Real-Time Systems

Hermann Härtig

Real-Time Communication
A short Overview

(following Kopetz, Liu, Schönberg)

TECHNISCHE UNIVERSITÄT DRESDEN

30.11.2018

Contents

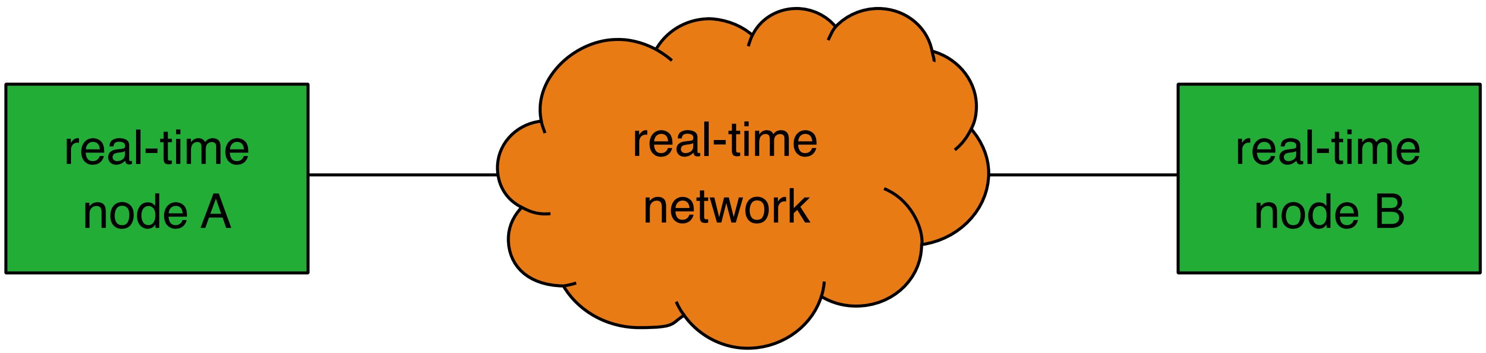
- Overview
- IO Busses: PCI
- Networks as schedulable resources:
  Priority / Time-Driven / Weighted Round Robin
  - Priority – Based Field: CAN/Token Ring
  - Time Division Multiple Access
    z.B. Ethernet in RT: TT-Ethernet
  - Weighted Round Robin in Wide Area

What to expect of a real-time network ?

Deliver communication services to the requesting nodes **timely (**and reliably, securely, efficiently)

- lower bound for bandwidth
- upper bound for latency and jitter

even in case of peak load and faults.



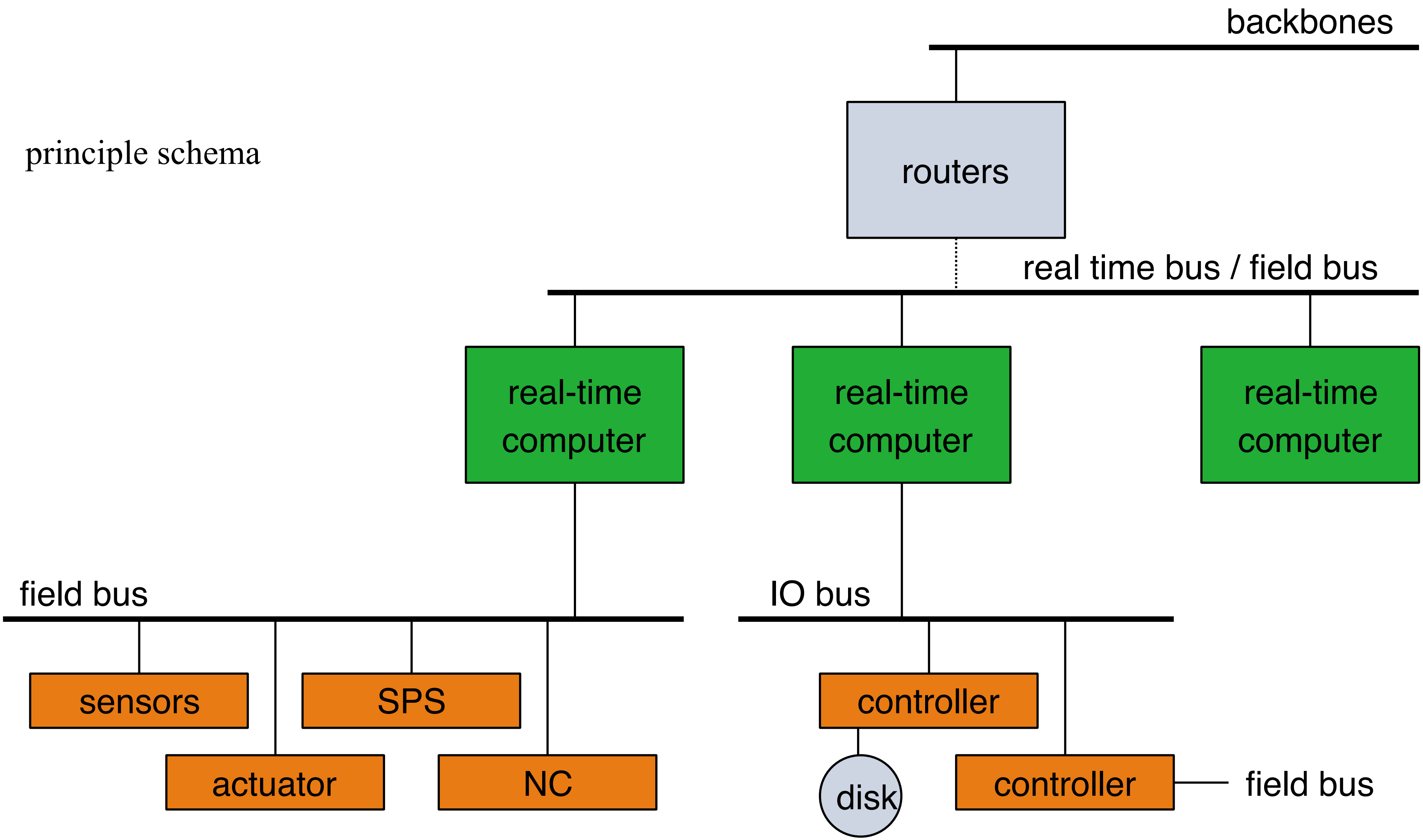
Networks as schedulable resources

- Analogous to CPU scheduling:
  - time driven
  - priority driven
  - (Weighted Round Robin)

