# Real-Time Communication 1

## Principles & Protocol Types

slide credits: H. Kopetz, P. Puschner

# Overview

- Communication system requirements
- Controlling the flow of messages
- Types of protocols
- Properties of communication protocols
- End-to-end protocols (more than just messages)

# 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**

# RT Communication-System Needs

- Determinism: timeliness, predictability, testability, certification
- Composability: temporal encapsulation (separation communication – computation) ➪ simple flow control
- Dependability: error-containment of failing nodes; non-intrusive monitoring, TMR (multicast)
- Support for different types of real-time applications, from safety-critical control systems to multimedia and open systems.
- 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

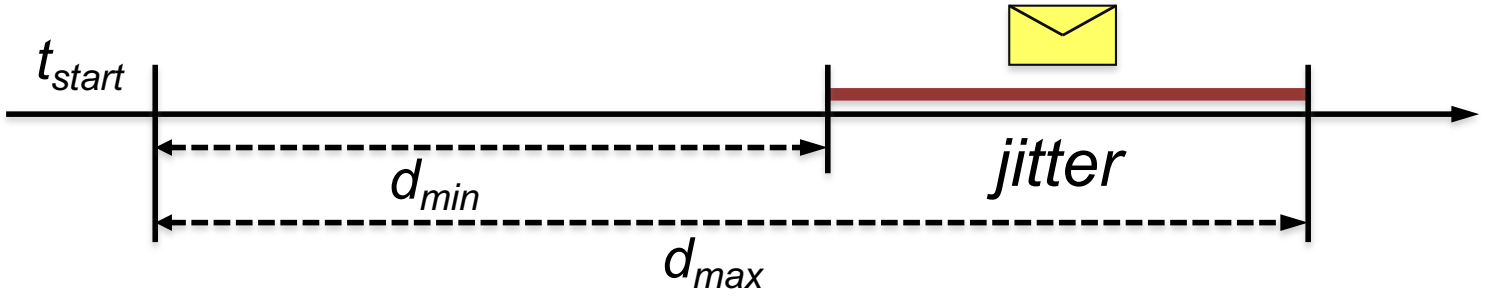
Atomic data structure transferred from sender to one or more receivers (multicast, broadcast)

Transmission timing

$t_{start}$ … start instant at sender

$d_{min}$ , $d_{max}$ … minimum / maximum transmission delay

$d_{max} - d_{min}$ … *jitter* of the transmission channel




Flow control: state vs. event messages

# Flow Control

… governs flow of information between communicating partners

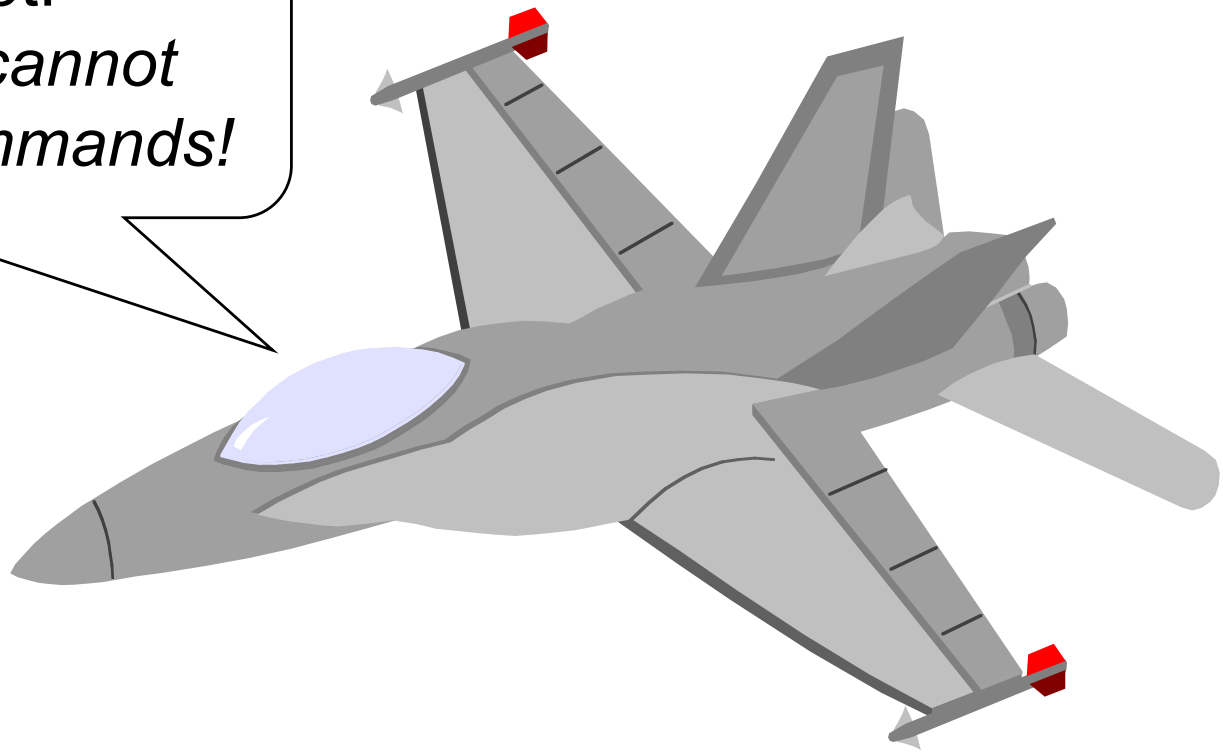
- The sender must not outpace the receiver, therefore
- The processing speed of the receiver should determine the pace of communication

