Real-Time Communication

slide credits: H. Kopetz, P. Puschner

Overview

- Communication system requirements
- Controlling the flow of messages
- Types of protocols
- Properties of communication protocols
- Protocol examples

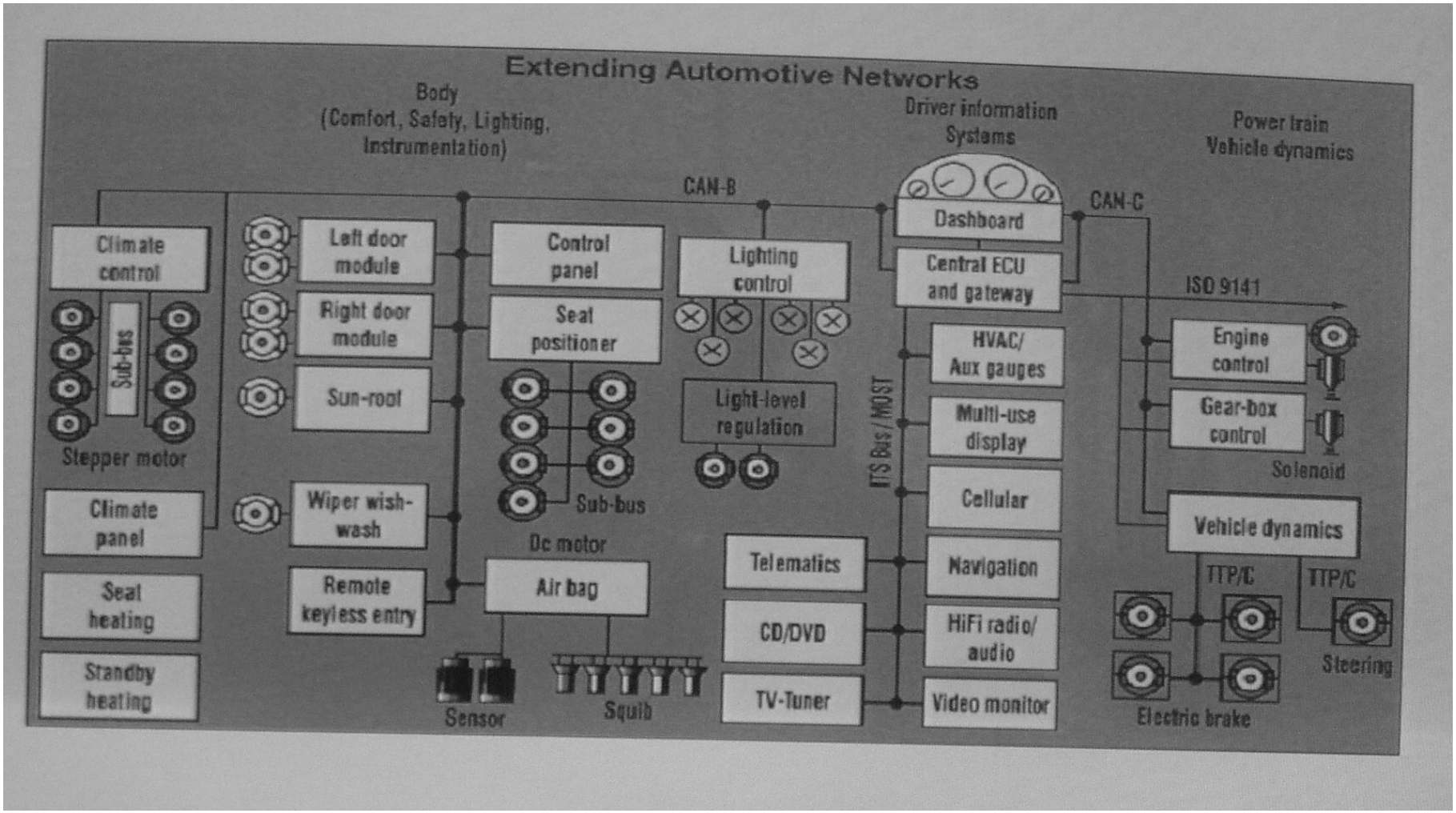
Importance of Distributed RTS

Reasons for distributedness:

- *Composability*: 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*: safety critical applications

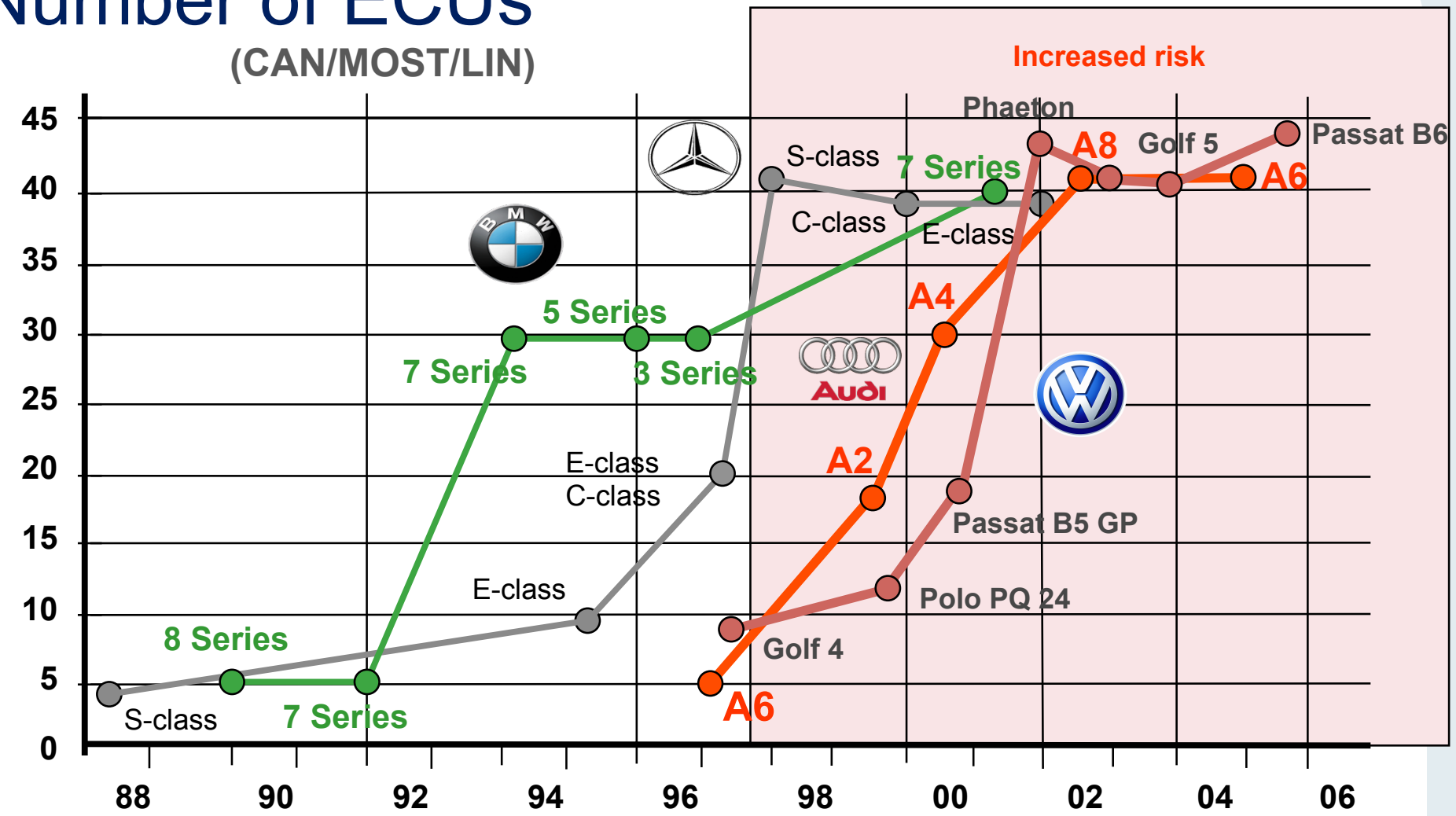
⇨ **Proper real-time communication is of central importance**

Example: Car Networks



Source: R. Basserone, R. Marculescu, Communication/Component Based Design, 6/2002

Number of ECUs



Source: Prof. Dr. Jürgen 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:

- 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.

Need for Protocol Consolidation

Technological and economic development needs:

- Cost of design and mask generation of an SoC is more 10 Million Dollars ➪ must be amortized over each hardware protocol implementation.
- Interoperability requirements ➪ protocol compatibility.
- Every unique protocol requires a unique set of software modules and development tools that must be developed and maintained.
- Reduction of human resource cost for learning/gaining experience with new protocols.

Properties of Successful Protocols

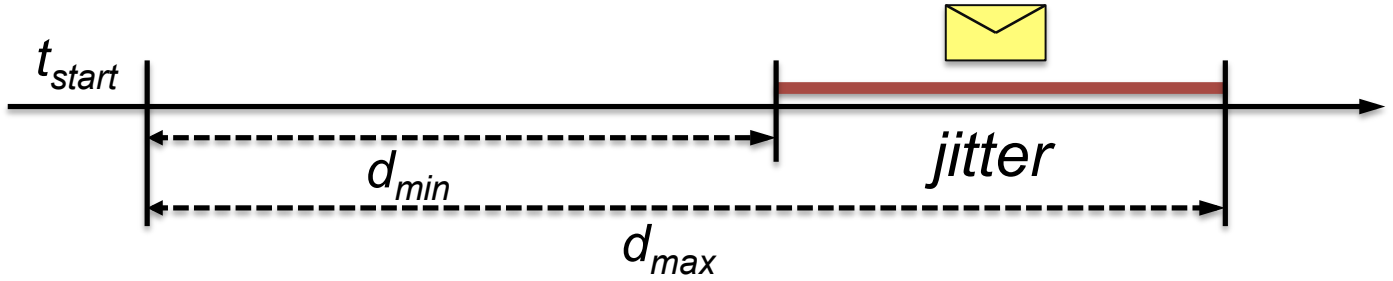
- Sound theoretical foundations w.r.t. time, determinism, security, and composability.
- Support for all types of real-time applications, from multimedia to safety-critical control systems.
- Support error containment of failing nodes
- Economically competitive – a hardware SoC protocol controller should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

Message

Atomic data structure transferred from sender to one or more receivers (multicast, broadcast)

Transmission timing

- t_{start} … start instant at sender
- d_{min} … minimum transmission delay
- d_{max} … maximum delay
- $[t_{start} + d_{min}, t_{start} + d_{max}]$ … interval of receive instants
- $d_{max} - d_{min}$ … *jitter* of the transmission channel