- Analogy:
  - WCET: Amount of traffic / message transmission time
  - Periodic tasks: periodic messages
    messages are usually non-preemptive !!
  - Deadline: max delay
  - Admission: connection establishment
  - Schedulers in switches or distributed on nodes

Where does Real-Time Communication matter ?

principle schema

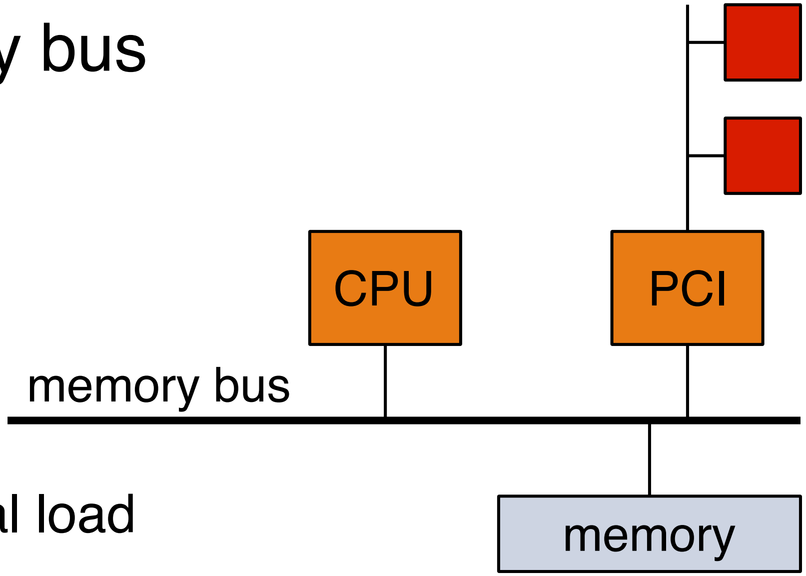


Typical requirements

- Short data in control applications
  Large data in media applications
- Short periods (ms)
  - monitoring, feedback control
- Fast aperiodic (ms)
  - alarms
- Non real-time data
  - configuration, logs
- Multicast
- Predictability in the presence of faults

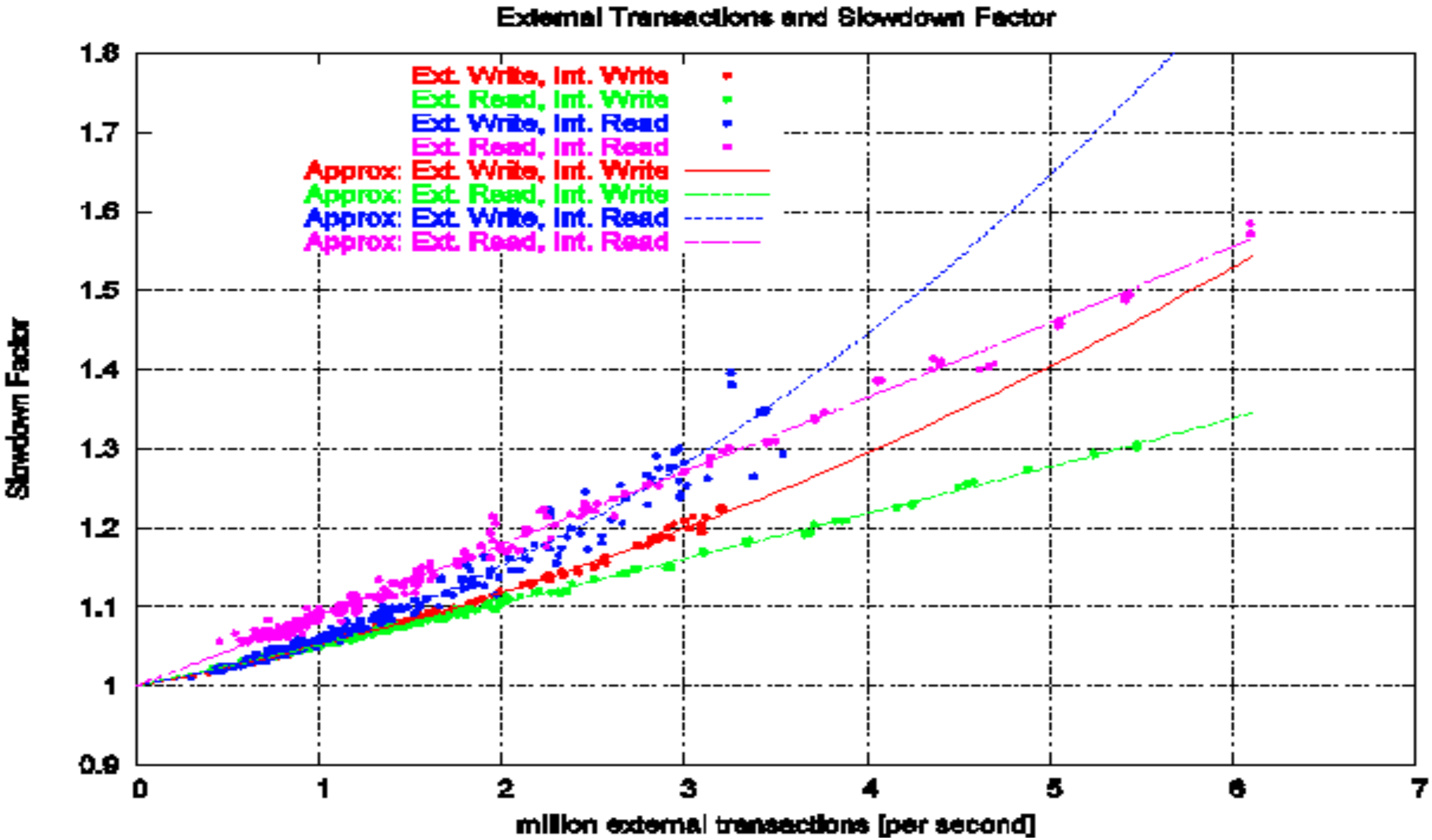
Intra-Node communication: IO bus (PCI)

- All data transfers share the memory bus
  - Memory bus conflicts
  - PCI devices start transfers when data is available
  - CPU-Memory-accesses are influenced by PCI devices
  - applications are slowed down by external load



- Slowdown factor F: (ET: execution time)
  ET under external load/ET without any external load
  - application-specific depends on memory access pattern
  - obtained by measurements attempts to statically predict

Some measurements



Source: Dissertation Sebastian Schönberg (see our Web-Pages)

Intra-Node communication: PCI bus

- PCI bus specification does not prevent real-time implementation
  - Arbitration responsible for granting bus to devices
  - Current implementations use round-robin arbitration

