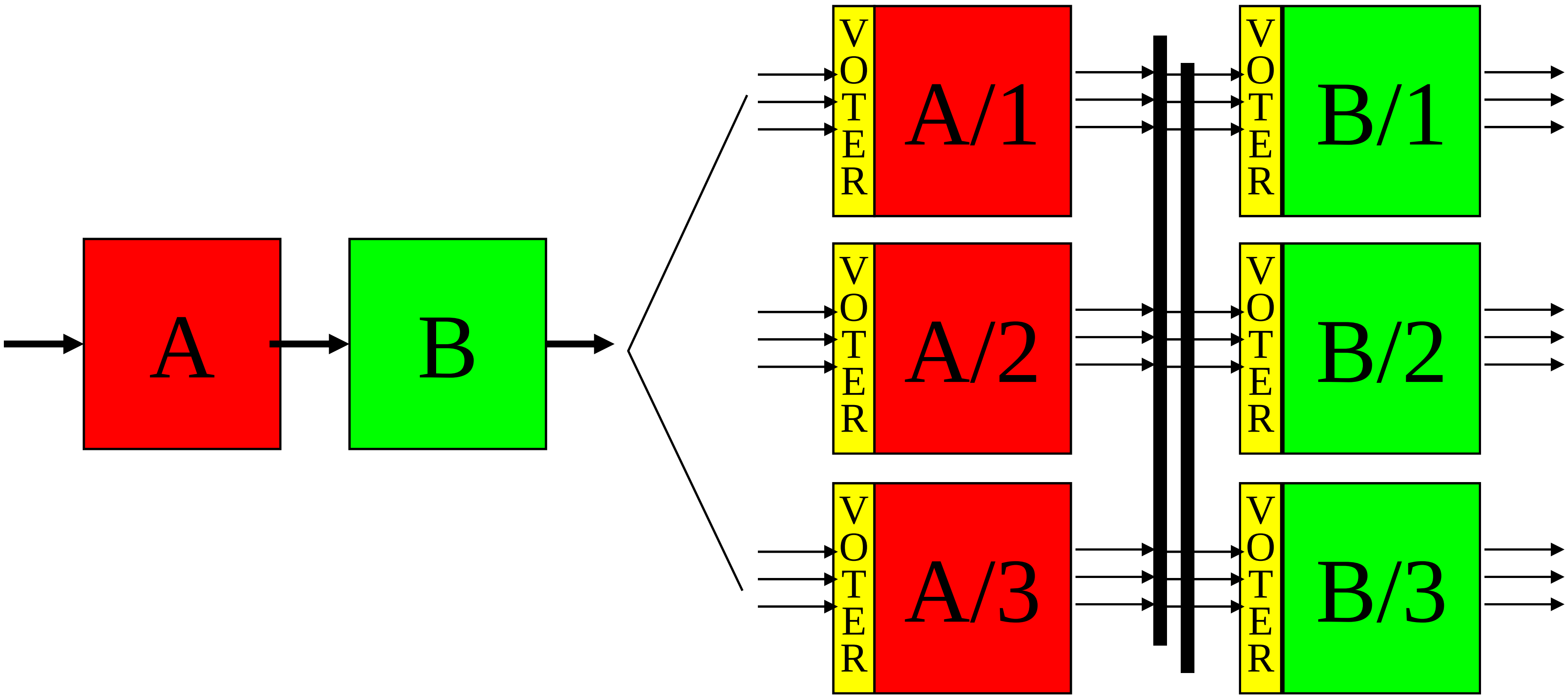
Mitigation at the Architecture Level: TMR

Triple Modular Redundancy (TMR) is the *generally accepted technique* for the mitigation of component failures at the system level:



Fault-Handling at the Architectural Level: TMR

Triple Modular Redundancy (TMR) is the *generally accepted technique* for the mitigation of component failures at the system level:



Innate Conflicts in Real-Time Protocol Design

- *Temporal Guarantees*
- *Synchronization Domain*
- *Error Containment*
- *Consistent Ordering of Events*
- *Determinism*

Temporal Guarantees (I)

*It is **impossible** to provide tight temporal guarantees in an open communication scenario.*

If every sending component in an open communication scenario is autonomous and is allowed to start sending a message at any instant, then it can happen that all sending components send a message to the same receiver at the same instant (the *critical instant*), thus overloading the channel to the receiver. In fielded communication systems, we find three strategies to handle such a scenario:

- The communication system *stores* messages intermediately
- The communication system exerts *back-pressure* on the sender.
- The communication system *discards* some messages.

None of these strategies is acceptable for real-time data.

Temporal Guarantees (II)

Tight temporal guarantees can only be given, if the senders cooperate and coordinate their sending actions such that there is no channel conflict among the senders of real-time messages.

Such a coordination can be achieved by establishing a conflict free schedule for sending real-time data, based on the reference to a common time-base.

Single Synchronization Domain

*It is **impossible** to support more than a single synchronization domain for the temporal coordination of components within a RT-cluster.*

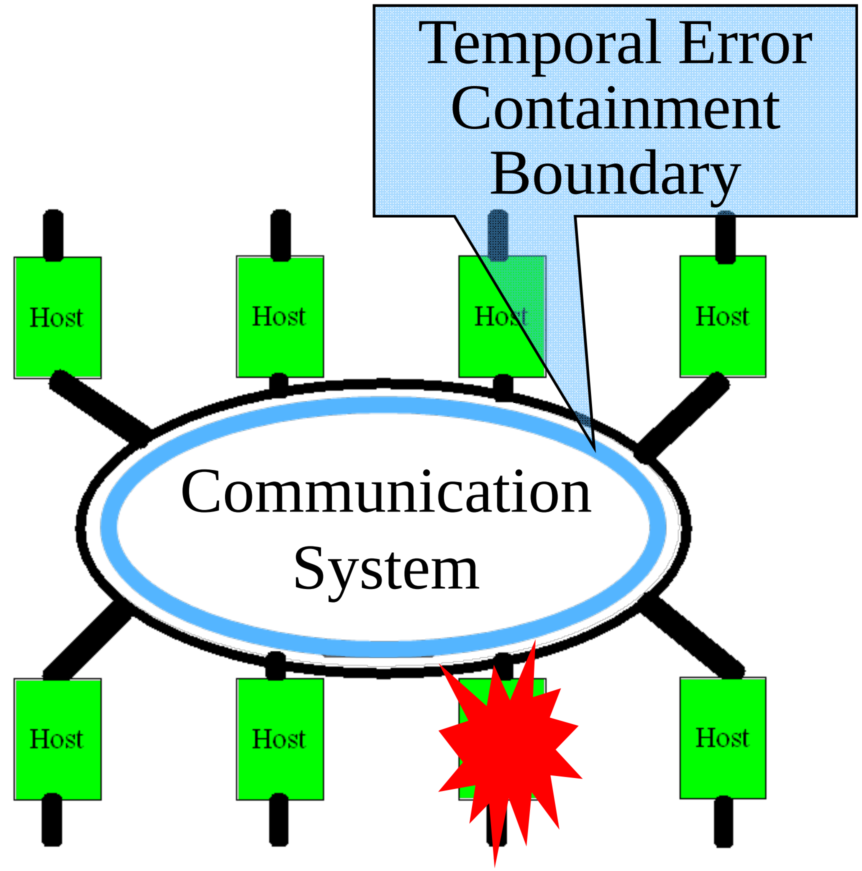
The synchronization within a cluster can be established either by reference to a ***single global time*** or by reference to a single ***leading data source***.

Example: A dynamic scenario that is observed by a set of smart cameras.

Error Containment

It is ***impossible*** *to maintain the communication among the correct components of a RT-cluster if the temporal errors caused by a faulty component are not contained.*

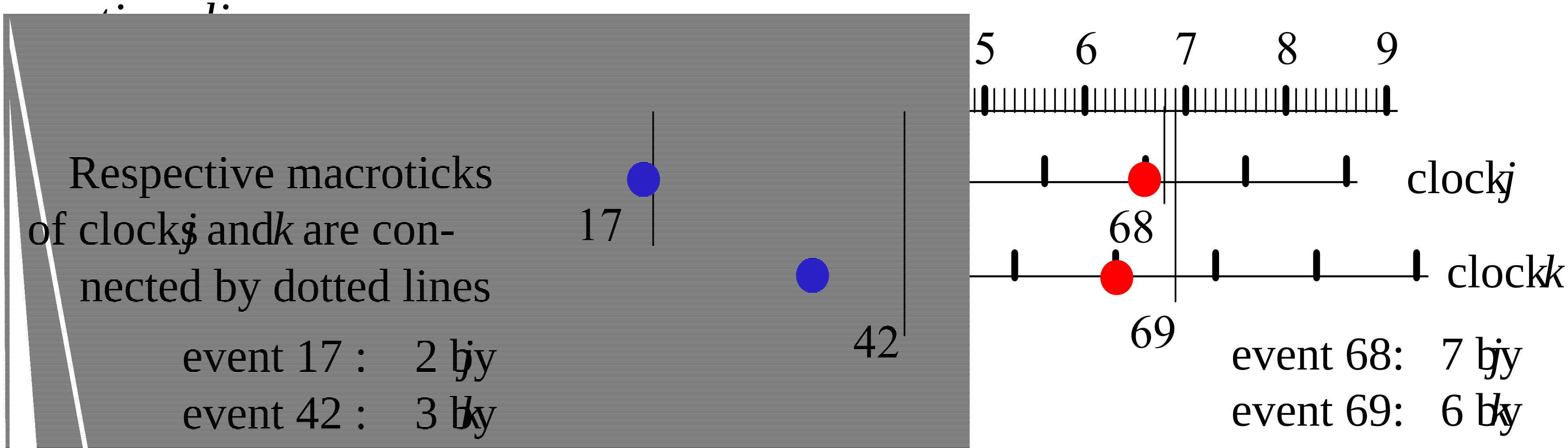
Error containment of an *arbitrary* node failure requires that the Communication System has *temporal information* about the allowed behavior of the nodes--it must contain *application-specific* state.