Flow Control

… governs the flow of information between communicating partners

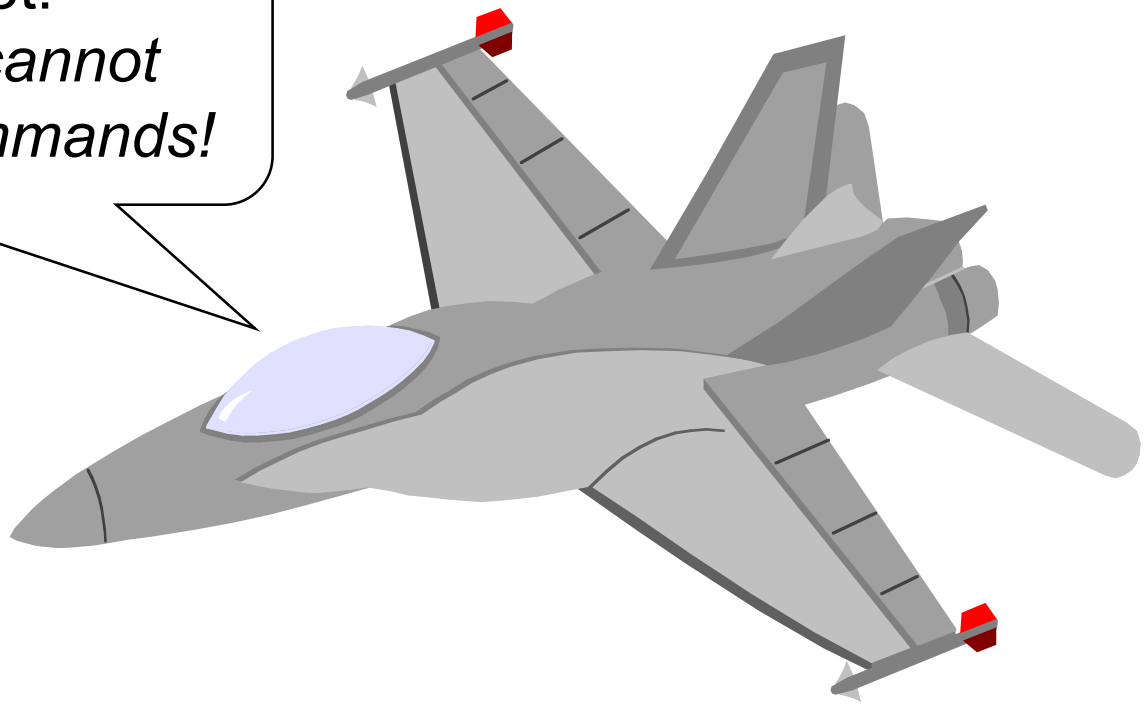
- The sender must not outpace the receiver, therefore
- The processing speed of the receiver should determine the pace of communication

Explicit Flow Control

- Sender (1) transmits a message to the receiver and (2) waits for an acknowledgement of receipt from the receiver.
- Receiver is authorized to slow down the sender (back-pressure flow control), i.e., the *sender is in the sphere of control of the receiver*.
- Error detection by sender.
- Missing acknowledgement of a message implies
    - Message loss,
    - Receiver is late, or
    - Receiver has failed.

Example: Explicit Flow Control

Computer to Pilot:
Please fly slower, I cannot keep up with your commands!



Implicit Flow Control

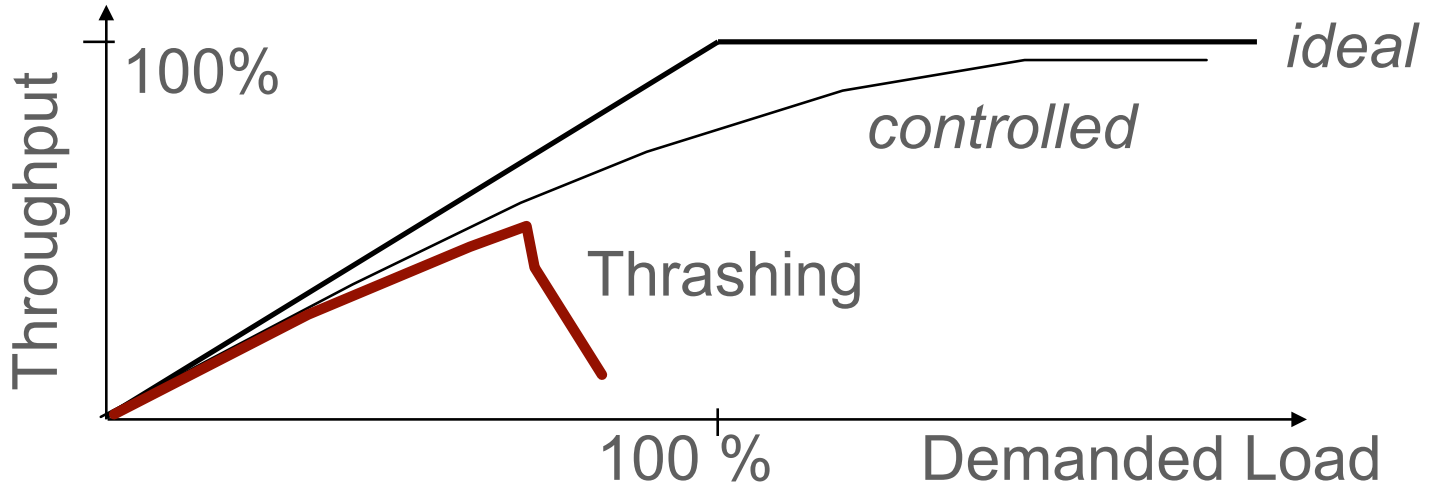
Sender and receiver agree a priori, i.e., before runtime, on the rate at which the sender will transmit messages.

- Agreed rate must be manageable by receiver.
- Error detection by receiver.
  - ⇨ no message acknowledgement.
  - ⇨ unidirectional use of communication channel.
- Well suited for multicast communication.

Explicit Flow Control – Thrashing

Thrashing under high-load conditions

- Collisions
- Message delays lead to timeouts / re-sending of messages
- Buffer overflows cause message loss and re-send
  - Traffic increase at worst possible time



RT Communication-System Needs

Predictable communication service for real-time data

- *Determinism*
  - Timeliness
  - Low complexity
  - Testing
  - Active Redundancy (e.g., TMR)
  - Certification
- *Multicast* – independent non-intrusive observation, TMR
- *Uni-directionality*: separate *communication – computation*

RT Communication System Needs (2)

Flexible best-effort communication service for the transmission of non-real-time data coming from an open environment

Support for streaming data

Dependability

Limits in RT-Protocol Design

- Temporal guarantees
- Synchronization domain
- Error containment
- Consistent ordering of events

Temporal Guarantees

Impossibility result: we cannot give tight bounds on communication times in an open communication scenario

> All autonomous senders may start sending a message to the same receiver at the same time (critical instant), thus overloading the channel to the receiver.

Traditional strategies to handle overload:

- Store messages temporarily
- Delay sending of message (back pressure protocol)
- Discard some messages

None of these strategies is suited for real-time data!

Temporal Guarantees (2)

Senders of real-time data have to coordinate their sending actions to avoid channel conflicts.

- Construct a conflict-free sending schedule for real-time messages
- Use a common time base as time reference for a-priori agreed sending actions

Synchronization Domain

It is impossible to support more than a *single coordination domain* for the temporal coordination of components in a real-time system.

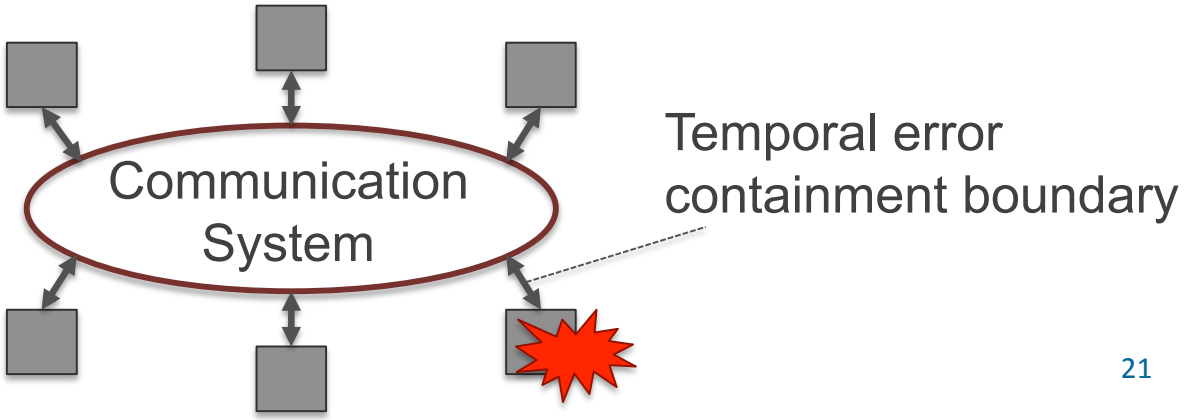
The synchronization can be established by:

- Reference to a *single global time base*
- Reference to a *single leading data source* (coordinator)

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*.



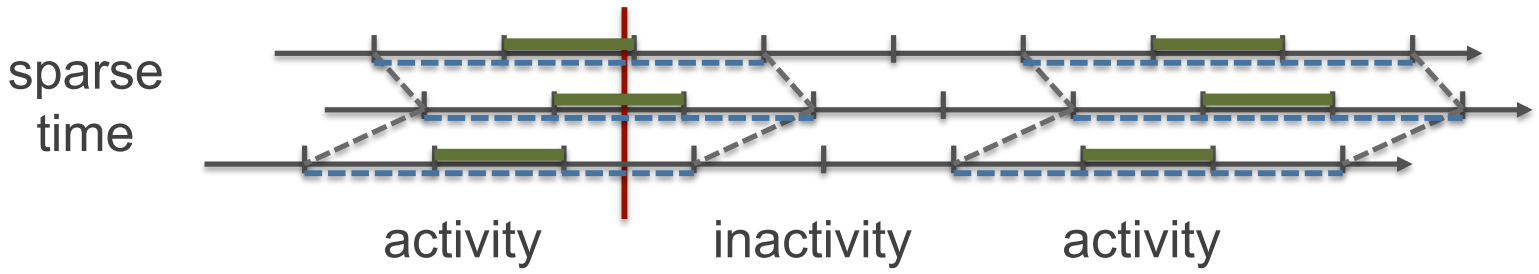
Consistent Ordering of Events

Sparse global time base for

- Correct ordering of sparse events
- Consistent time-stamping of sparse events
- Correct resolution of simultaneity

Generation of *sparse events*

- Computer system generates sparse events
- Environment events ➪ agreement protocol to map dense events to sparse time intervals



Protocol Categories

- Event-triggered (ET) protocols
- Rate-constrained (RC) protocols
- Time-triggered (TT) protocols

Event-Triggered (ET) Protocols

- Event at sender triggers protocol execution at arbitrary point in time.
- Maximum execution time and reading error of the protocol are large compared to the average execution time.
- Error detection is by sender.
- Error detection needs an acknowledgement. This creates correlated traffic in a multicast environment.
- No temporal encapsulation.
- Explicit flow control to protect the receiver from information overflow. Sender in sphere of control of receiver.