- Telecom Systems
  - PCI bus for control data
  - TDMA bus for real-time data

Intra-Node communication: PCI bus

Alternatives (research proposals)

- real-time capable PCI bus arbiter
  - Assign individual share of the PCI bus to devices
  - Enforce reservation
  - Give unused bandwidth to time-sharing devices
- RT-Bridges
  - Between device and bus
  - Enforces bandwidth restriction

Memory Bus in MultiCore systems: active area of research

- Memory throttling

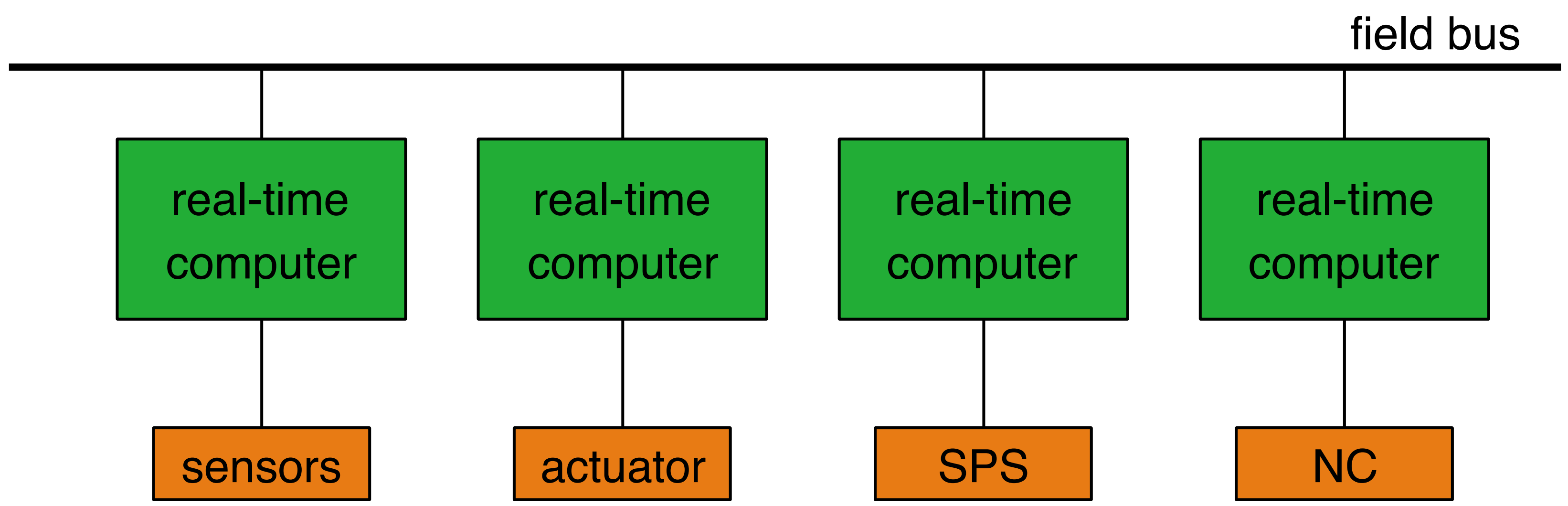
Examples

- Fieldbus:
  - CAN, Token Ring: priority based scheduler
  - TTP/Flexray/TT-Ethernet: time driven scheduling

- Switch:
  - (Delay EDD)
  - Weighted Round Robin
    (neither priority nor time driven !)

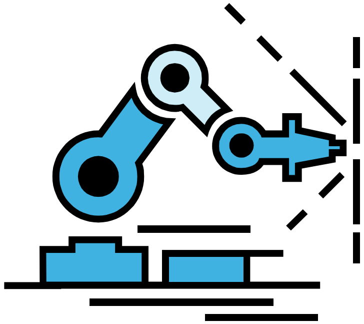
Intra-Node communication: field busses

- Situation is moving
  - Smart sensors, actors
  - Wireless lans



Fieldbus

- Networks for process control, factory automation, cars, avionics, X-by-wire, embedded applications



- Networks are typically called field buses:
  - collapsed network stacks (application services access the data link directly)
  - real-time transport
  - short application messages (no need for fragmentation and reassembling)
  - often a bus, hence a single broadcast domain, no routing

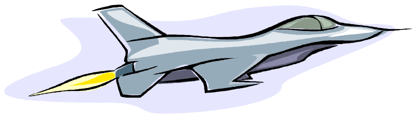


Image source: Microsoft Clip-Art Gallery

Multiple access to shared medium

- CSMA/CD shared ethernet
- CSMA/CR or CSMA/BA with priorities (CAN)
- Token Ring with priorities
- TDMA

CAN – controller area network

- Bosch GmbH, Version 2.0 released in 1991
- ISO Standards 11519 (94) and 11898 (95)
- used in automotive industry, process control, manufacturing automation, embedded application domains
- cost-effective

- CSMA/BA:
  carrier sense multiple access/bitwise arbitration

carrier sense (bitwise arbitration)

Message 1: 00100 xxxxx

Message 2: 00010 xxxxx

"carrier sense":

Message 2 should not be sent

carrier sense bitwise arbitration

Message 1: 00100 xxxxx

Message 2: 00010 xxxxx

here: one message should “win”

immediate comparison,

either 0 or 1 should be “dominant”

sender of the non-dominant message sees

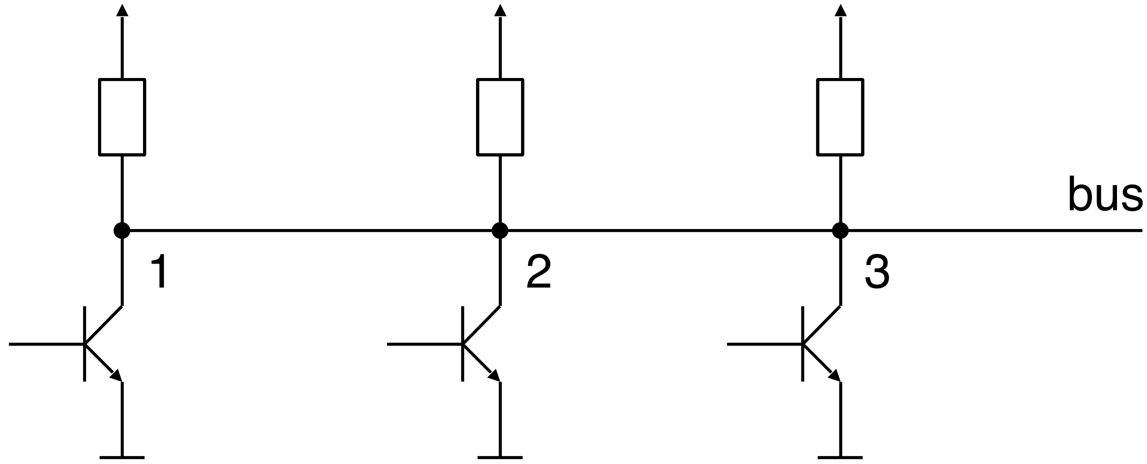
message on bus different from the one sent