Error Containment

Consistent Ordering of Events

*It is **impossible** to establish, on the basis of their digital timestamps, a consistent view of the temporal order of events in a distributed system, if the events can be generated at any instant of a dense time line.*



Because of the accumulation of the synchronization error and the digitalization error, it is not possible to reconstruct the temporal order of two events from the knowledge that the global timestamps differ by one.

Determinism

*It is **impossible** to build a deterministic distributed real-time system without the establishment of some sort of a sparse global time base for the consistent time-stamping of the sparse events.*

Without a sparse global time base and sparse events, simultaneity cannot be resolved consistently in a distributed system, possibly resulting in an inconsistent temporal order of the messages that report about these simultaneous events. Inconsistent ordering results in the loss of *replica determinism*.

The assignment of events to a sparse global time-base can be established at the the system level by the generation of *sparse events* or at the application level by the execution of agreement protocols which assign consistently dense events to sparse intervals.

Event -Triggered (ET) Protocols

- The protocol execution is initiated by an event at the sender at an arbitrary point in time
- Maximum execution time and reading error are large compared to the average execution time
- Error detection is in the responsibility of the sender, since only the sender knows when a message has been sent
- To realize reply to the sender. This results in correlated traffic in a multicast environment.
- Temporal encapsulation is not provided.
- Explicit flow control must be implemented to protect the receiver from information overflow. This requires that the sender is in the sphere of control of the receiver.

Time - Triggered (TT) Protocols

- The protocol execution is initiated by the progression of the global time. The point in time when a message will be sent is known *a priori* to all receivers.
- Maximum execution time is about the same as average execution time. This results in a small reading error.
- Error detection is in the responsibility of the receiver, based on his a priori knowledge.
- The protocol is unidirectional, well suited for a multicast environment.

Event Message versus State Message I

Characteristic	Event Message	State Message
Example of message contents	"Valve has closed by 5 degrees"	"Valve stands at 60 degrees"
Contents of data field	event information	state information
Instant of sending	After event occurrence	Periodically at *a priori* known points in time.
Temporal control	Interrupt caused by event occurrence	sampling, caused by the progression of time
Handling at receiver	queued and consumed on reading	new version replaces previous version, not consumed on reading
Semantics at receiver	Exactly once	At least once

Event Message versus State Message II

Characteristic	Event Message	State Message
Idempotence [Kopetz97, p.110]	no	yes
Consequences of message loss	Loss of state synchronization between sender and receiver	Unavailability of current state information for a sampling interval.
Typical communication protocol	Positive Acknowledgment or Retransmission (PAR)	Unidirectional datagram
Typical communication topology	Point to point	Multicast
Load on communication system	Depends on number of event occurrences	Constant

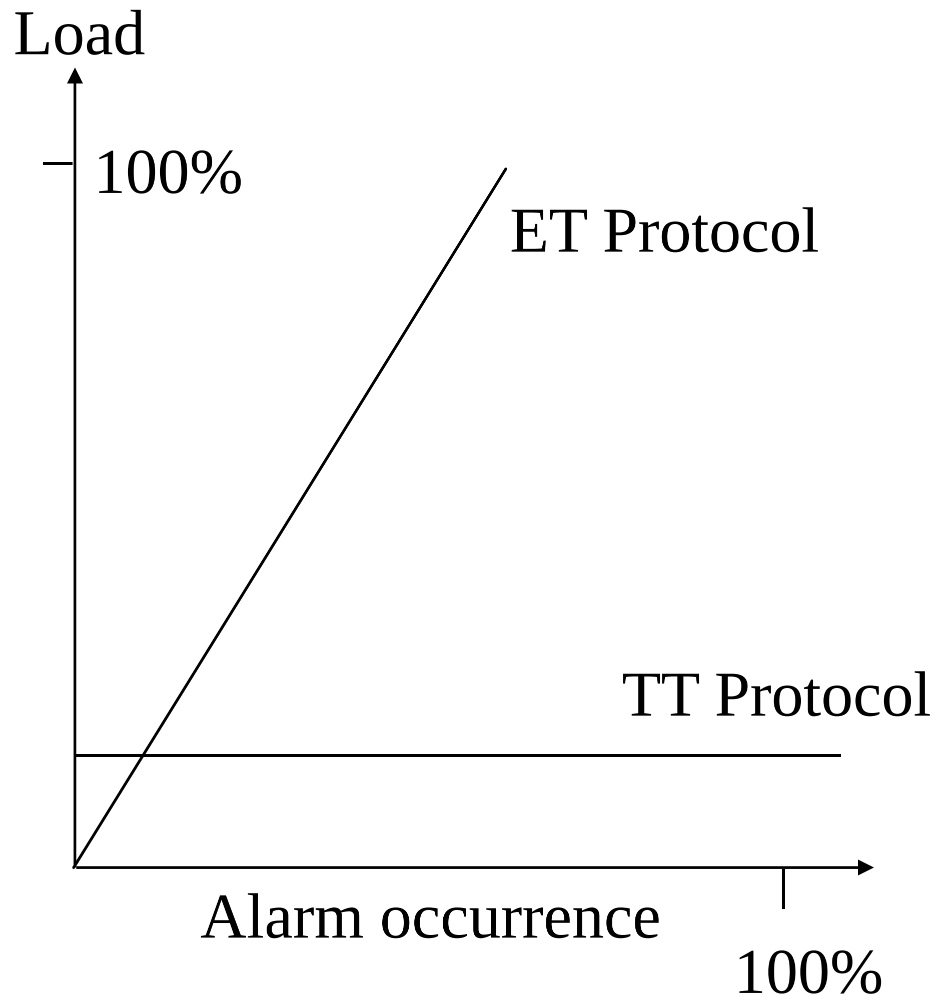
Performance ET vs. TT Protocols

Let us assume a distributed system with 10 nodes and a 100 kb communication channel. Each node monitors 40 different alarms that have to be communicated to a passive alarm monitor within 100 msec. CAN is used as a communication system

	ET System	TT system
control	alarm occurr.	100 msec/node
Alarm encoding	1/message	40/message
message length	8+44=52	40+44=84
Maximum number of mes.	ca. 2000	ca. 1200
Messages needed	0 ... 4000 (0-200%)	100 (constant 9 %)

Load Comparison

In peak-load situations, the ET protocols generates much more traffic than the TT protocol.



Example: PAR

The PAR (Positive Acknowledgment or Retransmission Protocol), the most common protocol class in the OSI standard relies on explicit flow control:

- ♦ The sender takes a message from its client and sends it as a uniquely identified packet
- ♦ The receiver acknowledges a properly received packet, unpacks it and delivers the message to its client
- ♦ If the sender does not receive an acknowledgment within the timeout period t_1 it retransmits the packet
- ♦ If the sender does not receive an acknowledgment after k retransmissions, it terminates the operation and reports failure to its client.

Action Delay of PAR

Consider a system where a PAR protocol with k (2) retries is implemented on top of a token protocol (transmission time can be neglected):

TRT Maximum Token Rotation Time (e.g. 10 msec)

Timeout of PAR: 2 TRT

dmin = 0

dmax = (2k + 1) TRT = 5 TRT

Maximum action delay = 10 TRT (100 msec)

In OSI implementations PAR protocols are stacked!

OSI and Real Time

The OSI model has not been targeted at real-time applications. In the real-time environment it has the following deficiencies:

- ♦ The implicit assumption that the sender is in the SOC of the receiver does not hold for RT-systems
- ♦ The maximum protocol execution time d_{max} and the reading error ε increase exponentially with the number of levels
- ♦ The automatic retry mechanism within the different protocol levels is a fertile ground for thrashing
- ♦ The protocols provide no temporal encapsulation of the subsystems--no constructive testing possible
- ♦ Replica Determinism is not supported.