Examples: CSMA/CD, CAN

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 a 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

$d_{min} = 0$

$d_{max} = (2k + 1)$ TRT = 5 TRT

Maximum action delay = 10 TRT (100 msec)

In OSI implementations PAR protocols are stacked!

Rate-Constrained (RC) Protocols

- Provide minimal guaranteed bandwidth.
- The message rate of the sender is bounded by the communication system.
- Temporal guarantees (maximum latency) for message transport, as long as the guaranteed bandwidth is not exceeded
  ➪ sender better obeys contract.
- No global time or phase control possible.

Examples: Token protocol, AFDX, AVB (TSN)

RC Protocols: Traffic Shaping/Policing

Enforce traffic compliance to a given profile (e.g., rate limiting)

By delaying or dropping certain packets, one can
(i) optimize or guarantee performance,
(ii) improve latency, and/or
(iii) increase or guarantee bandwidth

for other packets

Traffic shaping: delays non-conforming traffic

Traffic policing: drops or marks non-conforming traffic

Traffic Shaping

Traffic metering: check compliance of packets with traffic contract

Impose limits on bandwidth and burstiness

Buffering of packets that arrive early

- Buffer dimensioning (?)

Strategy to deal with full buffer

- Tail drop (→ policing)
- Random Early Discard
- Unshaped forwarding of overflow traffic

Traffic Shaping (2)

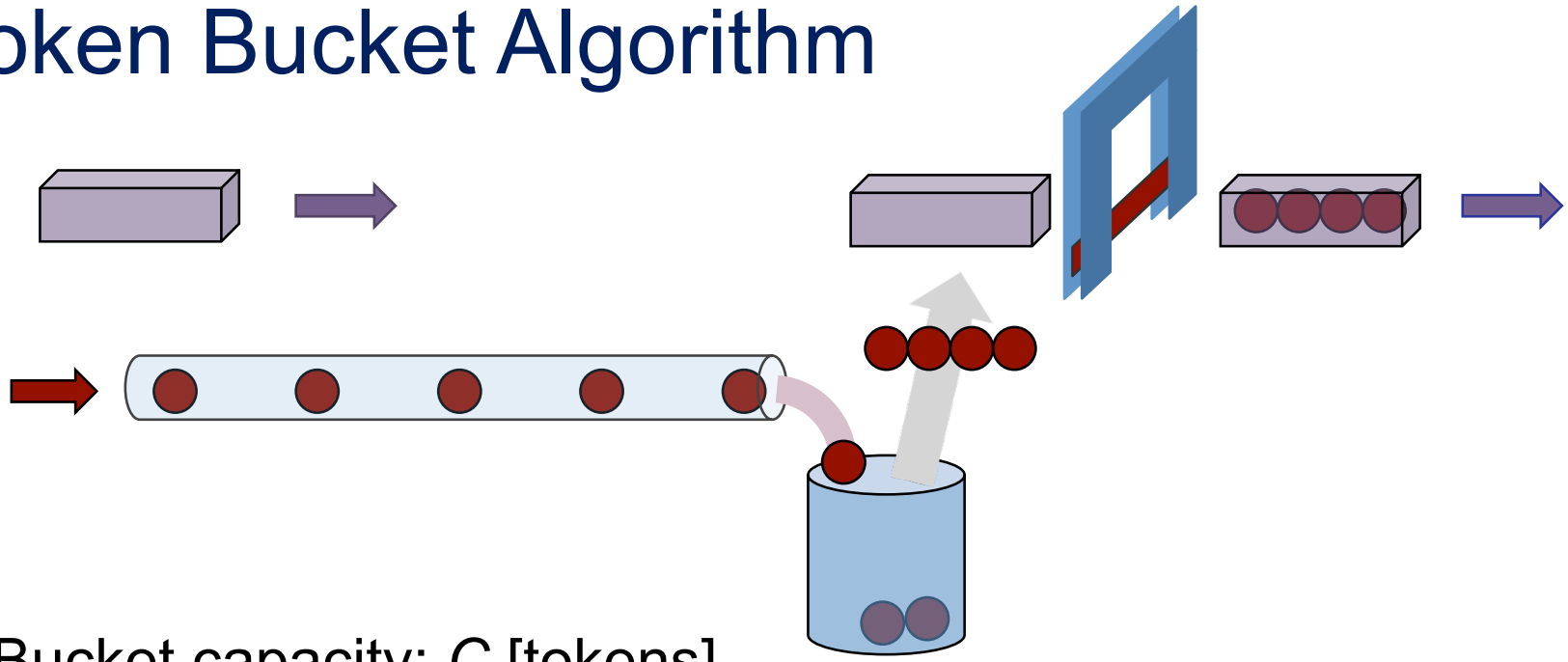
Traffic shaped by

- Self limiting sources
- Network switches

Shaping effect

- Shaping traffic uniformly by rate
- More sophisticated characteristics (allow for defined variability in traffic)

Token Bucket Algorithm



Bucket capacity: *C* [tokens]

Token arrival rate: *r* [tokens per second]

When a packet of *n* bytes arrives, *n* tokens are removed from the bucket and the packet is sent

If fewer than *n* tokens available, no token is removed and the packet is considered to be non-conformant

Time-Triggered (TT) Protocols

- Progression of global time triggers protocol. The point in time when a message is sent is *a-priory* known to all receivers.
- Maximum execution time is about the same as average execution time. Therefore small reading error (jitter).
- Error detection by receiver, based on a priori knowledge.
- The protocol is unidirectional, well suited for a multicast environment.

Event Message vs. State Message

Characteristic	*Event Message*	*State Message*
Example of message contents	“Valve has closed by 5 degrees”	“Valve position is 60 degrees”
Contents data field	Event information	State information
Sending instant	After event occurrence	Periodically at a-priory defined points in time
Temporal control	Interrupt caused by event occurrence	Sampling, triggered by progression of time
Handling at receiver	Queued and consumed on reading	New version replaces old one; no consumption
Semantics at receiver	Exactly once	At least once

Event Message vs. State Message (2)

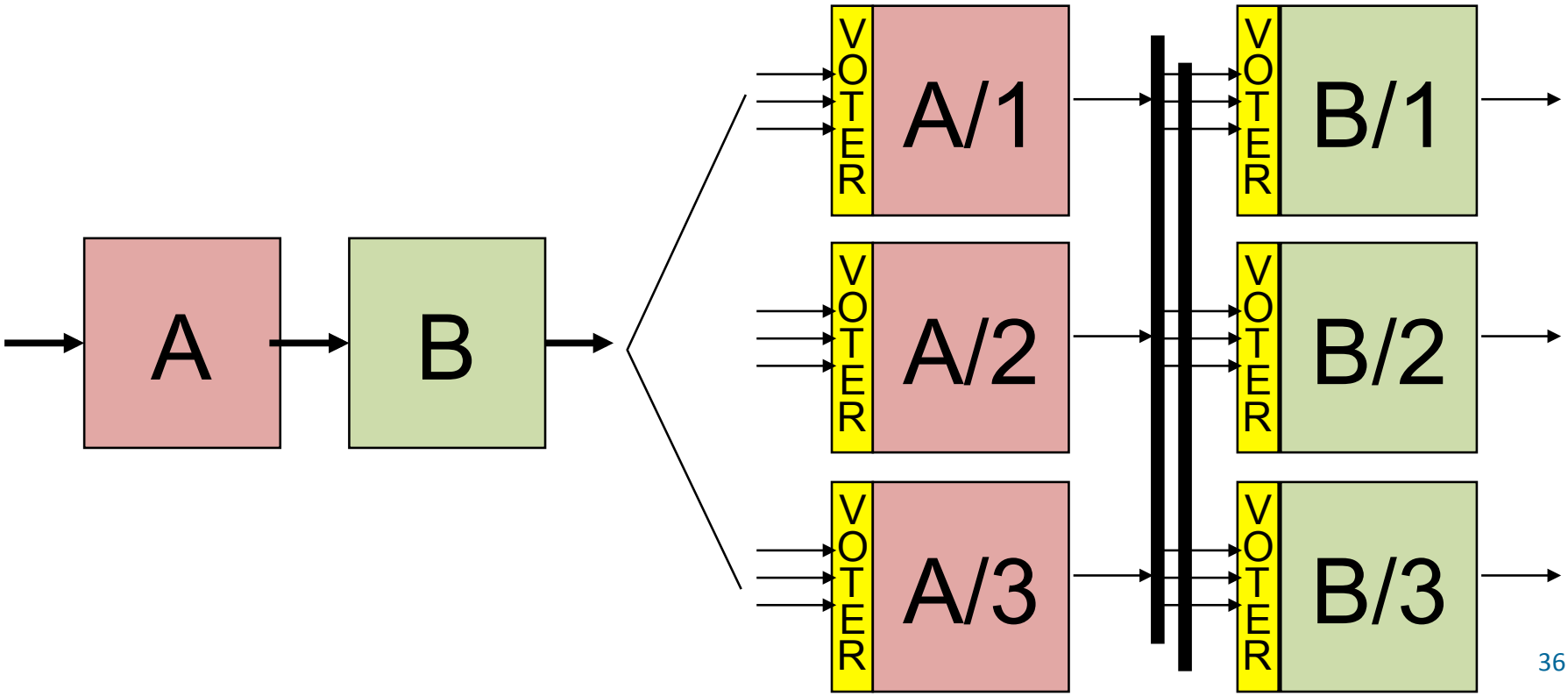
Characteristic	*Event Message*	*State Message*
Idempotence	no	yes
Consequences of message loss	Loss of state synchronization of sender and receiver	State information is not available for one sampling interval
Typical comm. protocol	Positive acknowledgement or retransmission (PAR)	Unidirectional datagram
Typical comm. topology	Point-to-point	Multicast
Load on comm. system	Depends on rate of event occurrences	Constant

Fault Tolerance and TT Communication

- Idempotence of messages supports active redundancy
- Message broadcast supports transparent TMR
- Detection of message loss based on a-priory schedule
- Broadcast of g-state supports re-intgration of components
- Regular, a-priory known transmission pattern supports error containment in the time domain (e.g., avoid that babbling idiot monopolizes the communication medium)

Mitigation of Node Failures by TMR

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



Reliable Communication is Not Enough

Successful message delivery

- Indicates successful command delivery
- does not guarantee correct service provision

Subsystems other than the communication system may fail (e.g., mechanical actuators)

⇨ Semantic feedback at the application level is needed (e.g., reading a sensor that observes the effect of the command)