Bus arbitration with CAN (priority based)

- CSMA/BA (carrier sense multiple access/bitwise arbitration)
- Bit-wise collision resolution using message identifiers, sent from HSB to LSB
- Wired "and", implemented with bus drivers

  0 - Dominant level

  1 - Recessive level



- Node stops if a “0” is seen when sending a “1”

CAN message

id: 11	control: 6	data up to: 64	crc: 16	various: 11

- Bit stuffing to allow synchronization at bit level
  - after 4 '0's or 4 '1's: dummy 1 or 0

- Transmission time of a message in bit-times:

=> 13+34+databits+ (34+databits-1)/4

Overhead: 44 Bits + ...

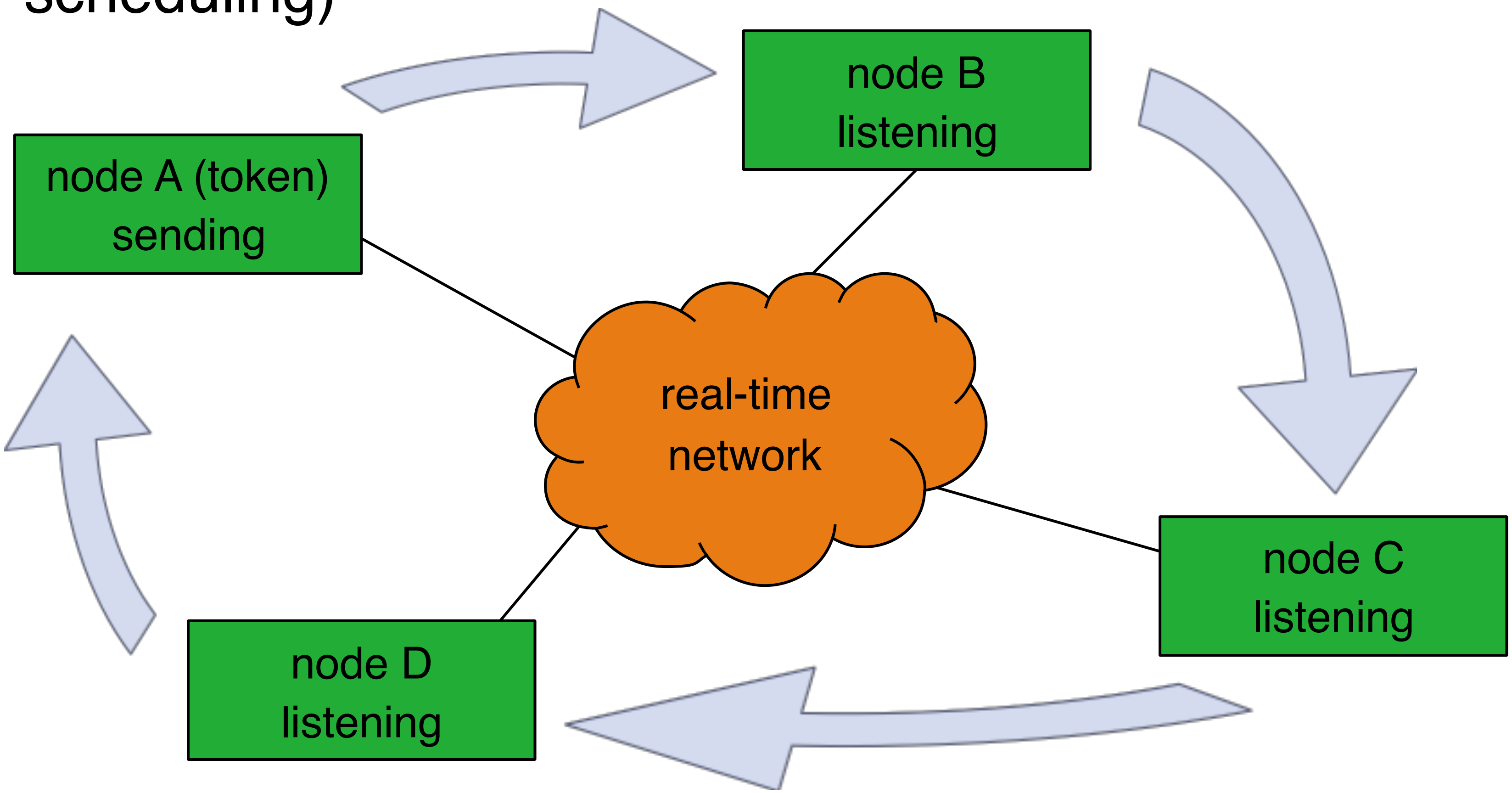
CAN performance

- Transmission rate from 5KBit/s to 1MBit/s
- Bit-synchronized bus access
- Bus length depends on transmission rate
  - 40m with 1MBit/s, 1000 with 50KBit/s
- transmission rate increases with shorter busses
  ... or with increased speed of light

- 2048 priorities (11 Bit) in version A,
  >500 Mio priorities (29 Bit) in version B
- Short messages (0-8 Byte)
- Efficiency depends on msg length: 0%..47%

Medium access control using a token ring

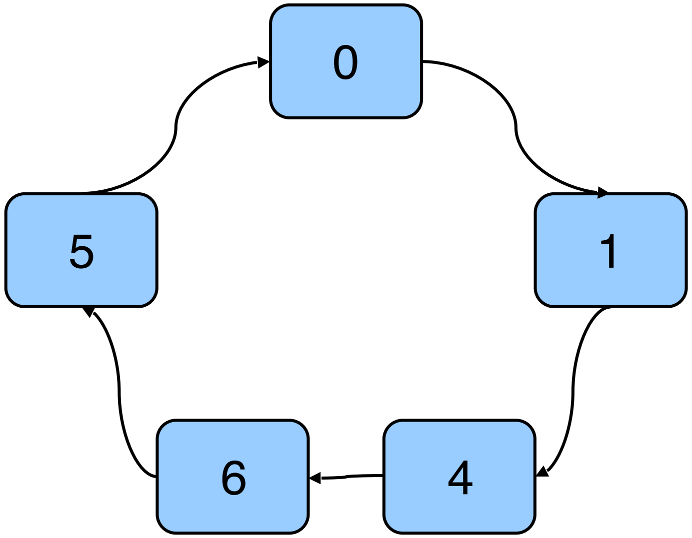
- Messages follow a logical ring
- Token required as permission to send (round robin scheduling)