Maximum Protocol Execution Time d_{max}

at the transport level depends on

- Protocol stack at sender (including error handling)
- Message scheduling strategy at sender
- Media access protocol
- Transmission time
- Protocol stack at the receiver
- Task scheduling at the receiver

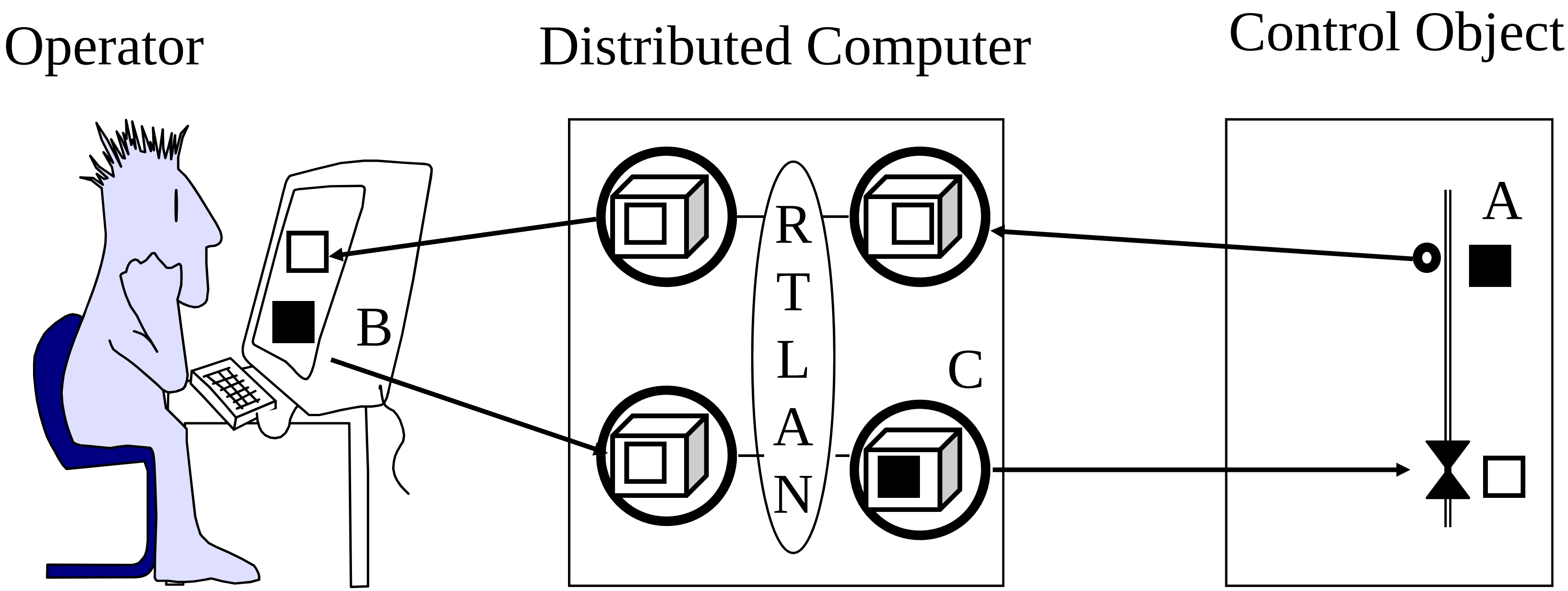
In general purpose operating systems, the execution path lengths for the transport of a single message can be tens of thousands of instructions.

Limit of the Communication Reliability

In a real time system, there is fundamental limit to the reliability of any communication:

- ♦ In a real-time system it is required to produce an effect in the environment (e.g., closed control valve)
- ♦ There are always subsystems (e.g. mechanical) in the loop that have to operate properly to achieve the desired effect.
- ♦ Successful communication can only indicate that the command has been issued to the subsystem, but not whether the subsystem has achieved its purpose
- ♦ Assurance can only be derived from a sensor that observes the desired effect.

End-to-End Example



■ RT Entity □ RT Image RT Object

A: Measured Value of Flow

B: Setpoint for Flow C:Intended Valve Position

End - to - End Protocol

An end-to-end Protocol is a protocol that monitors and controls the intended effect of a communication at the intended endpoints.

- ♦ In the previous example (control of the flow in a pipe), the end-to-end acknowledgment of a command message to the actuator is the sensor message reporting about the intended change in the flow through the pipe.
- ♦ Intermediate level protocols are only needed if the communication is less reliable than the other subsystems.
- ♦ Intermediate level protocols simplify the diagnosis.

High error detection coverage can only be achieved with end-to-end protocols.

Three Mile Island Accident

Quote [Sev81] about the Three Mile Island Nuclear Reactor #2 accident on March 28, 1979 :

Perhaps the single most important and damaging failure in the relatively long chain of failures during this accident was that of the Pressure Operated Relief Valve (PORV) on the pressurizer. The PORV did not close; yet its monitoring light was signaling green (meaning closed).

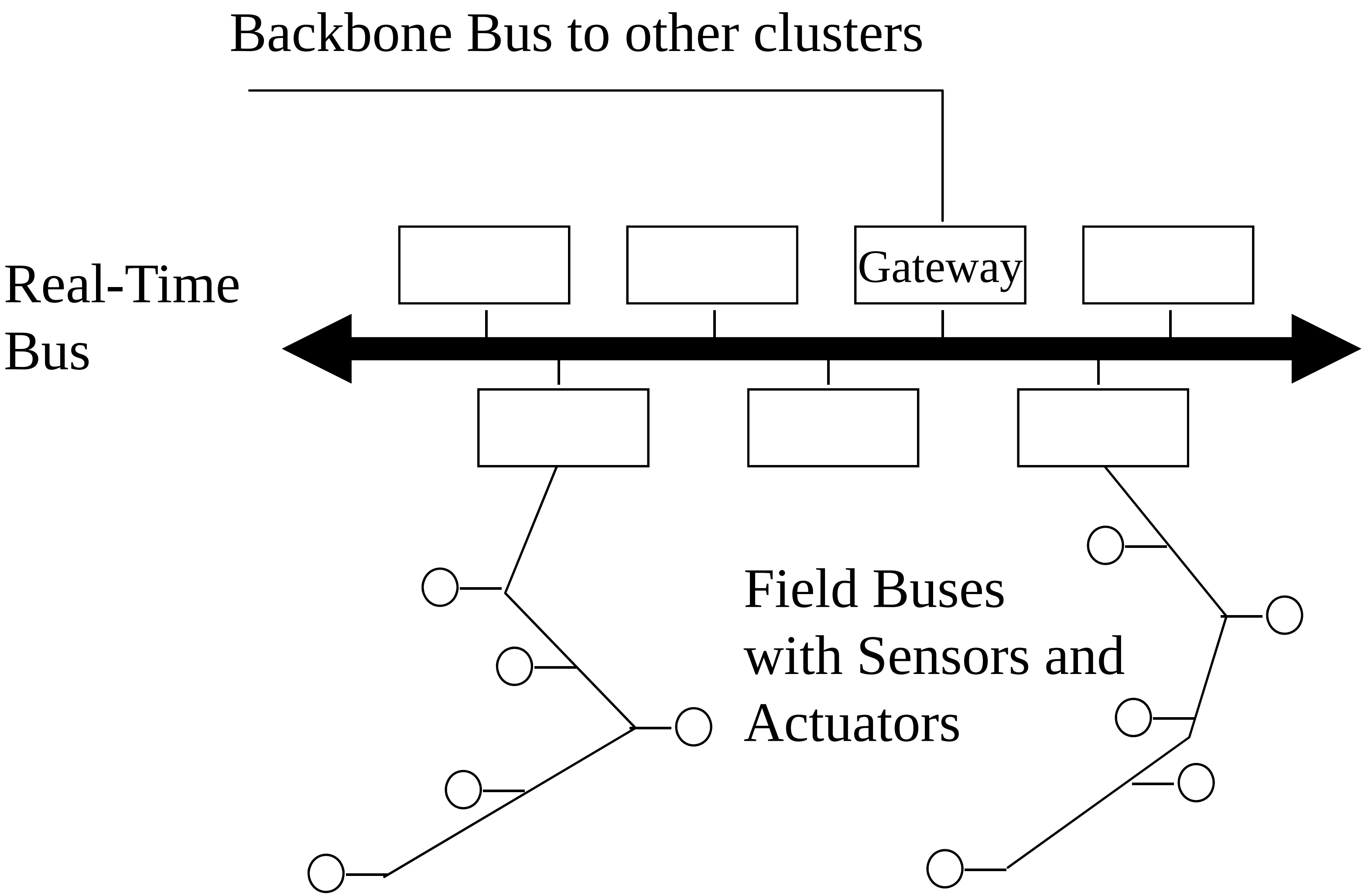
The designers assumed that the acknowledged arrival of a control output signal that is commanding the valve to close, implies that the valve is closed. Since there was an electromechanical fault in the valve, this implication was not true. A proper end-to-end protocol that mechanically senses the closed position of the valve would have avoided this catastrophic misinformation of the operator.

RT Communication Architecture/1

In the real-time community, a three level communication architecture is discussed

- ♦ Fieldbus, connecting the sensors to the nodes (real-time)
  - cheap
  - robust
- ♦ Real-time Bus, connecting the nodes within a real-time cluster (real-time)
  - fault-tolerant
- ♦ Backbone Network
  connecting the clusters for non real-time tasks (data exchange, software download, etc..)

RT Communication Architecture/2



Media Access Protocol

We distinguish the following protocol classes

- **Token**: Profibus
- **CSMA/CA** (Carrier Sense Multiple Access/Collision Avoidance): CAN
- **Minislotting**: ARINC 629, Byteflight, Flexray
- **TDMA** (Time Division Multiple Access): TTP/C, Secos, Interbus-S, Flexray
- **Central Master**: LIN, FIP, TTP/A

,

Media Access Protocol--Characteristics

	Reproduc- ibility	Temporal Encapsul.	Station Flow Control
Central Master:	yes	no	yes
Token:	yes	no	yes
CSMA/CA	no	no	no
CSMA/CD	no	no	no
Minislotting:	no	no	yes
TDMA	yes	yes	yes

Propagation Delay

The propagation delay of a channel is the time it takes for a bit to travel from one end of the channel to the other end of the channel.