# Explicit Flow Control

- Sender (1) transmits a message and
  Sender (2) waits for an acknowledgement from the receiver.
  PAR … Positive Acknowledge or Retransmission
- Error detection by sender.
- Receiver is authorized to slow down the sender
  (back-pressure flow control)
  ➪ *sender is in the sphere of control of the receiver*.
- What does a missing acknowledgement tell us?
  - Message loss? – original vs. ACK
  - Receiver is late?
  - Receiver has failed?

# Example: Explicit Flow Control

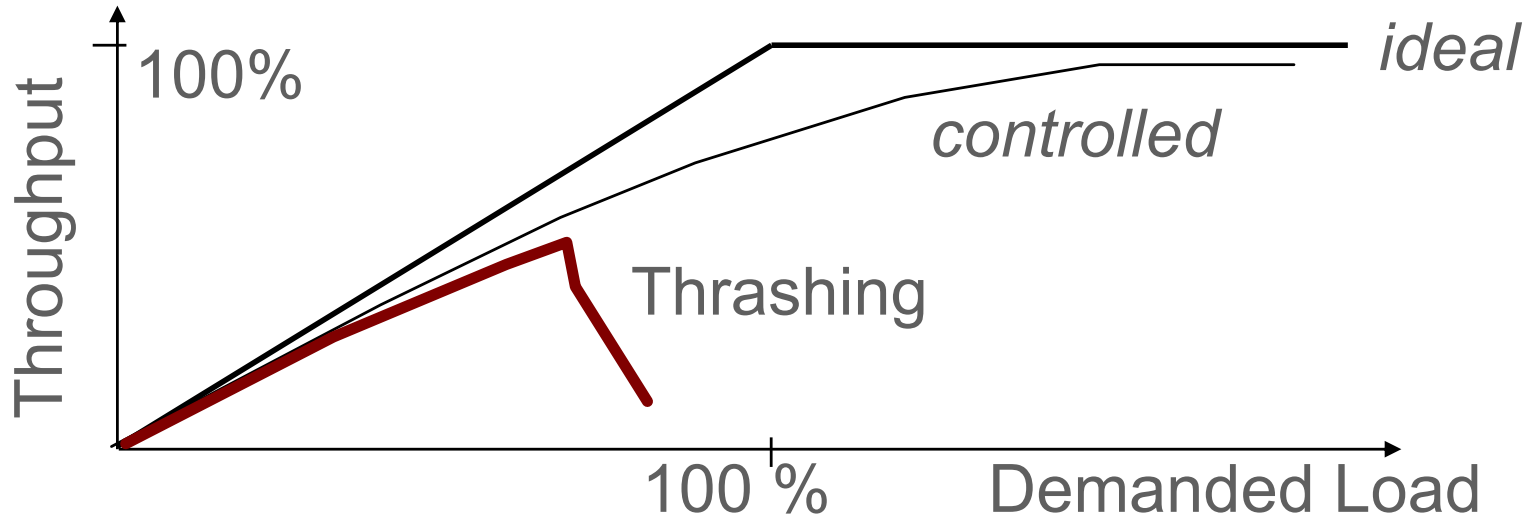
Computer to Pilot:
*Please fly slower, I cannot keep up with your commands!*



# 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



# 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.

# 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 are not suited:

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

# 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 – closed system
- Use a coordinator / 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.