Token Ring with priorities, simplified

- Frames: Data, Token
- **Token**: Token Priority field, Reservation Priority field
- **Data**: Data field, Reservation Priority field
- Token can be taken only, if Token Priority is lower than priority field in outgoing data frame
- Reservation Priority is set to highest priority of waiting messages that are passed by packet
- Upon arrival of a data frame at its sender, a token is generated with priority is set to the data frame’s priority reservation field

Example (simplified token ring)



Notation:
„Data, ResPrio“
„T-Prio, ResPrio“
0,1,4,6 want to send.
Token T-0,0 arrives at 0
NodeNr = Prio
High number → high prio

0	sends:	Data,0
1	waits, changes Resprio:	Data,1
4	waits, changes ResPrio:	Data, 4
6	waits, changes ResPrio:	Data, 6
5	does nothing	
0	creates token:	T-6, 0
1	waits, changes ResPrio:	T-6, 1
4	waits, changes ResPrio:	T-6, 4
6	claims token, sends:	Data, 4
6	creates token:	T-4, 0
1	waits, changes ResPrio:	T-4, 1
4	sends, sets ResPrio	Data, 1
4	creates token:	T-1, 0
1	sends,	Data, 0
1	creates token:	T-0, 0

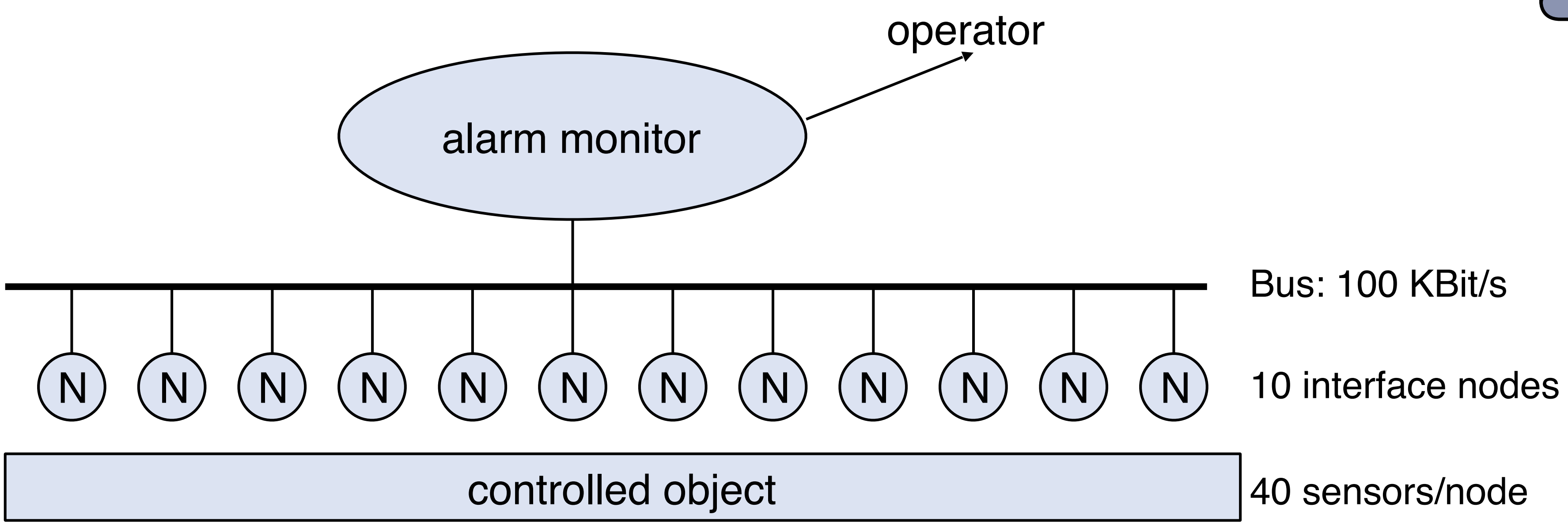
More on Token Ring

- Not necessarily a strict ring
- Functional Parameters
  - Maximum token holding time at each node
  - Rings: resulting maximum token rotation time
- Problems:
  - lost tokens
  - duplicated tokens

Time division multiple access (TDMA)

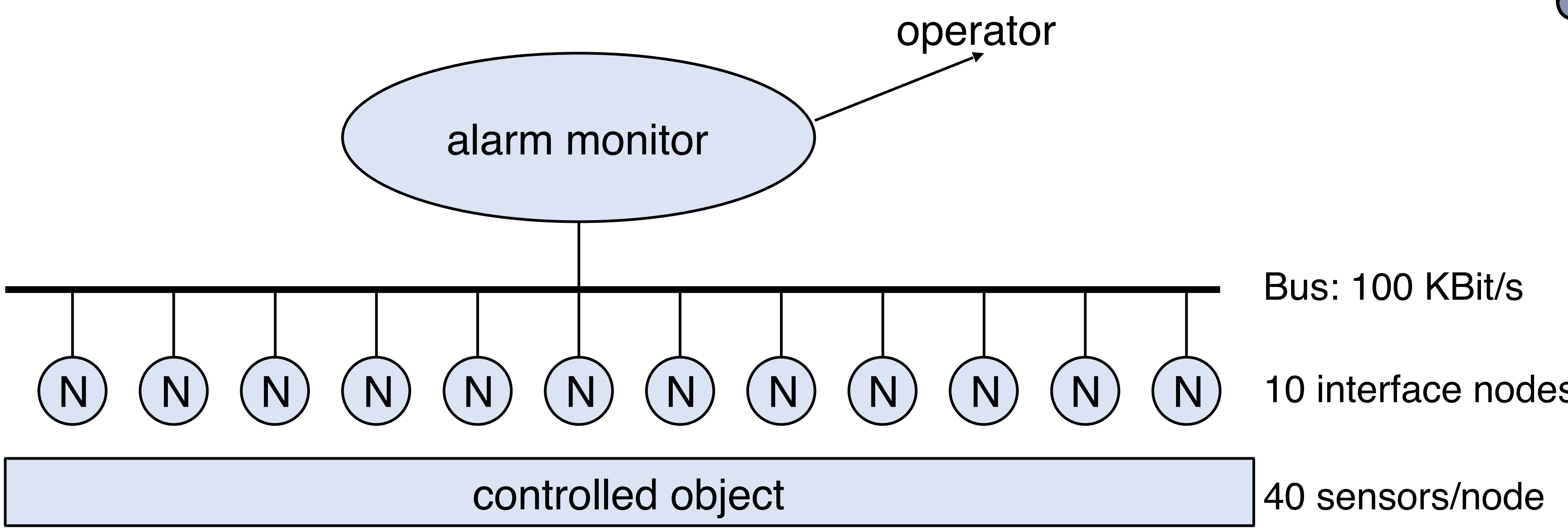
- Divide Time periodically into Slots
- Allocate Slots
- Slots:
  - Cooperative (BUT: failures)
  - enforced