It is determined by the transmission speed of an electromagnetic km/sec)

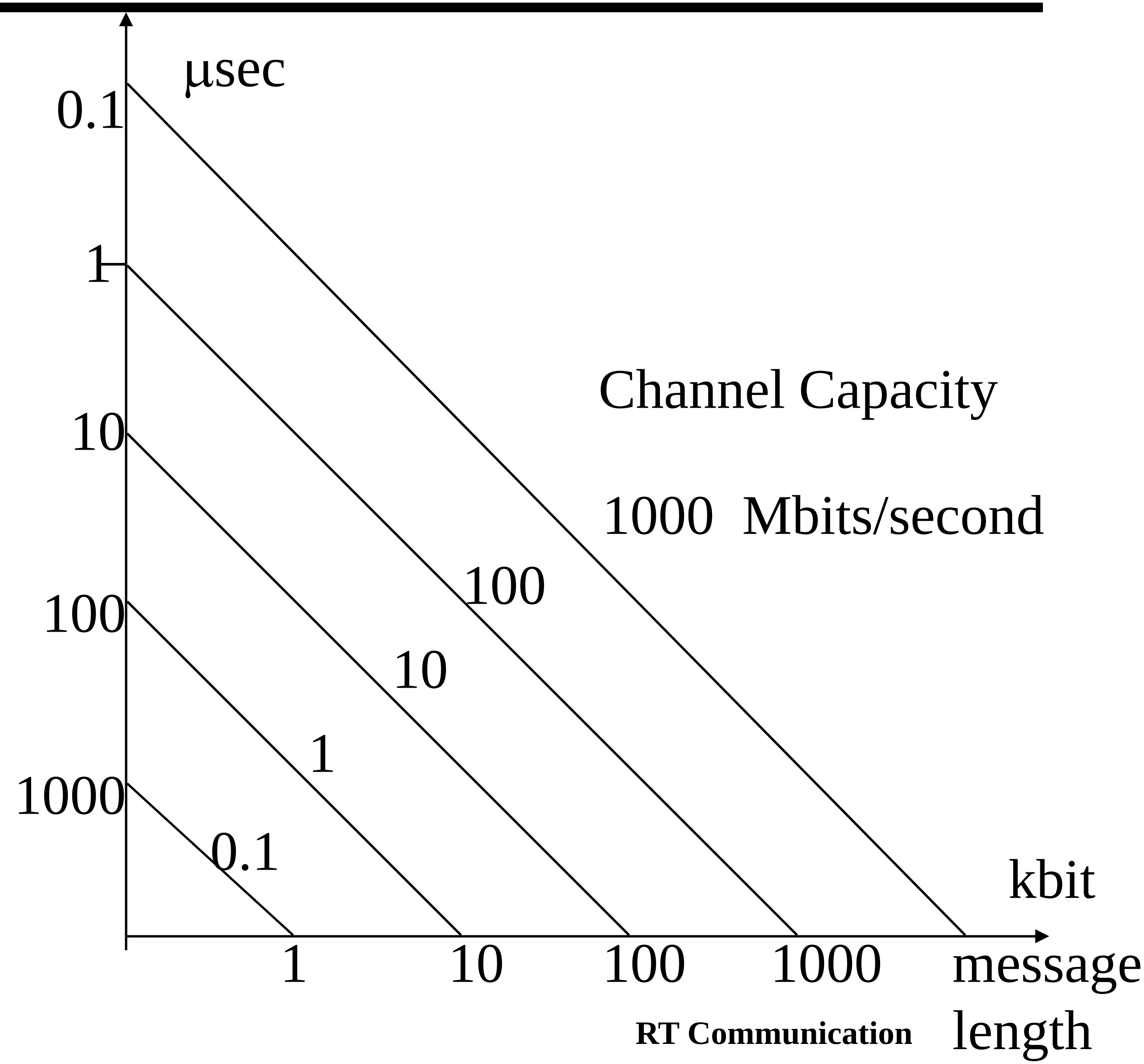
Cable: about 2/3 ns/lightfoot (200 000 km/sec)

In a channel of about 1 km length, the propagation delay is about 5 μsec.

Bandwidth

The number of bits that can traverse a channel in a second.

Bitlength of a channel: The number of bits that can traverse a channel during the propagation delay.



Limit to Protocol Efficiency

Assume a Channel with

- Bit length a
- Message length m

Then a limit to the data efficiency e of any communication protocol is given by:

$$\text{efficiency} < m / (m + a)$$

Example:

Bandwidth 100 Mbit

Channel 1 km (propagation delay 5 μsec, bitlength 500 bits)

Message length 100 bits

Limit to data efficiency: 100/(100+500) <1/6 = 16.6 %

Bitwise Arbitration

There are two states on the communication channel

- ♦ dominant and
- ♦ recessive

If two stations start to transmit at the same moment in time, then the station with a dominant bit in its arbitration field wins and the station with a recessive bit has to give in.

Assumptions:

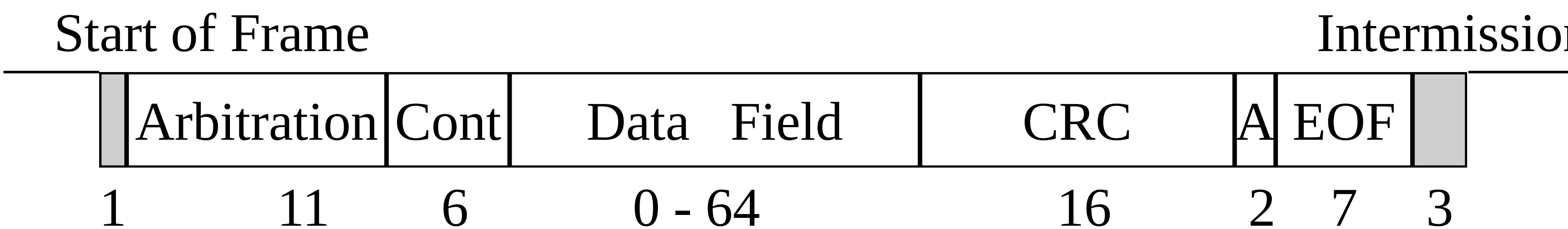
Propagation delay of the bus << length of a bitcell, since every bit has to stabilize before it can be arbitrated.

E.g. Bus length 40 m, propagation delay 200 nsec

length of a bitcell 1 μsec = 5 propagation delays!

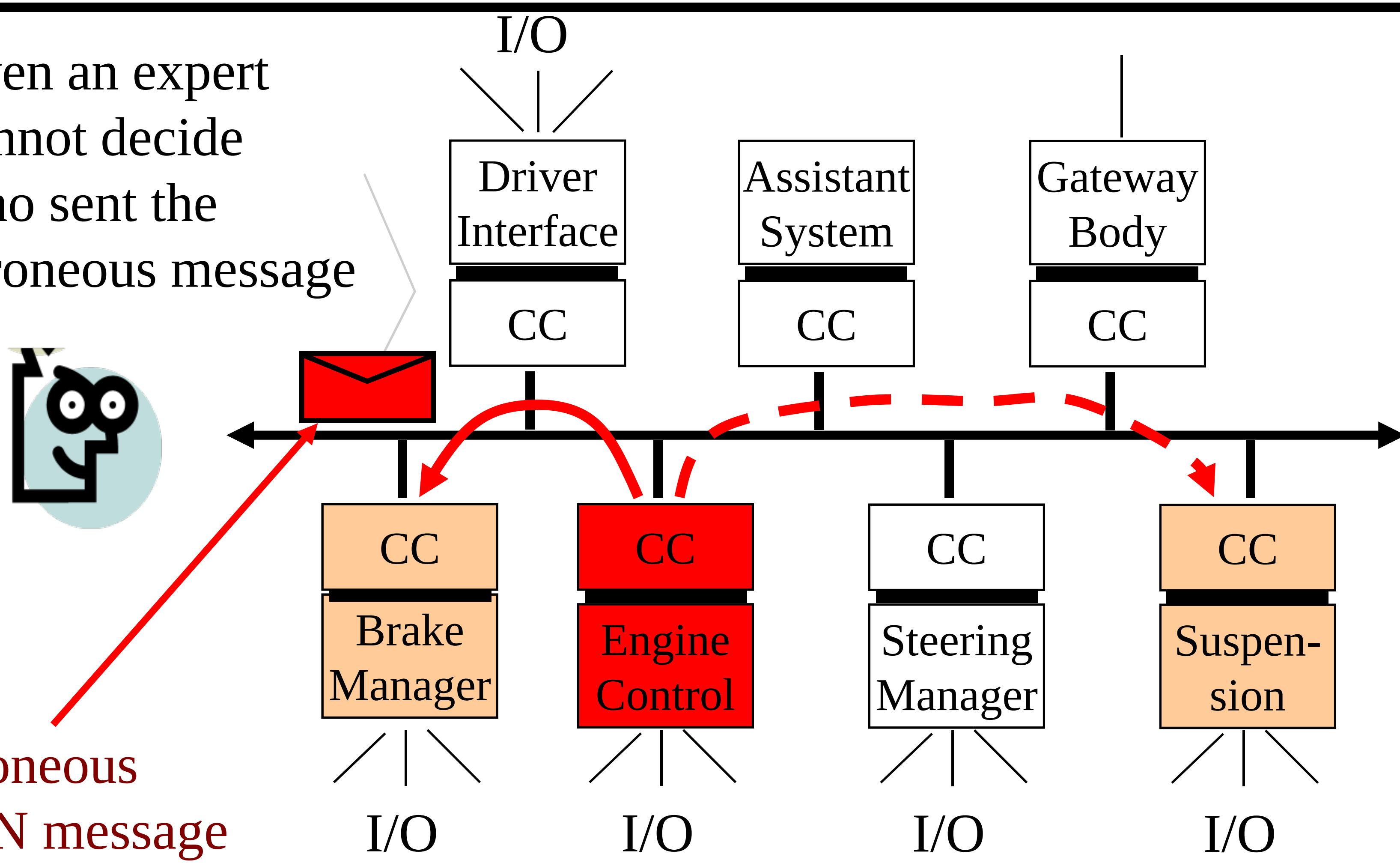
CAN - Control Area Network

A CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance)
Communication speed: 1 Mbit/second
Distance: about 40m
Standard Format: 2032 Identifiers
Extended Format: > 10^8 Identifiers
Message Format