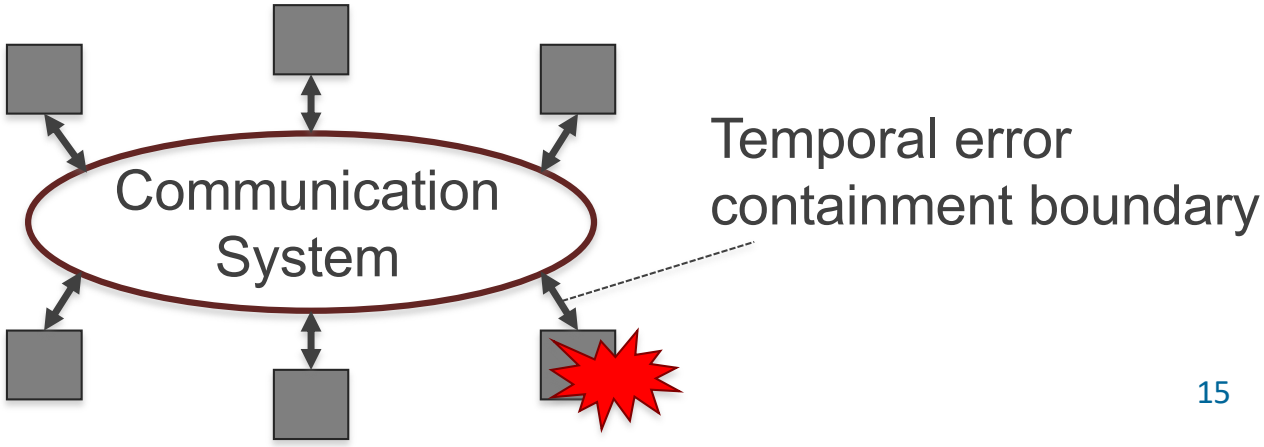
The synchronization can be established by:

- Reference to a *single global time base* (global pre-planning)
- 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*.



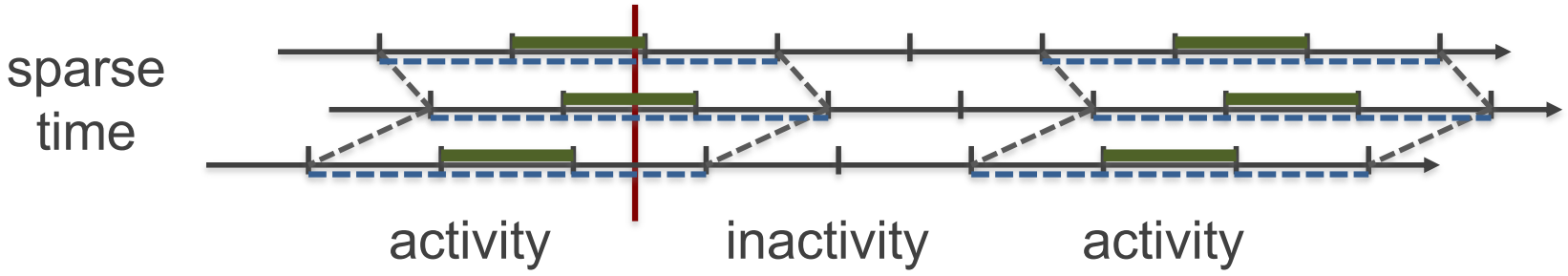
# Consistent Ordering of Events

*Sparse global time* *base* for

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

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 *event message* at any time.
- Error detection is by sender.
- Error detection needs an acknowledgement. This creates correlated traffic in a multicast environment.
- Maximum execution time and jitter of the protocol are large compared to the average execution time (re-transmission).
- No temporal encapsulation.
- Explicit flow control to protect the receiver from information overflow. Sender in sphere of control of receiver.