⇨ End-to-end feedback to reassure that a subsystem achieves its purpose

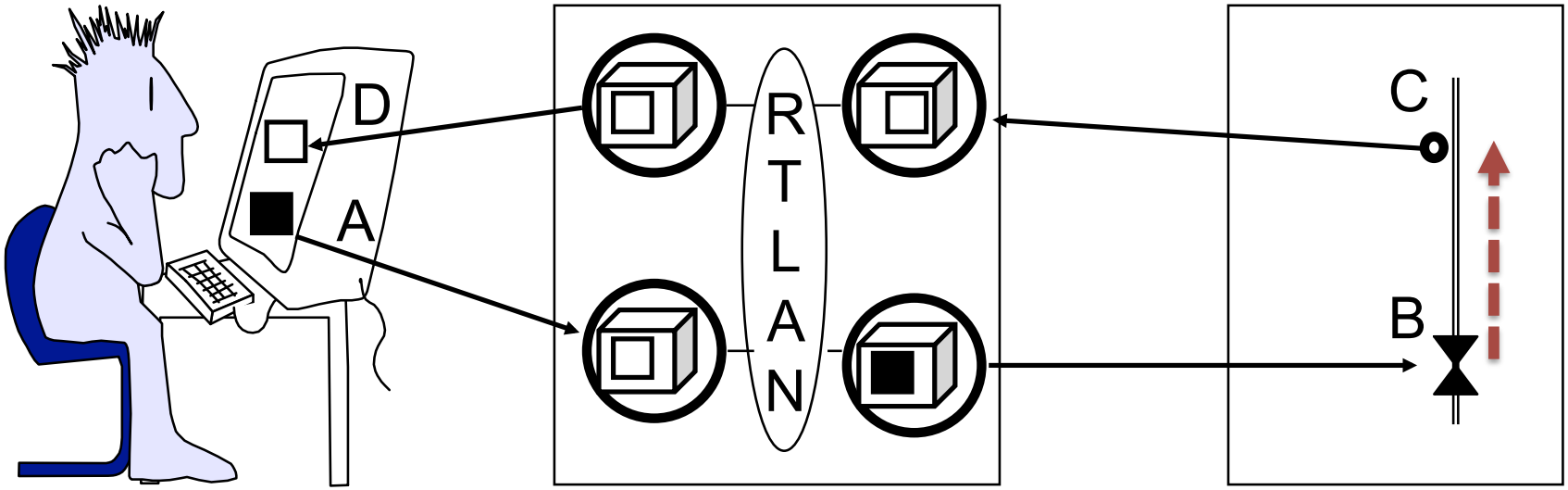
Three Mile Island Accident

Quote 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).*

- Designers assumed: ack of output-signal command to close the valve implies that valve is closed.
- Electromechanical fault in valve invalidated the assumption.
- End-to-end protocol using a valve-position sensor would have avoided the catastrophic misinformation of the operator.

End-to-End Example



A: Flow command
C: Flow sensor

B: Valve
D: End-to-end feedback

End-to-End Protocol

End-to-end Protocol

- monitors and controls the intended effect of a communication at the intended endpoints ➪ semantic feedback at appl. level.
- Provides high error-detection coverage.

  Previous example: sensor message reporting about change of flow is end-to-end acknowledgment of command message to the flow actuator.

Intermediate level protocols

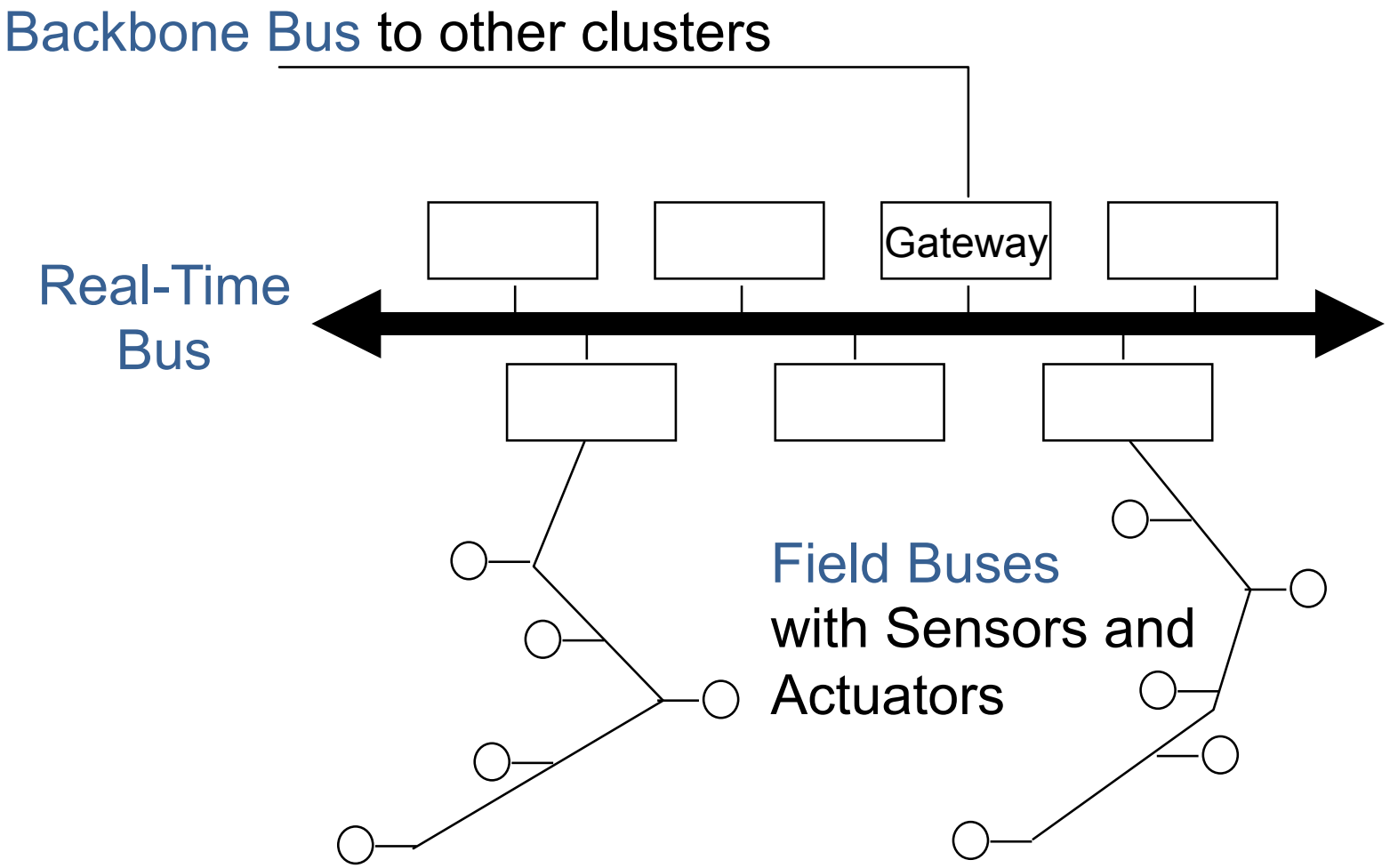
- needed if communication is less reliable than other subsystems.
- simplify the diagnosis.

RT Communication Architecture

Three levels of a RT communication architecture

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

RT Communication Architecture



Channel Characteristics

- Bandwidth
- Propagation delay
- Bit length
- Protocol efficiency

Bandwidth

- Number of bits that can traverse the channel in a unit of time
- Depends on
  - Physical characteristics of the channel (e.g., single wire, twisted pair, shielding, optical fiber)
  - Environment (disturbances)

Example:

- Bandwidth limitation in cars due to EMI
  (10Kbit/s for single wire, 1Mbit/s for unshielded twisted pair)

Propagation Delay

- time it takes a bit to travel from one end of the communication channel to the other
- Determined by
  - the transmission speed of the electromagnetic wave
  - the length of the channel

Examples

- Light in vacuum: $c_v \approx 3 \times 10^8$ m/s
- Light in cable: $c_c \approx 2 \times 10^8$ m/s
- Hence, a signal travels at about 200 m/μsec
- Propagation delay in a channel of length 1km: 5μsec

Bit Length

- Number of bits that can traverse a channel during the propagation delay
- Describes how many bits can “travel” simultaneously

Example

- Bandwidth of channel: b = 100Mbit/s
- Length of channel: l = 1000m

⇨ Bit length of channel: $bl = b/c_c \times l$

10^8 bit/s / (2×10^8 m/s) × 1000 m = 500 bit

Limit to Protocol Efficiency

Protocol efficiency limit

- Maximum percentage of channel bandwidth that an application can utilize for its data messages.

Assume multiple senders at arbitrary positions on channel.

Inter-message gap between messages to avoid collisions.

The minimum gap is the propagation delay.

- Bit length of channel: *bl*
- Message length (number of bits): *m*

⇨ Data efficiency: $deff < m / (m + bl)$

Example: Bandwidth: 100 Mbit/s, channel length: 1km, m = 100 bits

⇨ $bl \approx 500$ bit; $deff < 100/600 = 16.6\ \%$

Maximum Protocol Execution Time (d_{max})

At the transport level, d_{max} depends on

- Protocol stack at sender (including error handling)
- Message scheduling strategy at sender
- Medium 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.

Medium Access Protocols

Arbitration for shared communication medium access (not point-to-point)

- CSMA/CD
- CSMA/CA
- Token Bus
- Minislotting
- Central Master
- TDMA

Medium Access – CSMA/CD

CSMA: Carrier Sense Multiple Access

CD: with Collision Detection

- Communication controller that wishes to send, senses the bus for traffic; it starts sending if it detects no carrier signal
- Collision: different nodes start sending at the same time
- Transmitter listens to signal to detect collisions
- Collision: jam signal; re-send after random time interval; max. *k* (e.g., 10) attempts to send

Example: Ethernet (bus topology, shared medium)