Kopetz: event vs. time triggered @ Peak Load



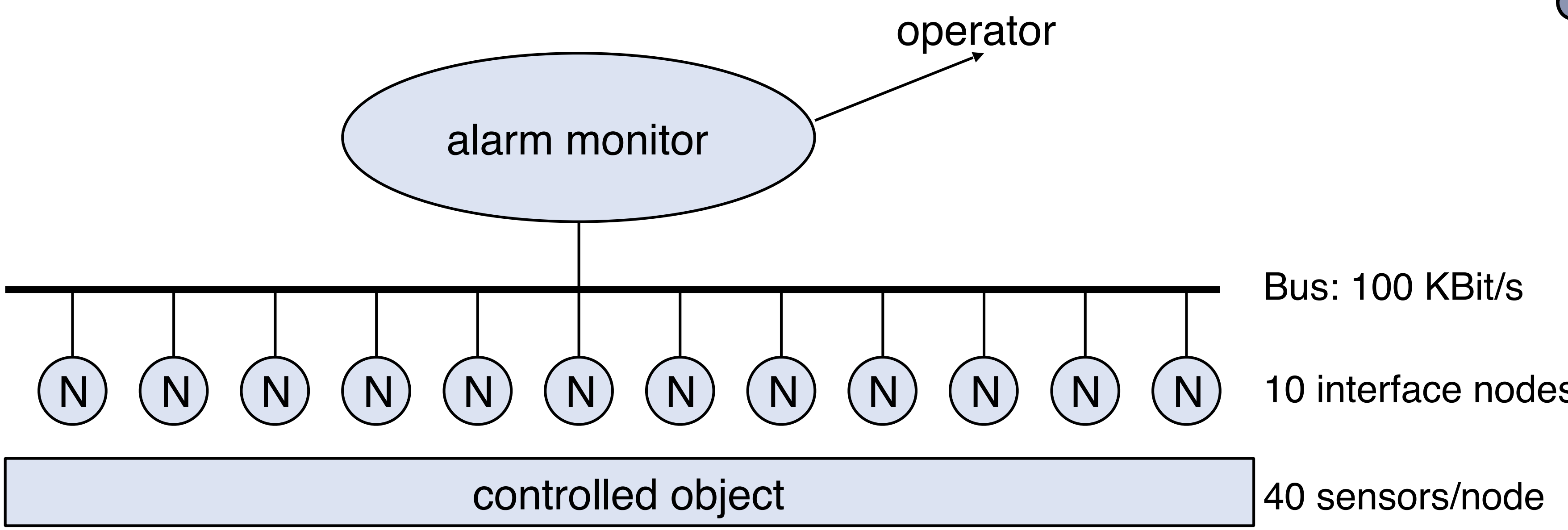
- Each node supervises 40 alarm conditions and sends messages
- Deadline: 100 ms to notify operator
- Communication bandwidth: 100 kbits/second

Kopetz: event triggered (ex. CAN)



- Send message as soon as alarm condition is seen
  size = alarm name (8 Bit) + CAN overhead (44+4 Bit) = 56 Bit
- allows 180 messages within deadline of 100ms
- worst case requirement: 400 messages ...

Kopetz: event triggered (TDMA bus)



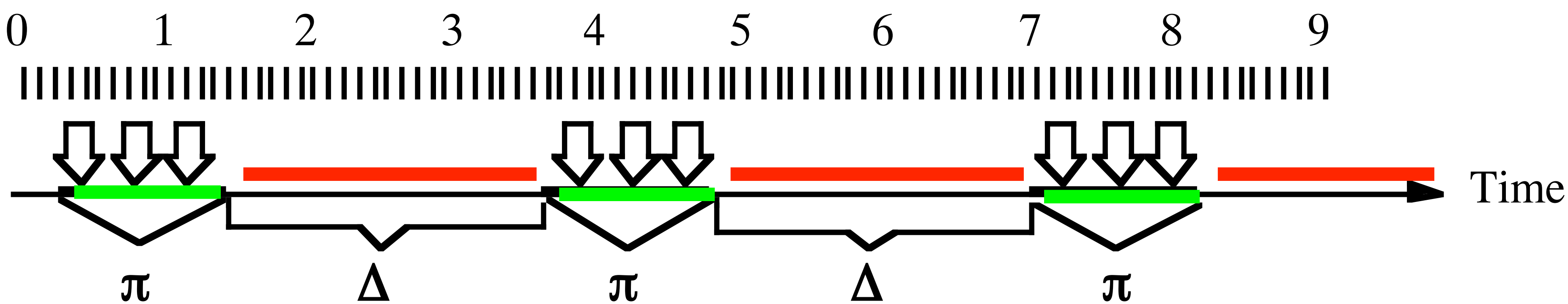
- Each node sends “state” message every 100ms
  size: Data field (40 Bit) + overhead (44) + gap (4) = 88 Bit
- allows 110 messages within 100ms period
- but, just 10 required!
  - i.e. 10 % of bandwidth

TT Ethernet as an example for TDMA

- Participants request slots
- Switch enforces slots
- Based on sparse time (determinism)

- TT-Ethernet set of slides taken from
  Keynote RTSS 2008 by Hermann Kopetz
  with permission