Examples: CSMA/CD, CAN

# Rate-Constrained (RC) Protocols

- Provide minimal guaranteed bandwidth.
- Enforce traffic compliance to a given profile.
- The message rate of the sender is bounded.
- 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.

Examples: Token protocol, AFDX, TSN (AVB)

# Traffic Shaping

Traffic metering: check compliance of packets with traffic contract

Traffic shaping: impose limits on bandwidth & burstiness

- Self limiting sources
- Network switches

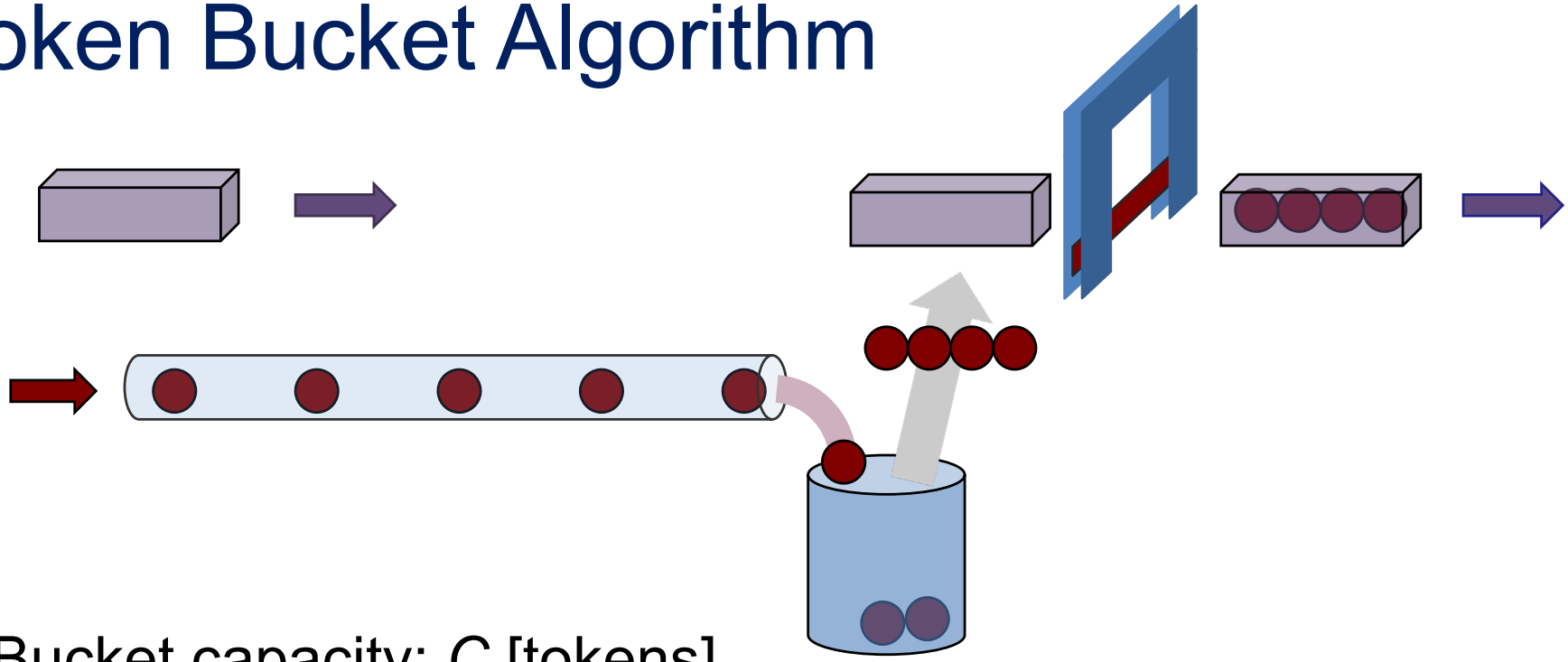
Buffering of packets that arrive early

- Buffer dimensioning (?)
- Strategy for deleting messages when buffers are full

Shaping effect

- Shaping traffic uniformly by rate
- More sophisticated characteristics – burstiness

# 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 sending of *state messages*.
- Message-send times are *a-priory* known to all receivers.
- Maximum protocol execution time is about the same as average execution time ⇨ small jitter (reading error).
- Error detection by receiver, based on a priori knowledge.
- Uni-directional protocol, well suited for a multicast.