Medium Access – CSMA/CA

CA: with Collision Avoidance

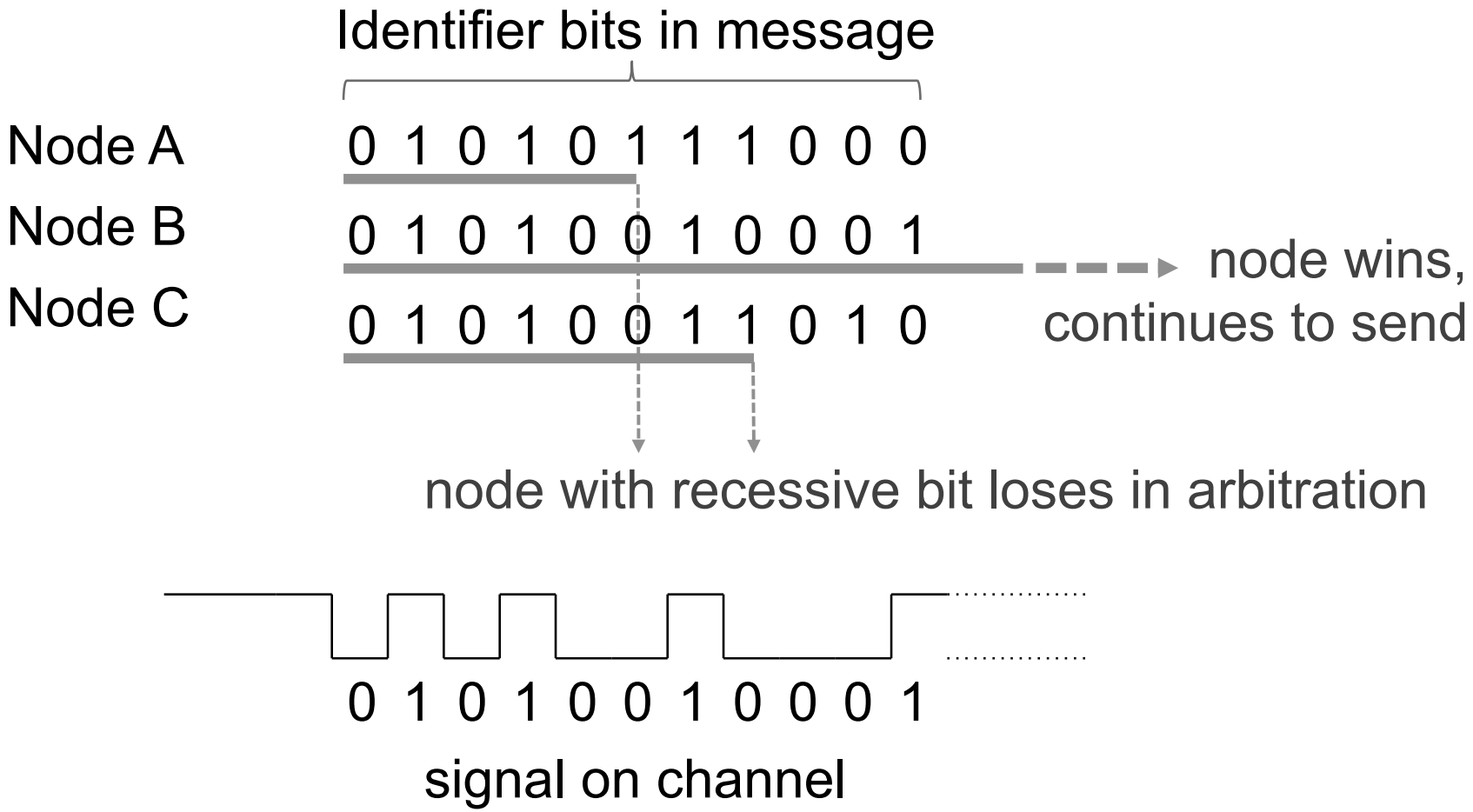
- Typical mechanism for CA: bit arbitration
- There are two states on the communication channel
  - dominant (e.g., bit = 0)
  - recessive (e.g., bit = 1)

If two stations start to transmit at the same time

⇨ the station with a dominant bit in its arbitration field wins

⇨ the station with a recessive bit has to give in.

Bit Arbitration in CSMA/CA



CSMA/CA Timing Parameters

Arbitration: every bit has to stabilize before arbitration

- Propagation delay of channel d_{prop} << length of a bitcell

Example:

Bus length: 40 m, d_{prop}: 200 nsec

- length of a bitcell: 1 μsec = 5 propagation delays!

CAN – Control Area Network

Arbitration: CSMA/CA

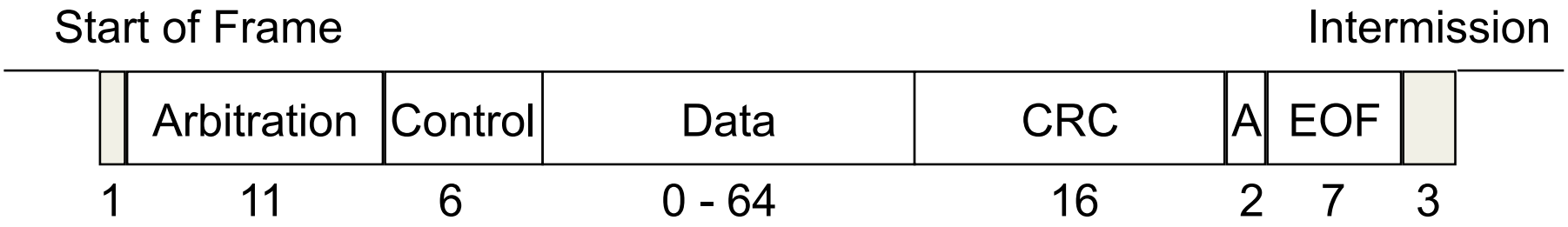
Communication speed: 1 Mbit/second

Channel length: about 40m

Standard Format: 2032 Identifiers, 11 bit arbitration field

Extended Format: > 10^8 Identifiers, 32 bit arbitration field

Frame format



Medium Access – Token Bus

- Token: special control message to transfer the right to transmit
- Only the token holder is allowed to transmit
- Senders form a physical or logical ring

- Central timing parameters
  - Token Hold Time: longest time a node is allowed to hold the token
  - Token Rotation Time: longest time for a full rotation of the token

- Token Loss constitutes a serious problem in HRT context
  - Token recovery: node creates new token after a random timeout – may lead to collision and retry

Medium Access – Minislotting

- Time is partitioned into a sequence of minislots
- Duration of minislot > maximum propagation delay
- Each node is assigned a unique number of minislots that must elaps with silence on the channel, before it can start to send

Example: ARINC 629

ARINC 629

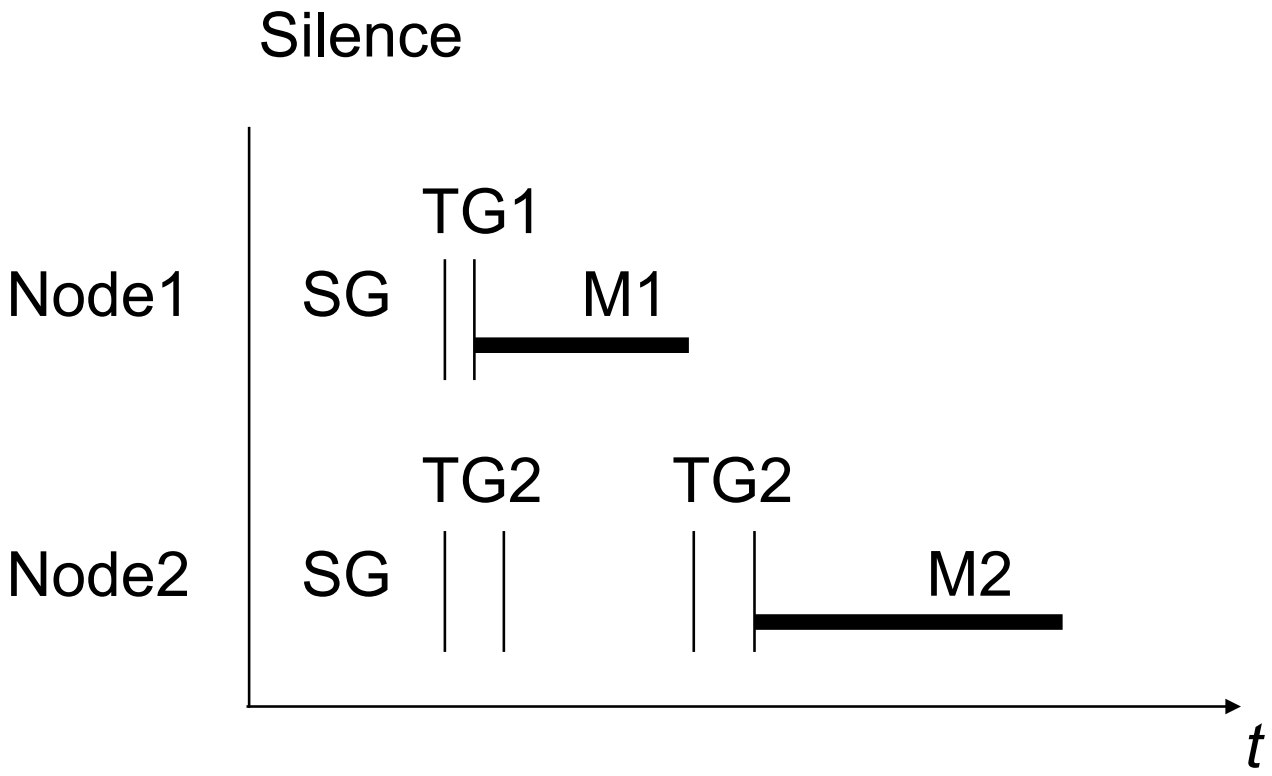
ARINC 629 is a mini-slotting protocol that is used in the aerospace community.

Media access is controlled by the intervals:

TG: Terminal Gap, different for every node, longer than the propagation delay of the channel, determines send order (node-specific number of minislots)

SG: Synchronization Gap, longer than longest TG

ARINC 629 – Timing Diagram



Medium Access – Central Master

- A central master controls the access to the channel
- In case the master fails, another node must take over the role of the master
- Central master is called bus arbitrator

- Master periodically broadcasts variable names
- Node producing the variable then broadcasts its value
- If time remains, nodes may also send sporadic data after being polled by the master

Example: FIP

Medium Access – TDMA

- TDMA: Time Division Multiple Access
- Static medium access strategy
- Requires a (fault-tolerant) global time base
- Time is statically divided into time slots; a static message schedule assigns each slot to one node that may send
- In assigned slot: node can send one frame per TDMA round
- The message schedule may provide for a sequence of different TDMA rounds, which form a cluster cycle

Example: TTP

The Time-Triggered Protocol, TTP

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

TTP: TTP/C for safety-critical applications
(TTP/A: master-slave protocol for field buses)

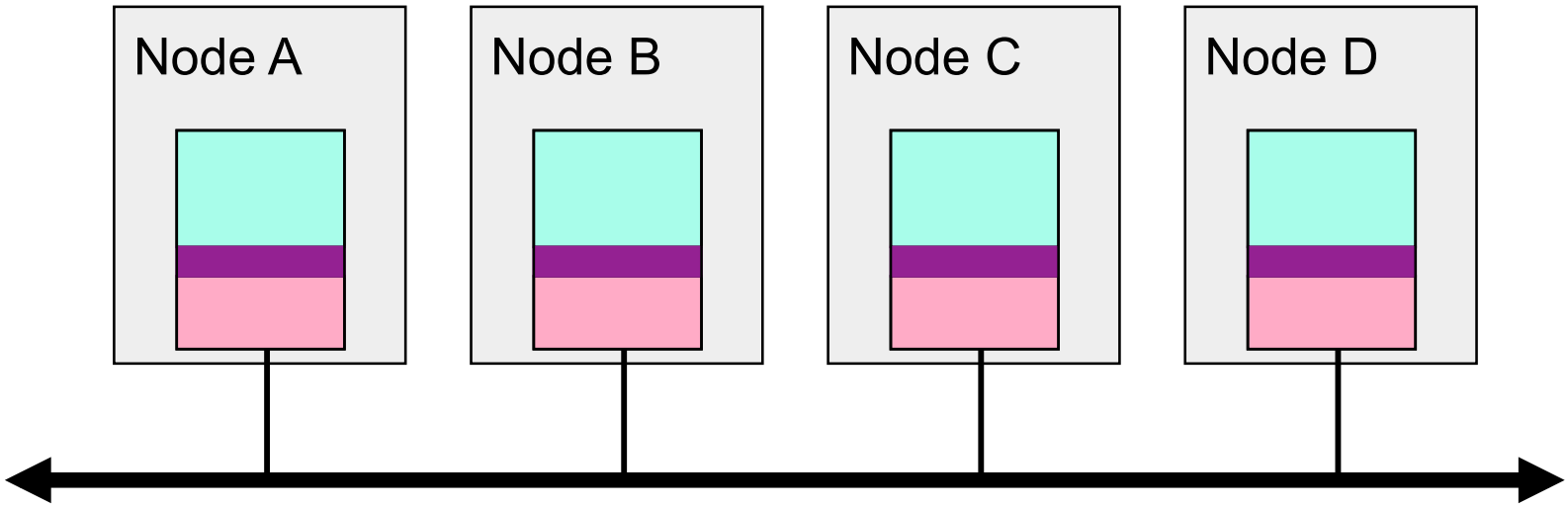
TTP Basics

TTP Node: one electronic control unit (ECU)