Diagnostic Deficiency in CAN

Even an expert cannot decide who sent the erroneous message



Erroneous CAN message with wrong identifier

CC: Communication Controller

TT CAN--Time-Triggered CAN

Time-Triggered CAN is an extension to CAN that supports clock synchronization.

TT-CAN is still under development.

A “time-master” node send cyclically a synchronization frame that is used by the other nodes of the cluster to set their local time and to initiate the transmission.

In case of the failure of the time master, a local time-out is provided in each node to initiate the transmission.

Frame structure and CRC are the same as in CAN.

LIN--Local Interconnet Network

The LIN Bus is being developed by the LIN Consortium (Audi, BMW, Daimler Chrysler, Motorola, VCT, Volvo,VW) for low cost sub busses.

- Multi-master bus with startup synchronization of low-cost microcontrollers (without external oscillator)
- Uses UART/SCI Interface Hardware
- Speed <20 kbit, reaction time <100 msec, single wire implementation on enhanced ISO 9141
- Wakeup function

Source: SAE Multiplex Meeting of October 13, 1999

ARINC 629

ARINC 629 is a mini-slotting protocol that is used in the aerospace community.

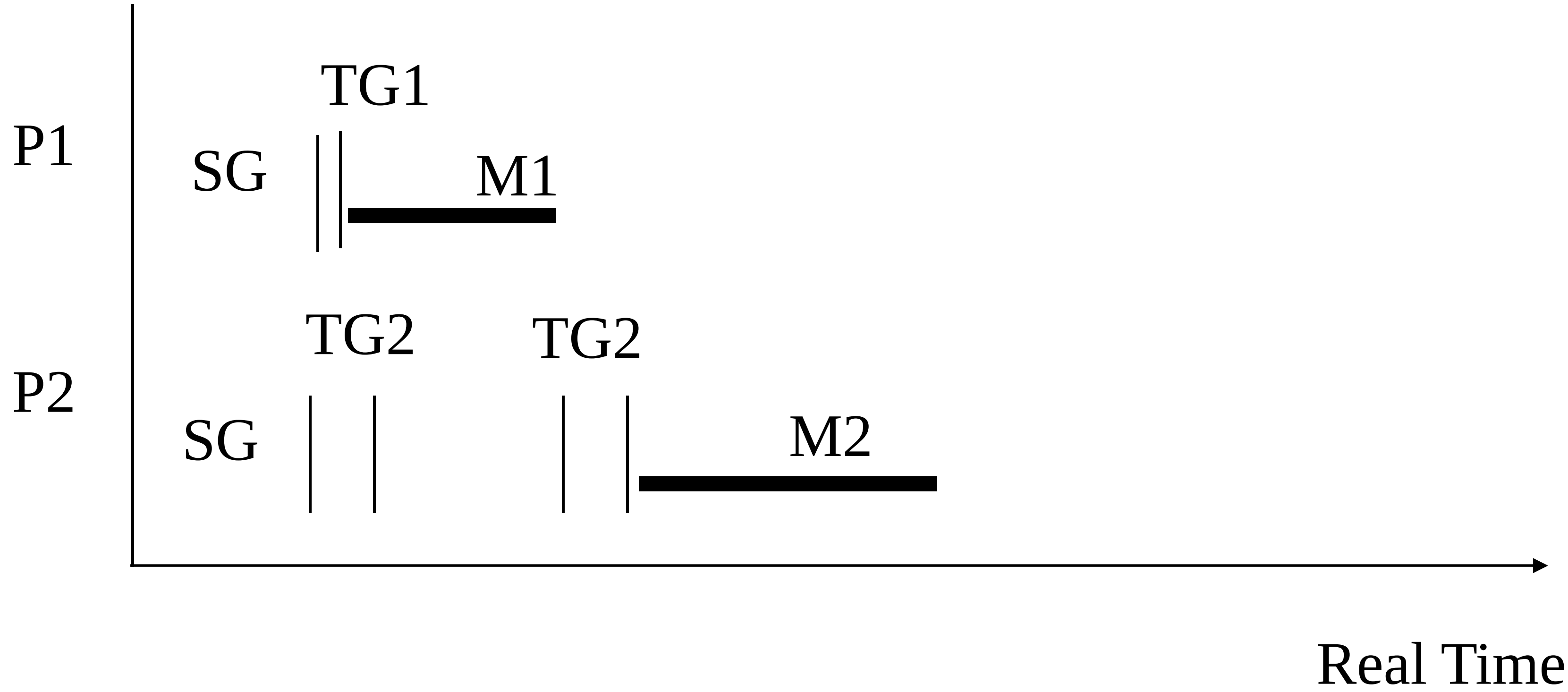
Media access is controlled by the intervals:

TG: Terminal Gap, different for every node, longer than the propagation delay of the channel

SG: Synchronization Gap, longer than longest TG

Arinc 629--Timing Diagram

Rare Event



ARINC 629 - Parameters

Typical Values for an ARINC 629 Bus with a transmit rate of 2 MHz (Basic Protocolμsec (determined by propagation delay and Terminal Number) determines minislotting interval

Sync Gap SG : Longer than the maximum TG determines formation of an epoch

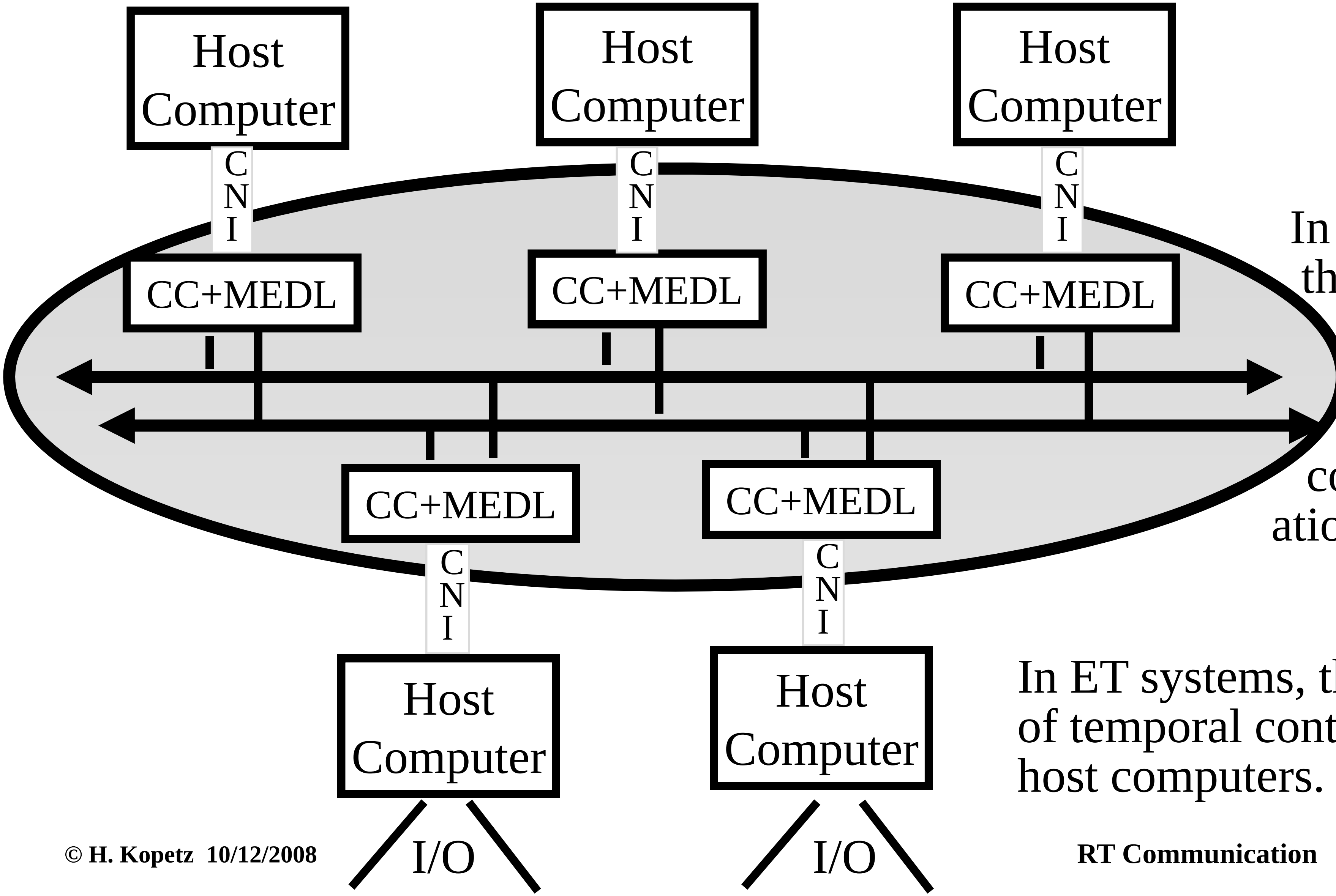
The Combined Protocol CP contains priorities for aperiodic messages.