# Idempotence

A set of messages is *idempotent*, if the effect of receiving more than one messages of this set is the same as the effect of receiving a single message.

- Duplicated state messages are idempotent.
- Duplicated event messages are not idempotent.

Idempotence of redundant messages simplifies the design of fault-tolerant systems.

# 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 |
| #Reads at receiver | Exactly once | At least once |

# Event Message vs. State Message (2)

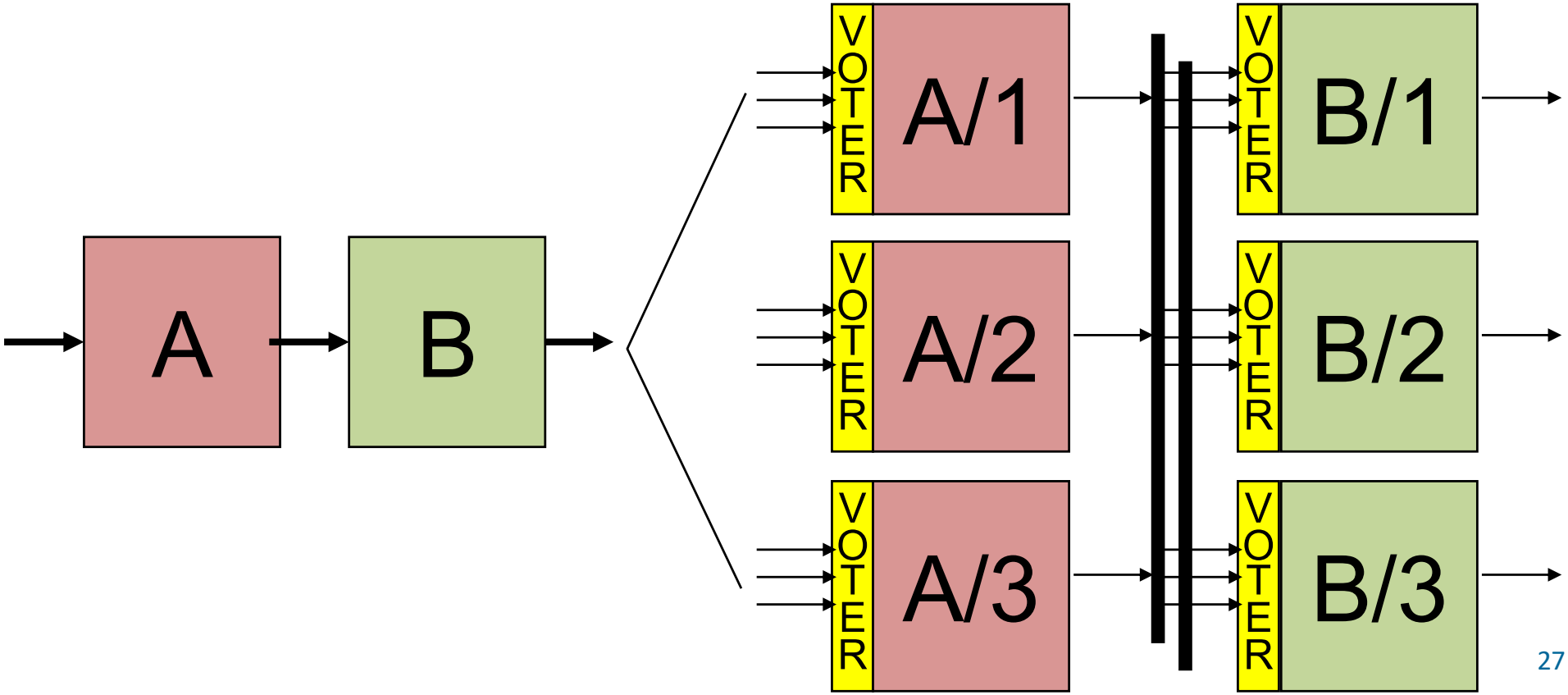
| *Characteristic* | *Event Message* | *State Message* |
|---|---|---|
| Idempotence | no | yes |
| Consequences of message loss | Loss of state synchronization of sender and receiver | State information is not available for one sampling interval |
| Typical comm. protocol | Positive acknowledgement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