TTP Cluster: nodes connected by TTP channel

Bus topology with broadcast communication

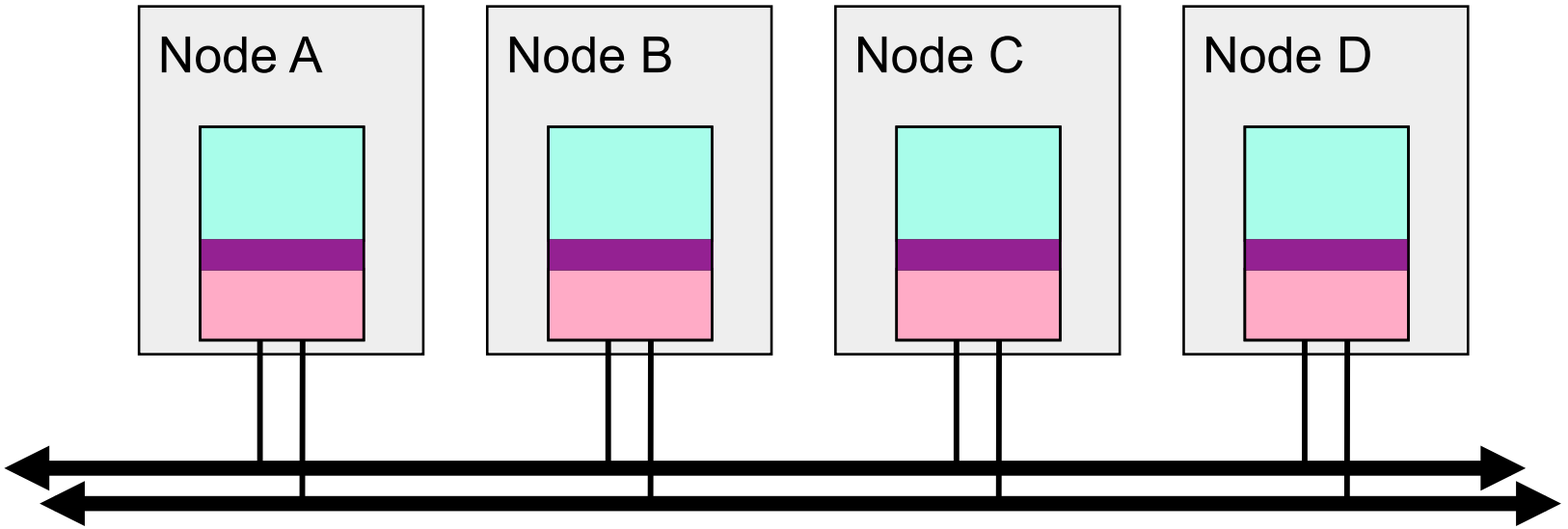
- Publisher writes to bus (e.g., sensor values)
- Subscriber: reads values from bus, reacts (e.g., actuator output))



Redundant Channels

Single fault hypothesis: communication system must tolerate one fault at a time

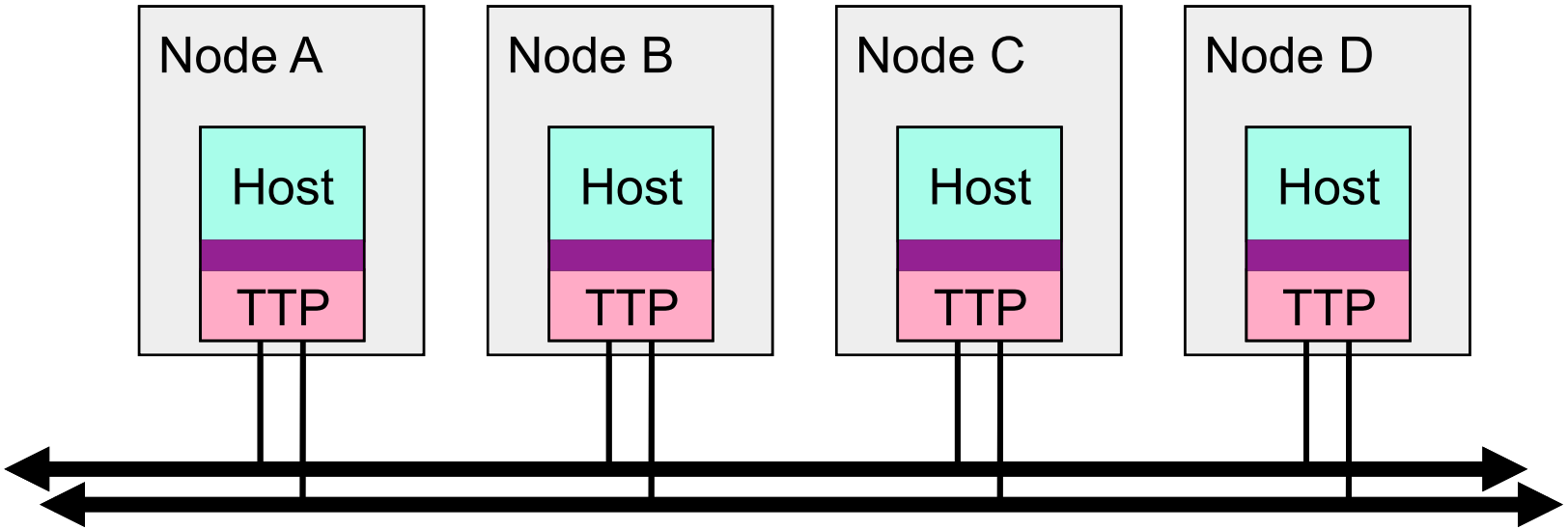
- Redundant communication channels make bus fault tolerant



Separation Host – TTP Controller

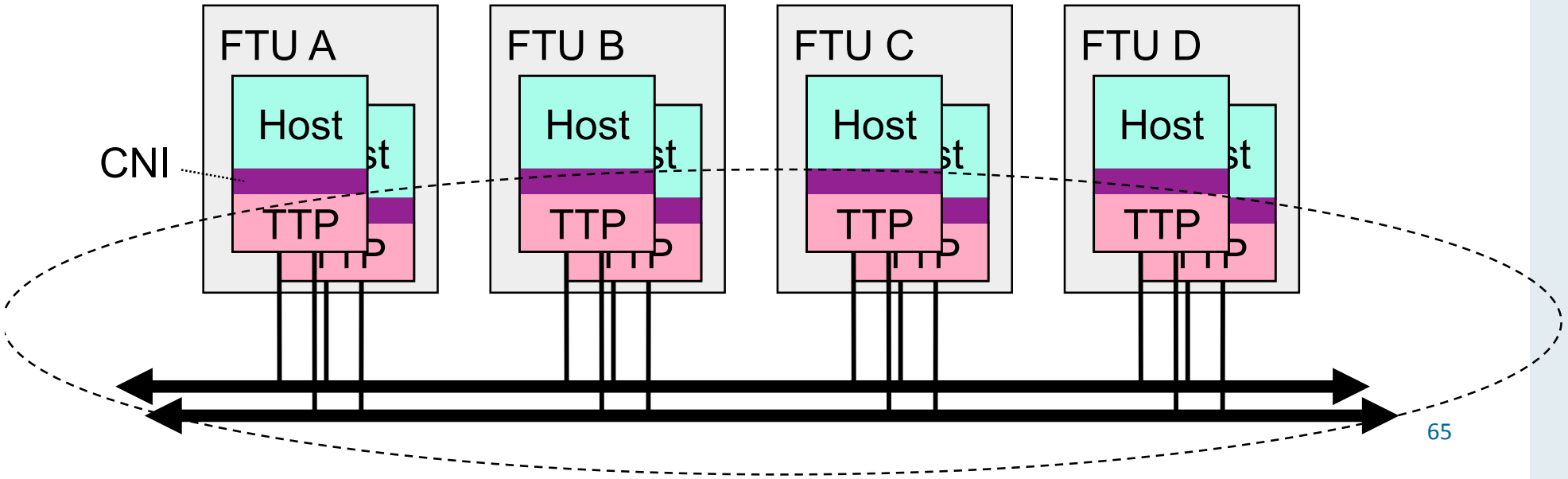
Single fault hypothesis: a single fault in a node must be tolerated

- Nodes are partitioned into two independent parts
  - Host computer: executes the application code
  - TTP controller: provides TTP services



Fault-Tolerant Architecture

- The communication network, consisting of the bus interconnect and the TTP controllers is duplicated to tolerate network faults
- Components are duplicated to tolerate faulty hosts



Communication Network Interface – CNI

CNI: data-sharing interface between host and TTP Controller

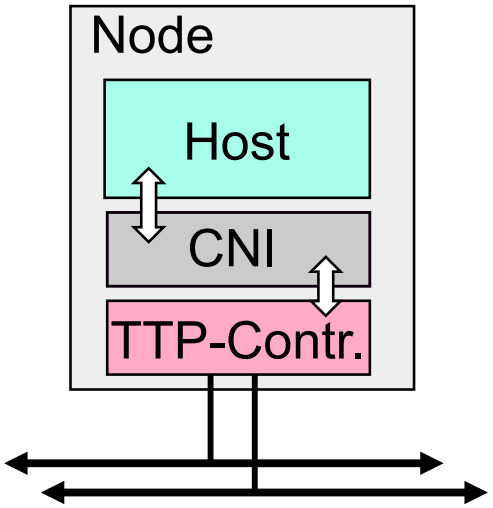
- Contains incoming/outgoing data; network status info., bits to control the TTP controller
- Acts as temporal firewall for control signals