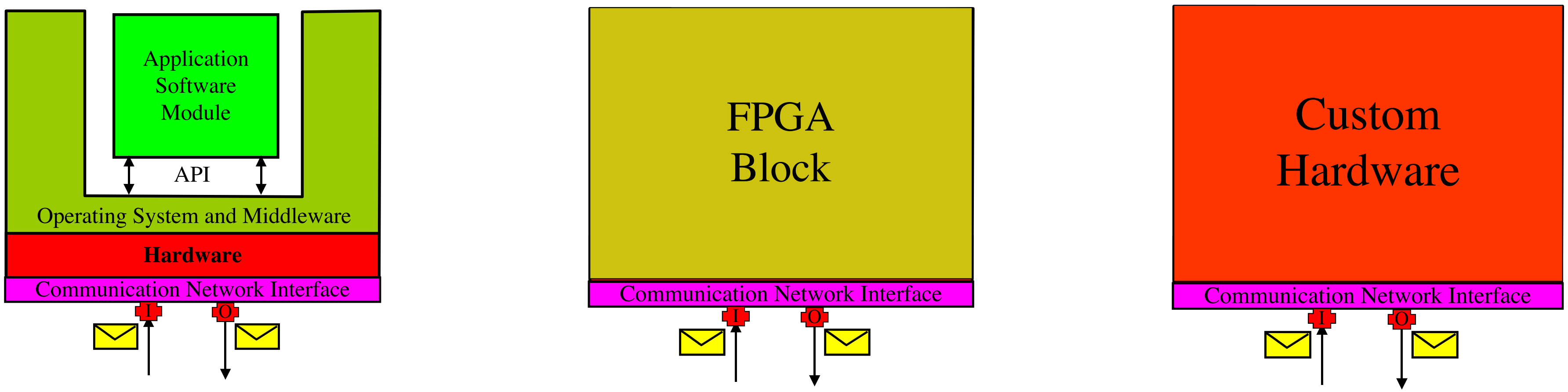
use sparse time



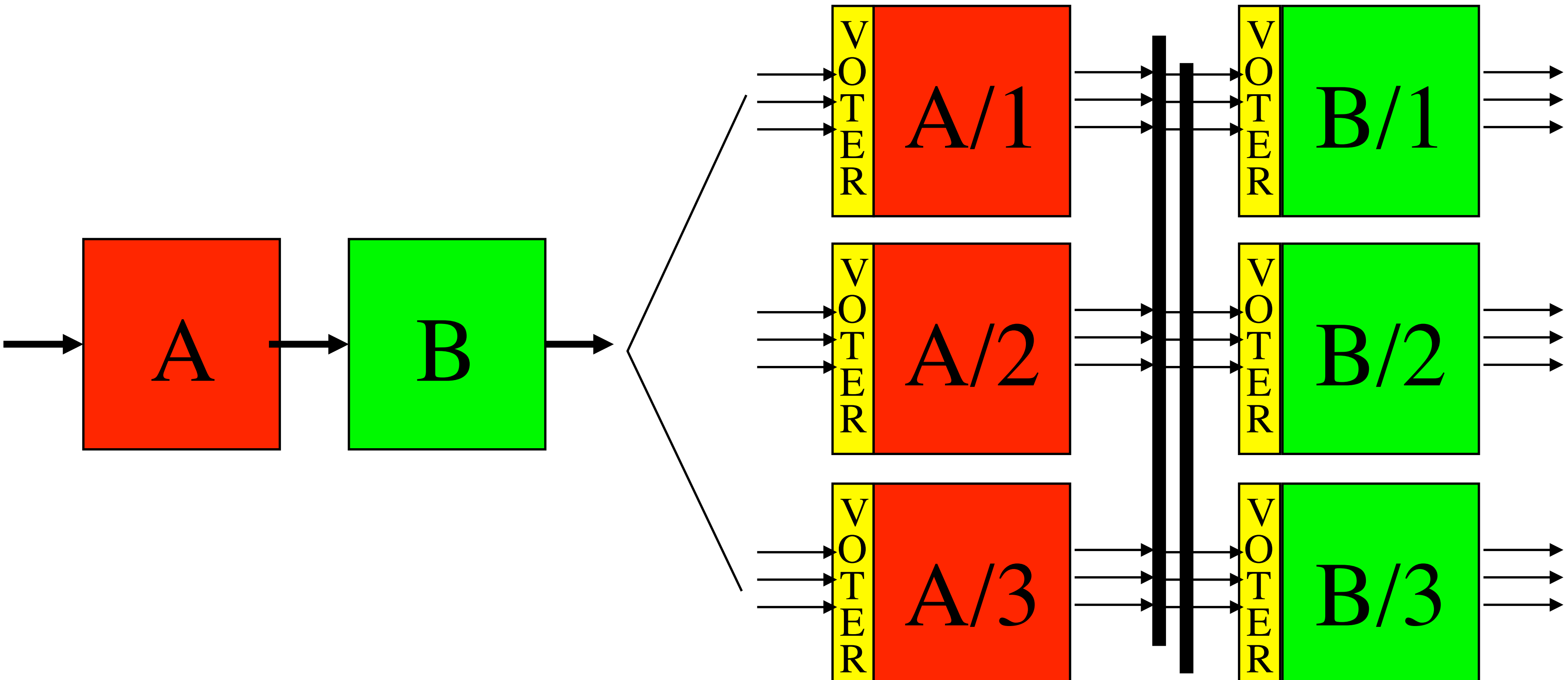
Events ⇩ are only allowed to occur at subintervals of the timeline

represented in TTE thru a standardized time format

RT Components



TMR for fault handling



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.

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

Preamble (7 bytes)
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)