The Time-Triggered Protocol TTP

TTP is an integrated time-triggered protocol for real-time systems that provides the following services:

- composability during system integration
- predictable transmission for all messages
- temporal encapsulation
- clock synchronization
- membership service
- temporary blackout handling
- support for mode changes
- fault-tolerance support

Global Interactions versus Local Processing



In the TTA, the locus of temporal control is in the communic-ation system.

In ET systems, the locus of temporal control is in host computers.

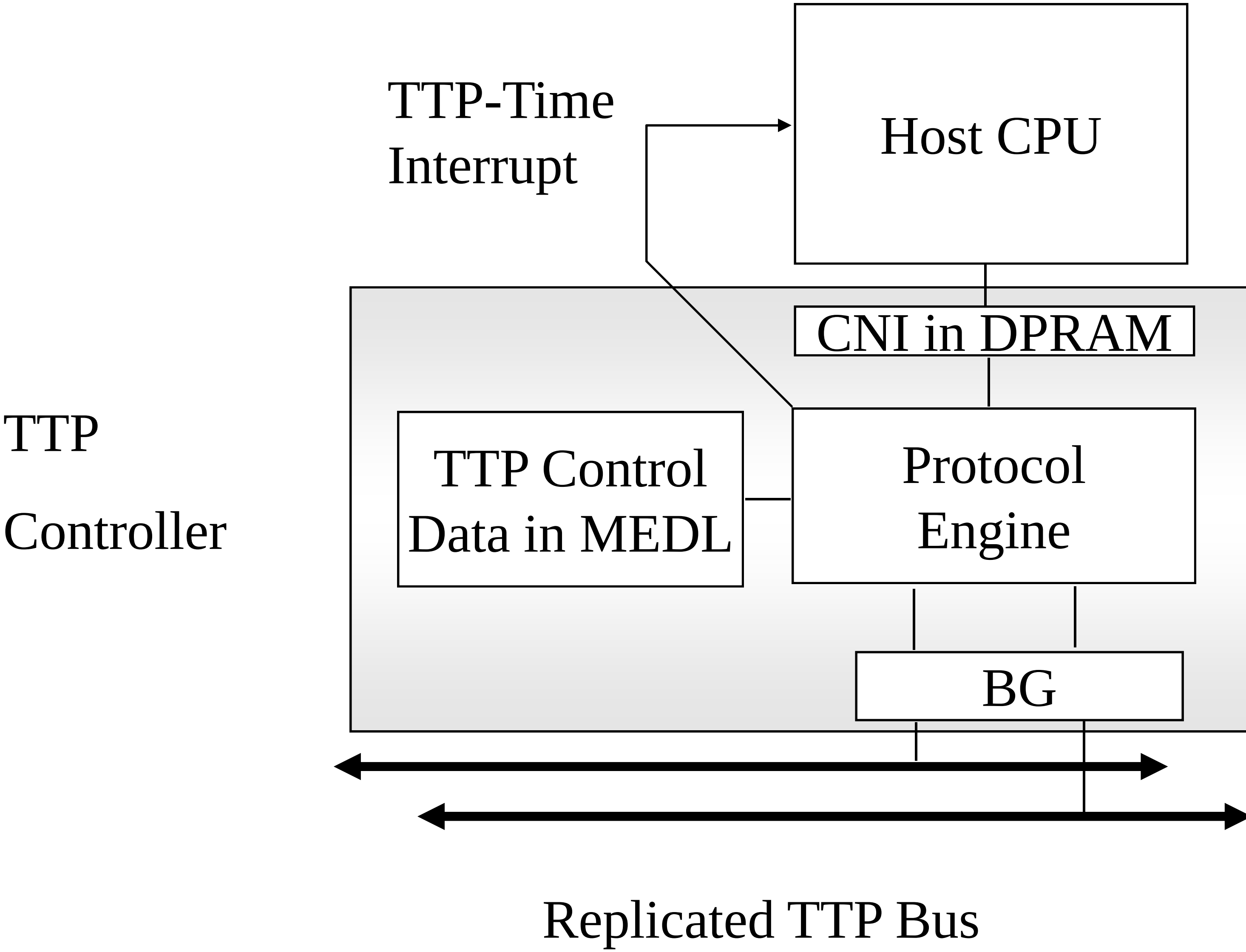
TTP - Principle of Operation

- TTP generates a global time-base
- Media access is controlled by TDMA, based on this time
- Acknowledgement implicit by membership
- Error detection is at the receiver, based on the a priori known receive time of messages
- State agreement between sender and receiver is enforced by extended CRC calculation
- Every message header contains 3 mode change bits that allow the specification of up to seven successor modes

TTP Layers

Application Software in Host Application Membership	Host Layer
	FTU CNI
FTU Membership Permanence of Messages	FTU Layer
	Basic CNI
Start up (Cold and Warm) Redundancy Management (RM)	RM-Layer
SRU Membership Clock Synchronization Implicit Acknowledgment	SRU Layer
Media Access: TDMA Bit Synchronization Bit Encoding/Decoding: MFM Code	Data Link/Physical Layer

TTP-Controller



Use of Apriori Knowledge

The a priori knowledge about the behavior is used to improve the Error Detection: It is known a priori when a node has to send a message (*Life sign for membership*).

- ♦ Message Identification: The point in time of message transmission identifies a message (*Reduction of message size*)
- ♦ Flow control: It is known a priori how many messages will arrive in a peak-load scenario (*Resource planning*).

For event-triggered asynchronous architectures, there exists an impossibility result: *'It is impossible to distinguish a slow node from a failed node!'* This makes the solution to the membership problem very difficult.

Fail-Silent Nodes

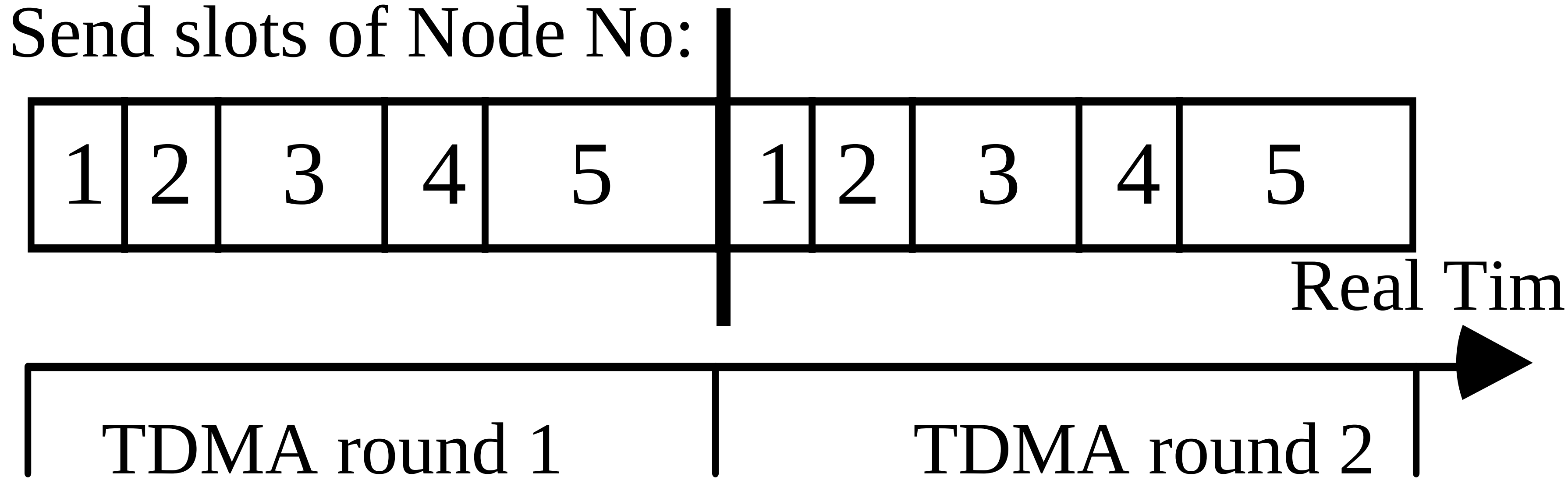
Error Containment and Fault Management are simplified and accelerated, if the nodes of a distributed system exhibit "clean" failure modes.

If a node sends either correct messages (in the value and time domain) or detectably incorrect messages in the value domain, then the failure mode is clean, i.e., the node is fail-silent.

TTP is based on a two level approach:

- ♦ At the architecture level the fault management is based on the assumption that all nodes are fail-silent
- ♦ At the node level mechanisms are provided that increase the error detection coverage to justify the fail-silent assumption.