Host

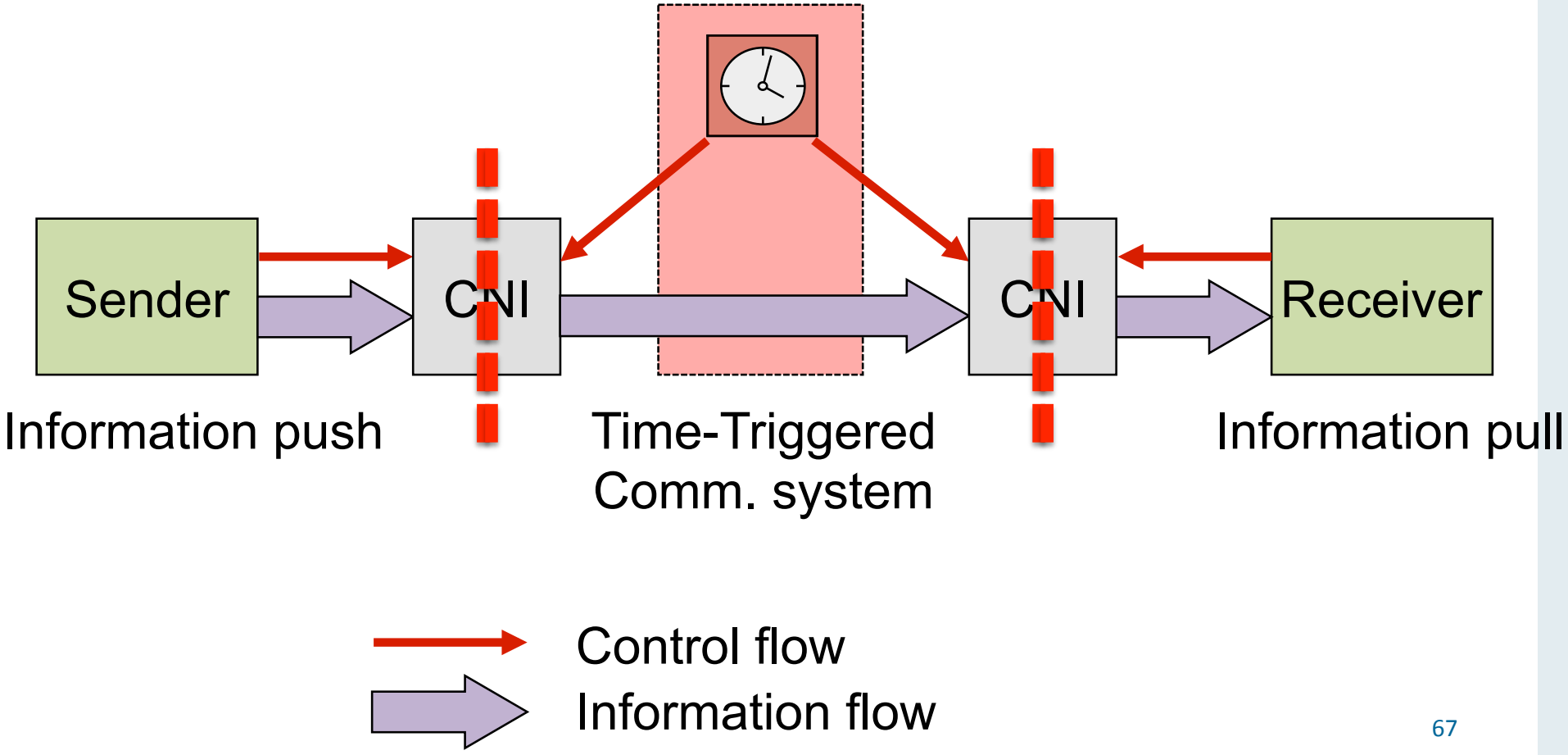
- writes data for sending to CNI
- writes control data to CNI
- reads received data/status info from CNI

Controller

- writes received message data to CNI
- sets status
- sends messages composed from CNI data over the network

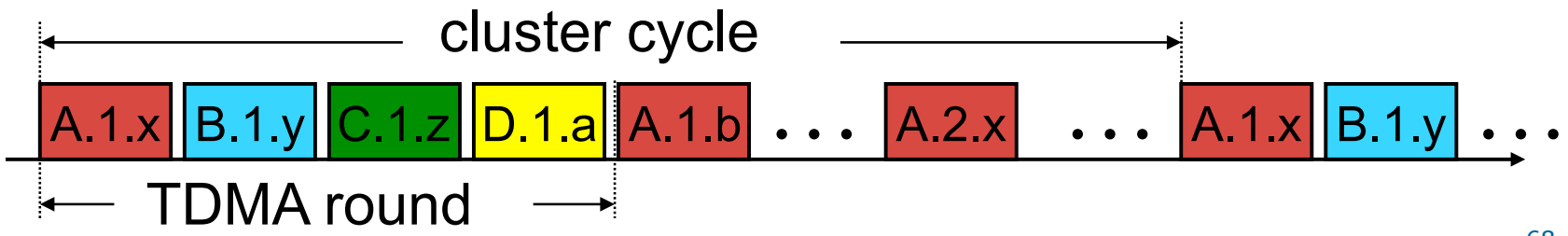


Avoiding Control Conflicts in TT Comm.



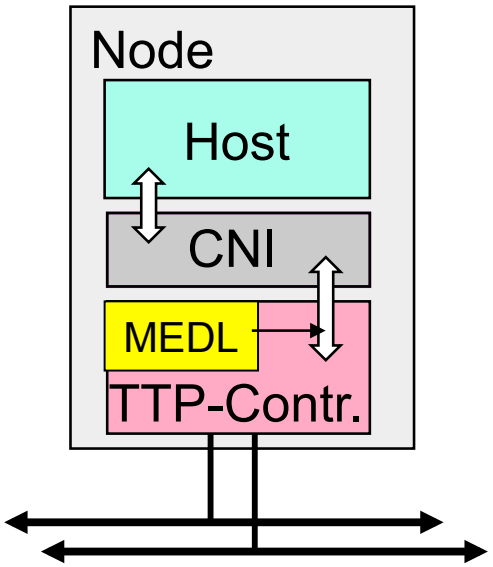
Time-Triggered Communication

- TDMA – Time Division Multiple Access to communication medium ⇨ global time base
- Components send regularly in pre-defined time slots
- Messages have different periods – components send different messages in successive TDMA rounds
- Cluster cycle: global sequence of TDMA rounds that is repeatedly executed
- Implicit naming/addressing



Message Schedule

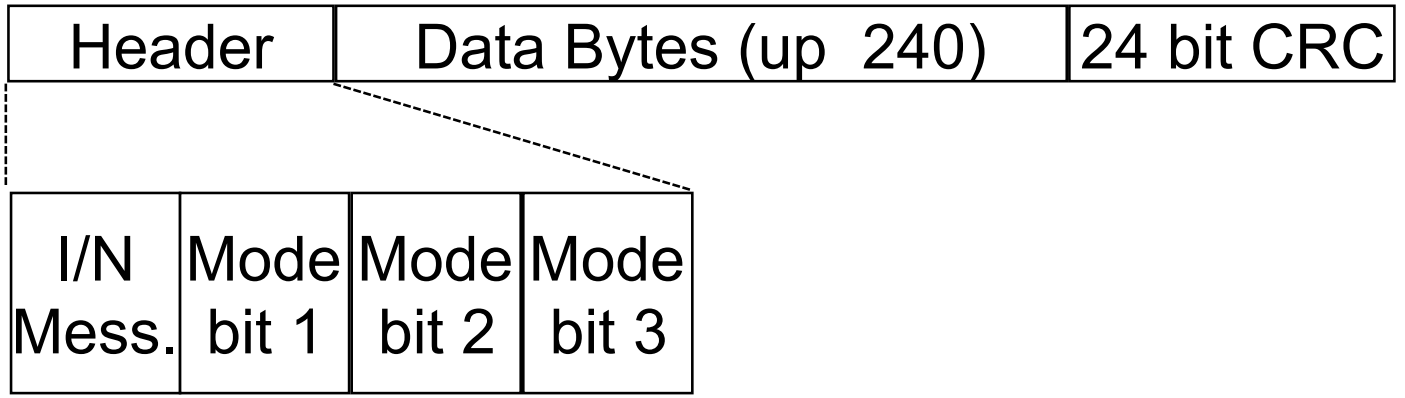
- The message schedule is stored in the Message Descriptor List (MEDL)
- MEDL is application dependent
- MEDL is stored in every TTP controller (flash memory initialized at startup)
- Controller sends messages according to MEDL
- Controller works completely independent from host, no waiting



TTP – Principle of Operation

- TTP generates a global time-base
- 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
- Every message header contains 3 mode change bits that allow the specification of up to seven successor modes

TTP Message Format



- Excluding the inter-message 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.

Use of A-Priori 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:

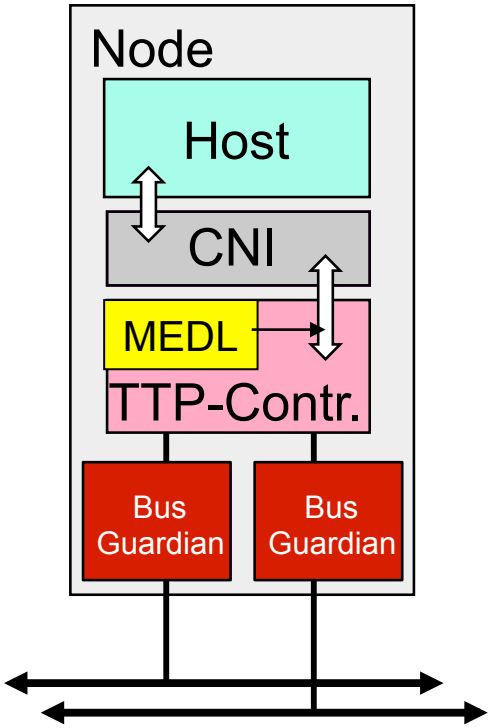
- Architecture level: fault management is based on the assumption that all nodes are fail-silent
- Node level: mechanisms are provided that increase the error detection coverage to justify the fail-silent assumption.