# Dependability and TT Communication

- Idempotence of messages supports active redundancy
- Message broadcast supports transparent TMR (next slide) and non-intrusive observation/monitoring
- Broadcast of g-state supports re-intgration of components
- Detection of message loss based on a-priory schedule
- 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

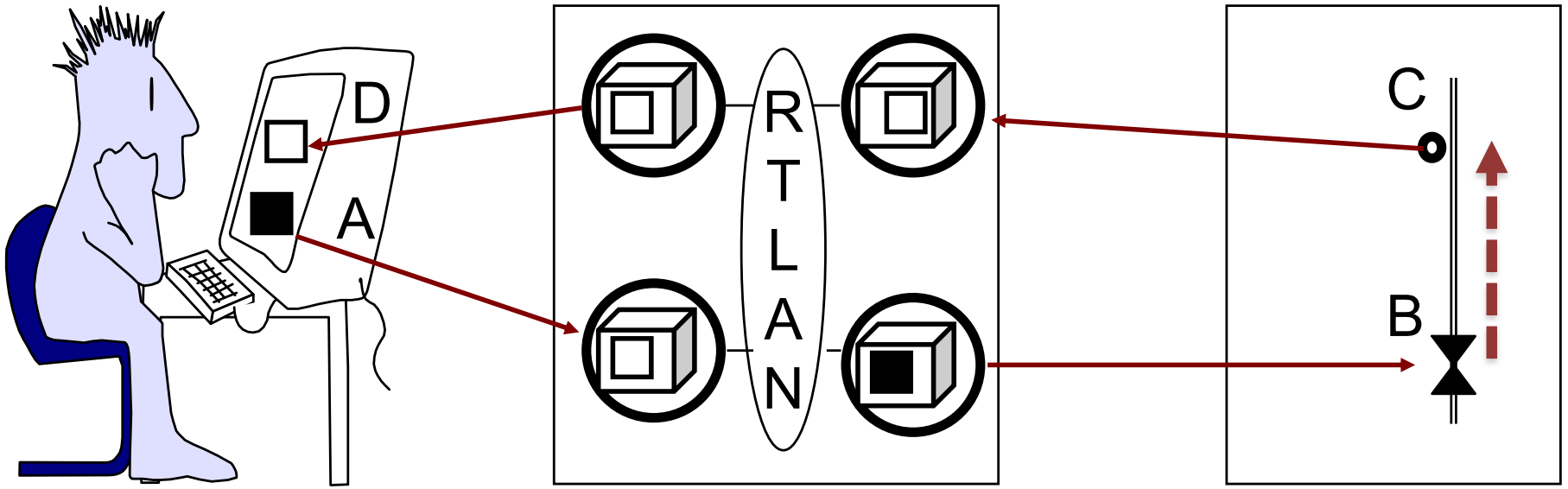
Successful message delivery

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

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

- ⇨ End-to-end feedback: semantic feedback at application level (e.g., reading a sensor that observes the effect of a command)
- ⇨ Reassures that a subsystem achieves its purpose