TDMA sequence



The global time is established by TTP using a fault-tolerant clock synchronization algorithm.

TTP Message Format on the Network

Header	Data Bytes (up 240)	24 bit CRC

I/N Mess.	Mode bit 1	Mode bit 2	Mode bit 3

Excluding the intermessage gap, the overhead of a TTP frame is 32 bits--No identifier field is required, since the name of a message is derived from the time of arrival.

Continuous State Agreement

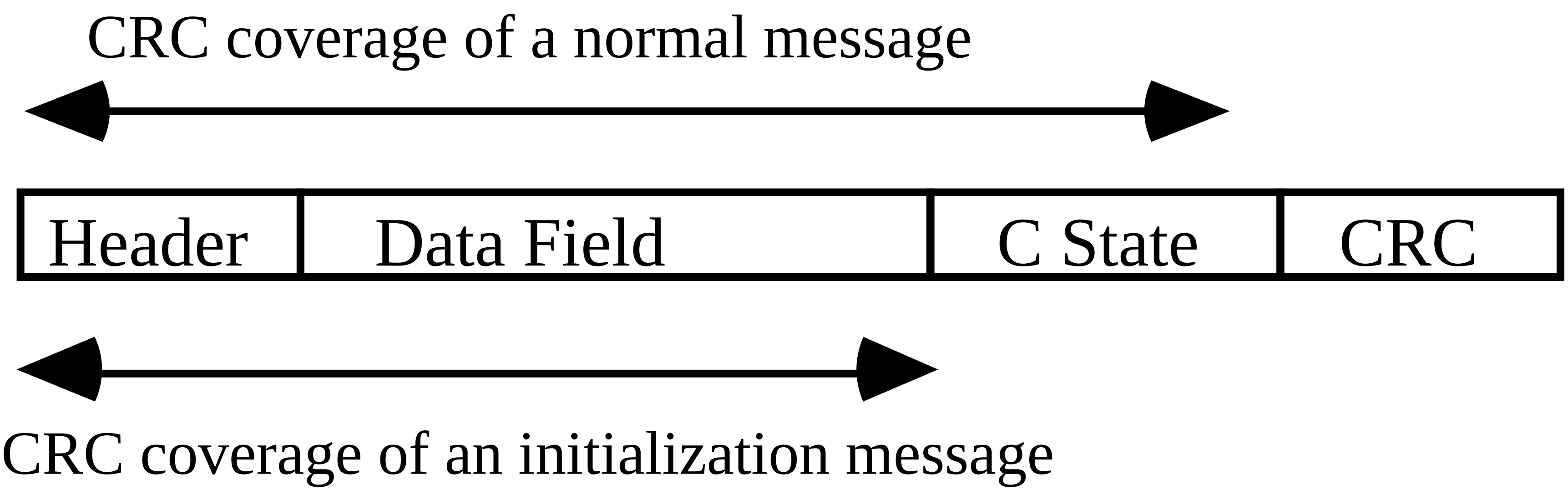
The internal state of a TTP controller (C-state) is formed by the

- Time
- Operational Mode, and
- Membership

The Protocol will only work properly, if sender and receiver contain the same state.

Therefore TTP contains mechanisms to guarantee continuous state agreement (extended CRC checksum) and to avoid clique formation (counts of positive and negative CRC checks).

CRC Calculation in TTP



C -State: Time, Membership Vector, and MEDL Position at sender respectively at receiver

Clock Synchronization in TTP

The expected arrival time of a message is known a priori,

The actual arrival time of a message is measured by the controller.

The difference between the expected and the actual arrival time is an indication for the deviation between the clock of the sender and the clock of the receiver.

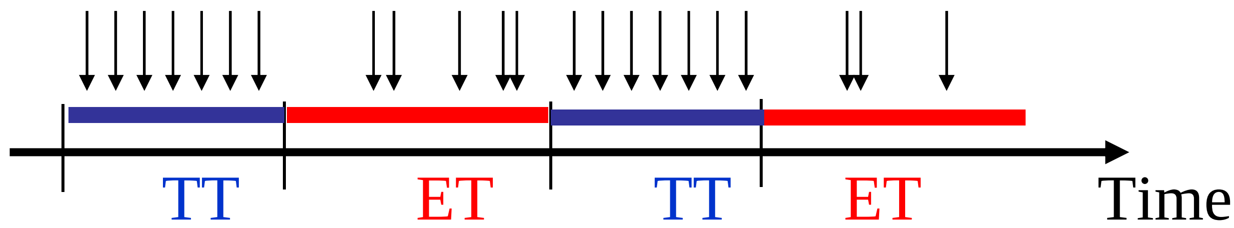
These differences are used by the FTA clock synchronization algorithm to periodically adjust the clock of each node.

There is no extra message or no special field within the message required to achieve this fault-tolerant clock synchronization.

Integration of TT and ET Services--the Options

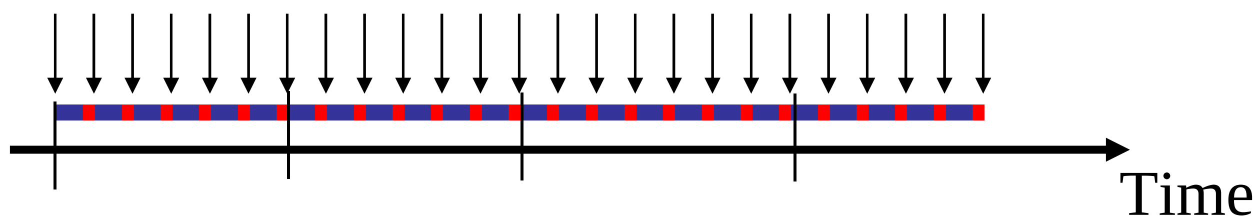
(i) **Parallel**: Time Axes is divided into two parallel windows, where one window is used for TT, the other for ET, Two media access protocols needed, one TT, the other ET

Loss of Temporal Composability



(ii) **Layered**: ET service is implemented on top of a TT protocol Single time triggered access media access protocol.

Loss of Global Bandwidth Sharing



What are the consequences for global time and state?

FlexRay

FlexRay is a time-triggered protocol that has been designed by the automotive industry for automotive applications within a car:

Combination of two protocols

ET: mini-slotting, similar to ARINC 629

TT: TDMA, similar to TTP

Distributed clock synchronization

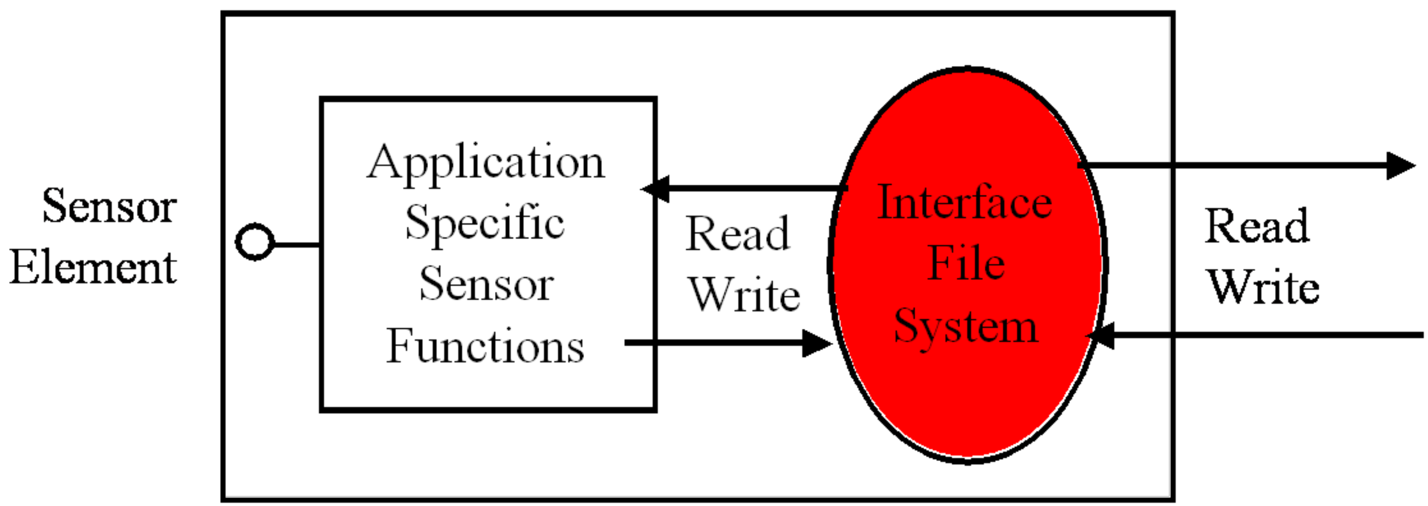
At present no membership

TTP-A Objectives

- Composability and Testability
- Latency Guarantee for State Estimation
- Good Error Detection for fail safe operations
- Use of Standard UARTS (8 data bits with parity)
- High Data Efficiency (>50 %) and small latency
- Single wire (10 kbits) or twisted pair operation
- Clock Synchronization better than 1 msec

TTP/A Sensor Bus

- Low Cost Time-Triggered Sensor Bus to provide a uniform interface to the different types of smart transducers
- Has been standardized by the OMG in January 2003
- Optimized for 8 bit microcontrollers: requires in its minimum version less than 4 kbyte of ROM and 64 bytes of RAM
- Central to TTP/A is the concept of an interface file system (IFS)



Purpose of TT Ethernet

The *purpose* of TT Ethernet is to provide a *uniform* communication system for all types of distributed non-real-time and real-time applications, from very simple uncritical data acquisition tasks, to multimedia systems and *up to safety-critical control applications*, such as *fly-by-wire* or *drive-by wire*.