Bus Guardian

- Babbling Idiot: erroneous controller sends at arbitrary times, i.e., outside its assigned time slot
- Babbling idiot violates fault hypothesis (node + comm. medium affected)

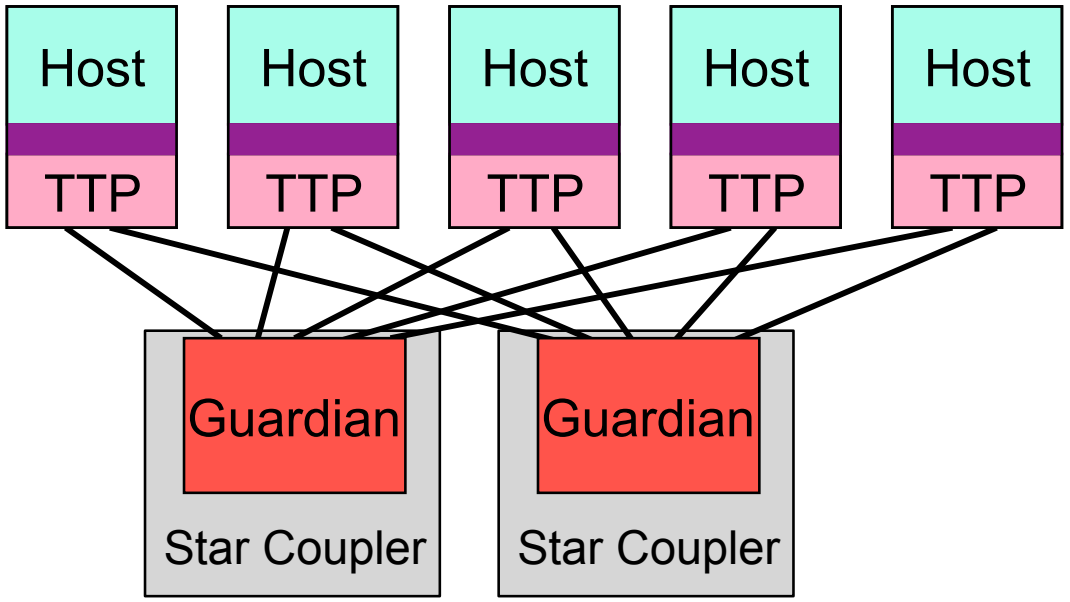
Bus guardian: independent controller for bus access (gate keeping function)

- Knows when node is allowed to send
- Opens gate to bus for sending only during the designated sending slot



TTP with Star Topology

- TTP can be laid out in a star topology
- Bus guardians are only needed in star-couplers switches
- Star coupler deals with slightly-off-specification (SOS) errors
  - SOS errors: inputs that some nodes classify as correct, others as erroneous (e.g., bus-signal voltage at tolerance limit)
  - Unambiguous interpretation and propagation of messages



Continuous State Agreement

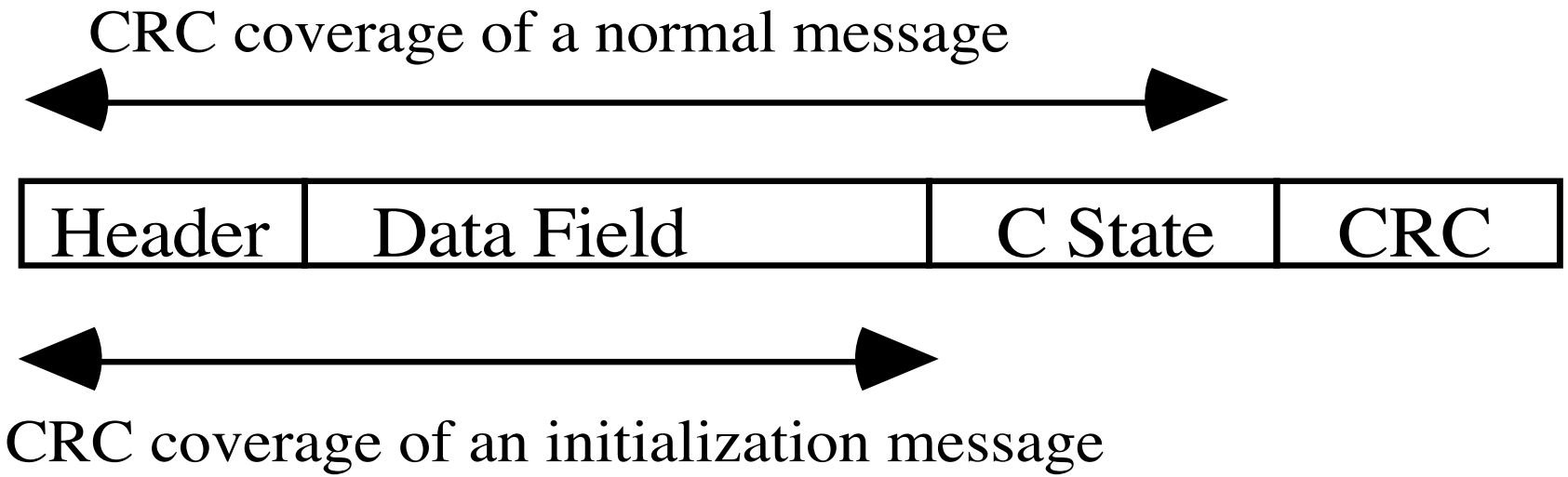
The internal state of a TTP controller (C-state) is formed by

- 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



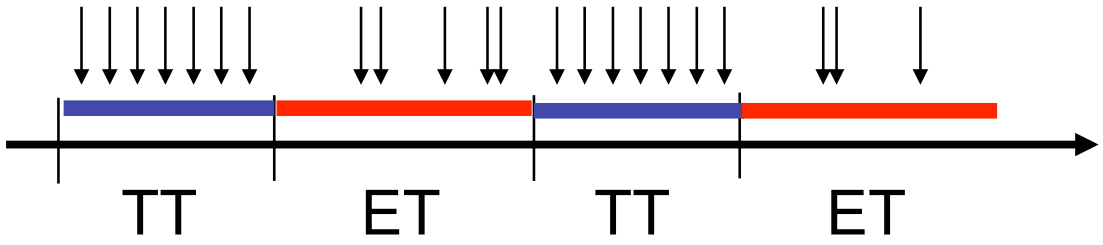
C -State: Time, Membership Vector, and MEDL Position at sender respectively at receiver

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.

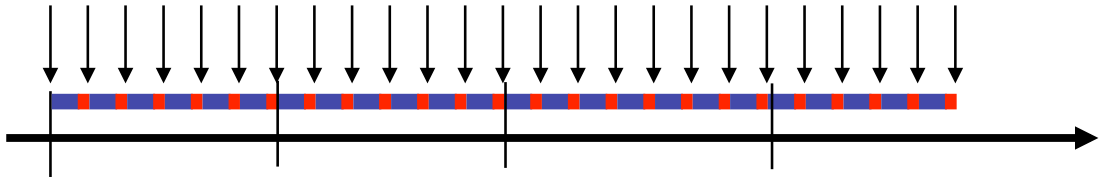
Integrating TT and ET Messages

- Alternating time windows for TT and ET communication
- Two different communication protocols (TT, ET)
- Loss of temporal composability



Integrating TT and ET Messages (2)

- Layered protocol: ET services on top of TT protocol
- Single TT communication protocol
- Loss of global bandwidth as TT messages that transport ET contents are assigned to a-priory defined sending nodes



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

- TT: TDMA, similar to TTP
- ET: mini-slotting, similar to ARINC 629
- TT and ET transmission phases alternate with fixed period

Distributed clock synchronization

At present no membership

TTEthernet

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* (*fly-by-wire*, *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.

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

Two Categories of Messages

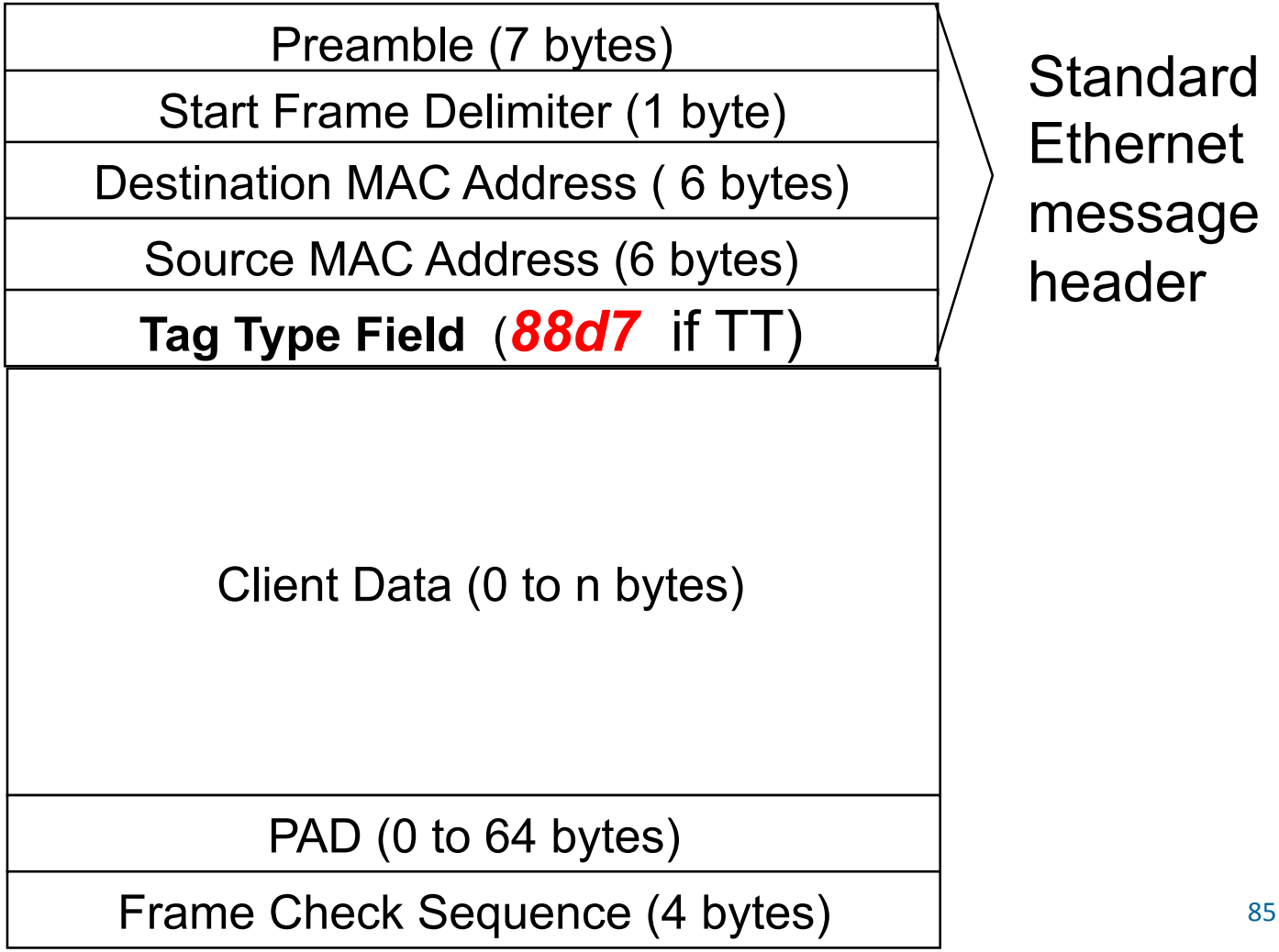
ET-Messages:

- 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 Message Formats



Conflict Resolution in TTEthernet

Assumes use of Switched Ethernet

⇨ the *switch* acts as a message arbitrator:

- TT versus ET: TT message wins, ET message is delayed.
- TT versus TT: Failure, since TT messages assumed to be properly scheduled (closed world system)
- ET versus ET: One message has to wait until the other is finished (standard Ethernet policy)

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

Points to Remember

Communication System Needs

Implicit vs. explicit flow control

Limits to protocol design

Protocol types: ET – RC – TT

End-to-end protocol

Medium access protocols

RT-Protocol example: TTP