# End-to-End Example




A: Flow command
B: Valve
C: Flow sensor
D: End-to-end feedback

# 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 Protocol

## End-to-end Protocol

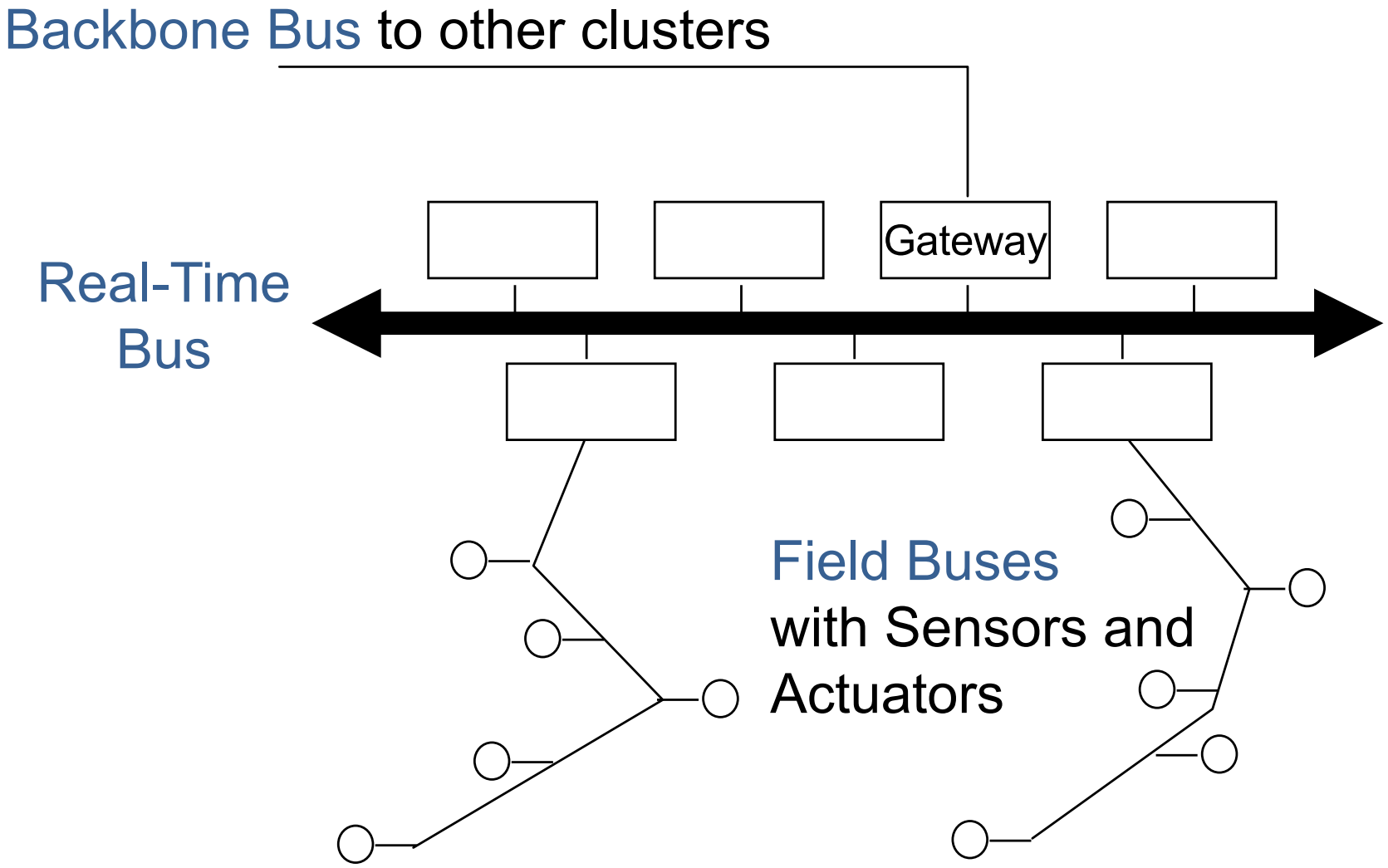
- monitors and controls the intended effect of communication at the intended endpoints ➪ semantic feedback at appl. level.
- Provides high error-detection coverage.

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

## Error detection of intermediate level protocols

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

# RT Communication Architecture



# 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

# Communication-Channel Performance

- 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: 5 µsec

# 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: $bl = b / c_c \times l$

$$10^8 \text{ bit/s} / (2 \times 10^8 \text{ m/s}) \times 1000 \text{ m} = 500 \text{ bit}$$

# 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; $deff < 100/600 = 16.6$ %

# Points to Remember

RT-Communication System Needs

Flow control

Limits to protocol design

Event-triggered, rate-constrained, time-triggered protocols

Event messages vs. state messages

End-to-end protocol

Performance parameters