It should be possible to upgrade an application from standard TT- Ethernet to a safety-critical configuration with minimal changes to the application software.

Certification as a Design Driver

Certification is only possible if a system has been *designed for certification:*

- Independence of Fault-Containment Units (FCU)
- Elimination of *temporal error propagation* from one FCU to the network and in consequence to the other FCUs.
- Deterministic Operation
- Formal Analysis of Critical Algorithms
- Modular Composition of Correctness Argument

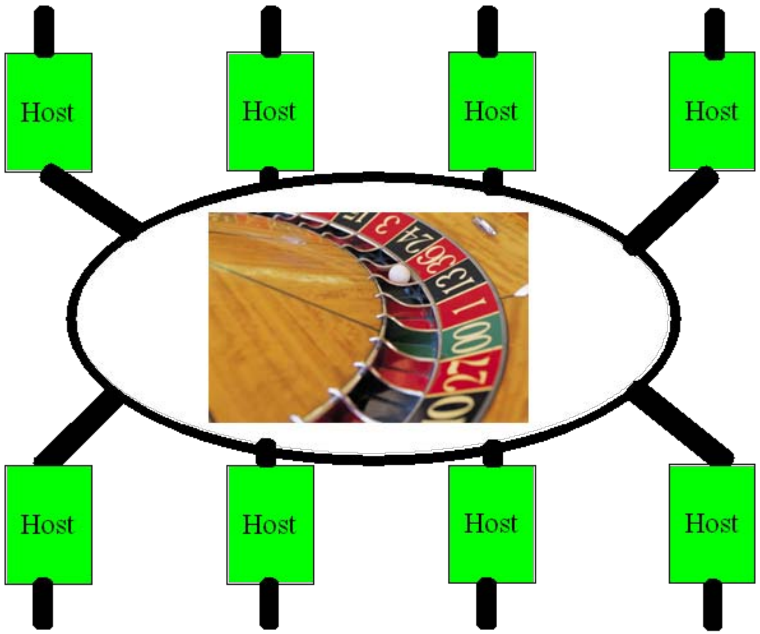
Legacy Integration

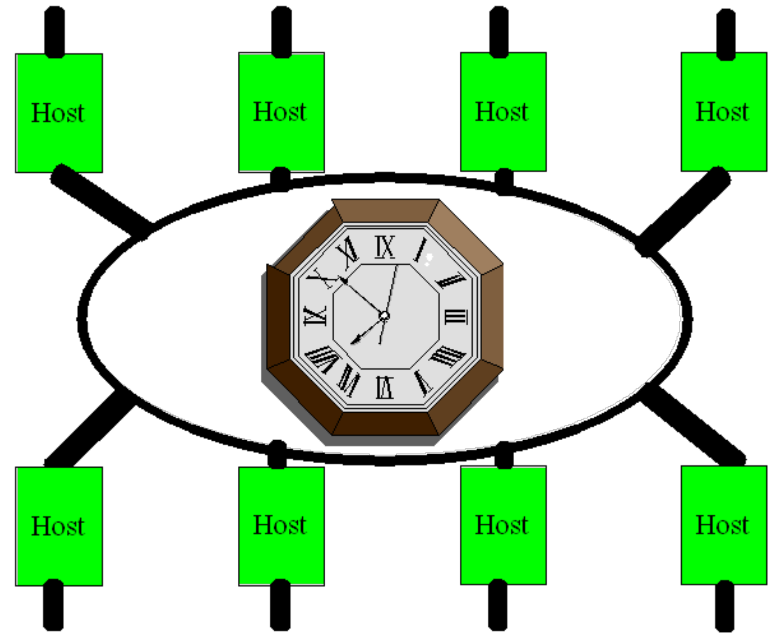
TT-Ethernet is required to be fully compatible with existing Ethernet systems in hardware and software:

- ♦ Message format in full conformance with Ethernet standard
- ♦ Standard Ethernet traffic must be supported in all configurations
- ♦ Existing Ethernet controller hardware must support TT Ethernet traffic.
- ♦ IEEE 1588 standard for global time representation is supported

The Key Decision- *Two Message Categories*

Competition vs. Cooperation





Internet Model
Standard (ET) Ethernet

Embedded System Model
TT Ethernet

Distinguish between two Categories of Messages

ET-Messages:

- ♦ Standard Ethernet Messages
- ♦ Open World Assumption
- ♦ No Guarantee of Timeliness and No Determinism

TT-Messages:

- ♦ Scheduled Time-Triggered Messages
- ♦ Closed World Assumption
- ♦ Guaranteed *a priori* known latency
- ♦ Determinism

TT and ET Ethernet Message Formats are Alike

Field	
Preamble (7 bytes)	Standard Ethernet Message Header
Start Frame Delimiter (1 byte)	
Destination MAC Address (6 bytes)	
Source MAC Address (6 bytes)	
Tag Type Field (***88d7*** if TT)	
Client Data (0 to n bytes)	
PAD (0 to 64 bytes)	
Frame Check Sequence (4 bytes)	

Conflict Resolution in TT Ethernet

- **TT versus ET:** TT message wins, ET message is interrupted (preempted). The switch will retransmit the preempted ET message autonomously
- **TT versus TT**: Failure, since TT messages assumed to be properly scheduled (closed world system)
- **ET versus ET:** One has to wait until the other is finished (standard Ethernet policy).

There is no guarantee of timeliness and determinism for ET messages!

Global Time

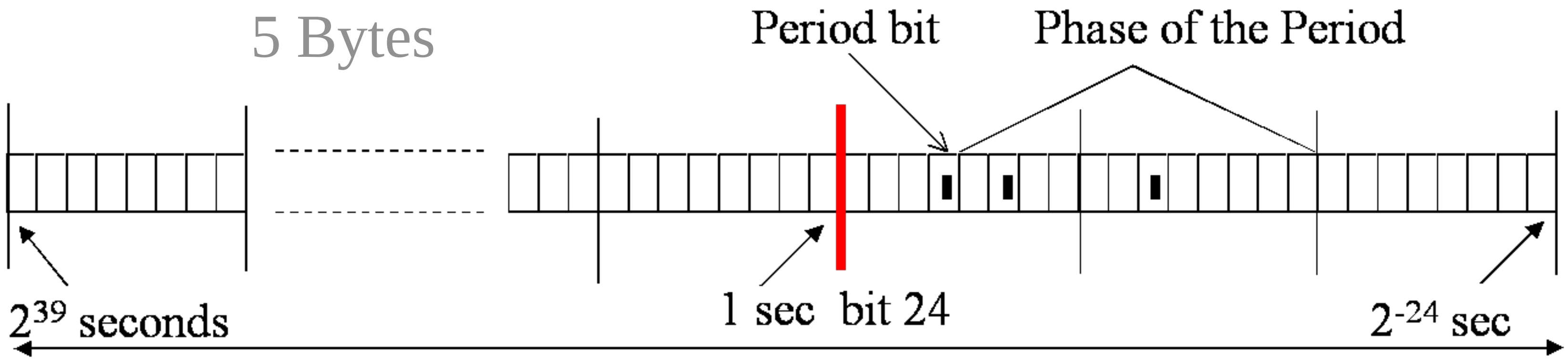
- TT Messages are used to build a global time base
- TT Ethernet time format is a *sparse binary time format*. Fractions of a second are represented as 24 negative powers of two (down to about 60 nanoseconds), and full seconds are presented in 40 positive powers of two (up to about 30 000 years) of the physical second.
- This binary time-format has been standardized by the OMG and IEEE 1588.
- TT Ethernet gives the user the option to make a tradeoff between dependability and cost of the global time.

TT Ethernet Periods

- The TT Ethernet *recommends* to restrict the period durations to the positive and negative *powers of two* of the second, i.e. a period can be either 1 second, 2 seconds, 4 seconds, and so forth, or 1/2 second, 1/4 second, 1/8 second and so forth.
- The duration of each period can then be characterized by the corresponding bit (*period bit*) in the binary time format.
- The *phase of a period*, i.e. the *offset* to the start instant of the selected duration in the global time format, is designated by the specification of a pattern of twelve bits (*the phase bits*) to the right of the *period bit*.

  We then can represent a cycle with two Bytes (*four period bits* i.e. 16 periods, and *twelve phase bits*).

TT Ethernet Periods--Example



Specification of a period of $1/2^4$ (i.e 1/16) second with a phase (i.e. the offset from the periodic 1/16 second instant) of $1/2^6+1/2^{11} =$ 16113 μseconds.

TT Ethernet Protocol Family

TT Ethernet forms of an upward compatible family of protocols, starting with low-cost low-function controllers and going up to safety critical configurations with fault-tolerant time base, supported by certification:

- ♦ *Low-level* TT Ethernet system which is not time-aware and provides no or minimal error containment.
- ♦ *Professional* TT Ethernet system which is time-aware and contains configuration state to perform error containment of failing nodes.
- ♦ *Advanced* TT Ethernet system with multiple switches that supports fault-tolerant clock synchronization and triple modular redundancy.