Standard Ethernet Message Header

Conflict Resolution in TT Ethernet

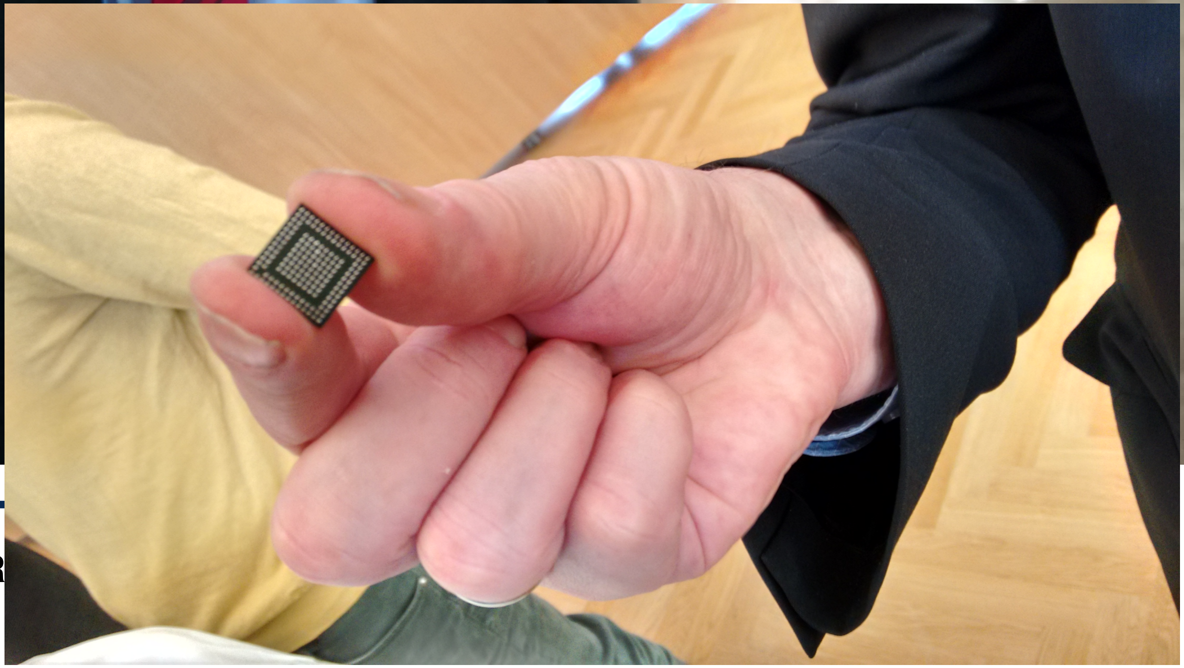
- TT versus ET:
  TT message wins, ET message is interrupted (preempted). The switch will retransmit the preempted ET message autonomously
- TT versus TT: Failure, since TT messages assumed to be properly scheduled (closed world system)
- ET versus ET: One has to wait until the other is finished (standard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

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

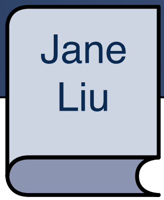
Global Time

- 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

Der Meister selbst



Real-Time communication in WANs



Principles

- "find" connection (reservations)
- determine local properties at each node of connection depending on node-local scheduler
- add up delays
- take care for jitter (provide buffer, traffic shapers)

Example in this lecture: Weighted Round Robin

Other:

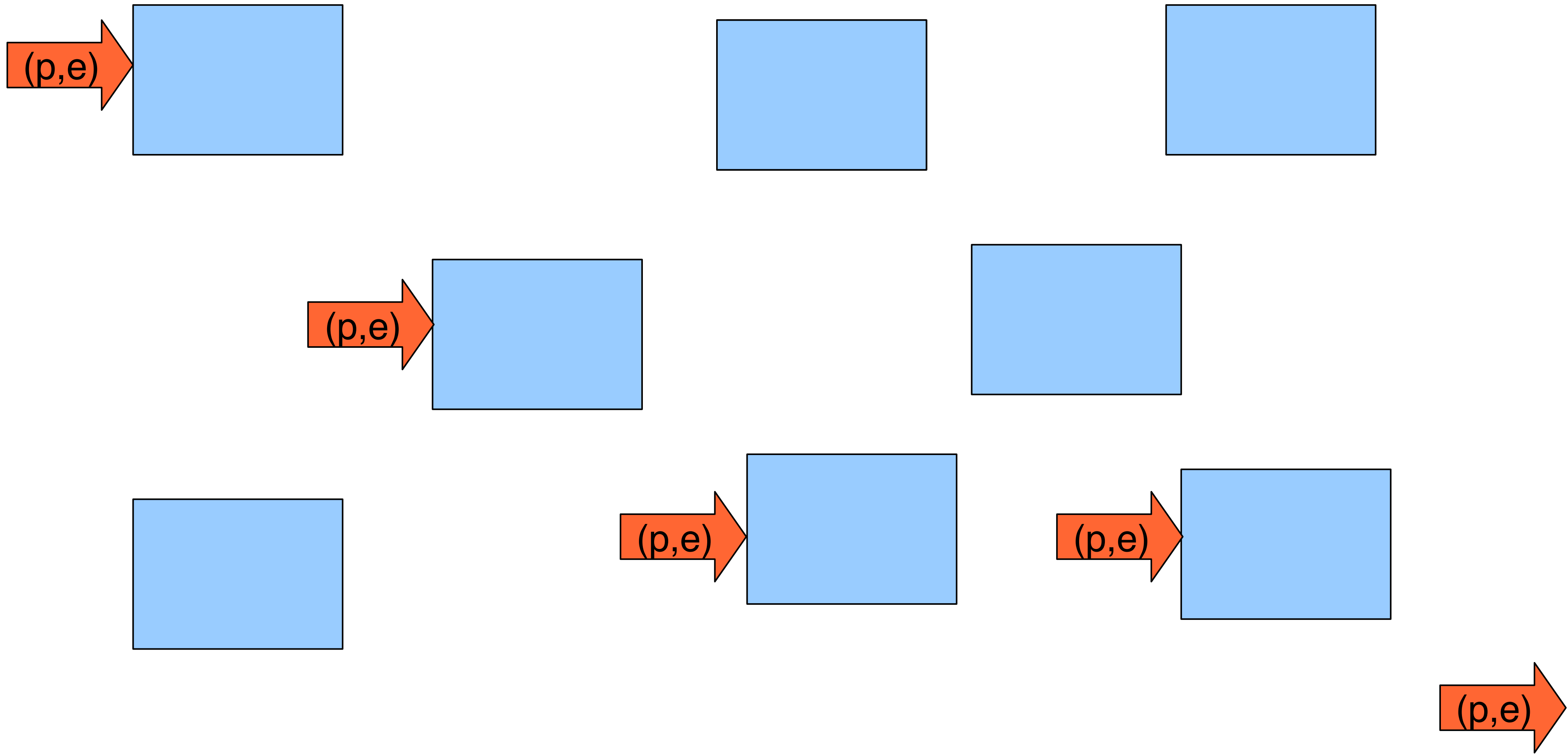
- Dynamic, EDF: e.g., Delay-EDD "Service Discipline"

See text books

Scheduling example: reservation

Jane Liu

(p,e)



Weighted Round Robin

Jane Liu

Principle:

- Message passes multiple nodes from source to dest.

During set up (admission) a *weight(wt)* is assigned to each input buffer of all involved switches

- Scheduler at each node
  - Visits input buffers in round robin
  - processes *wt* number of units of messages at each buffer per round

Scheduling example: Weighted Round Robin

Jane Liu

Messages:

- e_i max „length" of message in "units"
- p_i Period: minimum inter-arrival time

- ρ number of hops
  nodes passed by message from
  source to destination

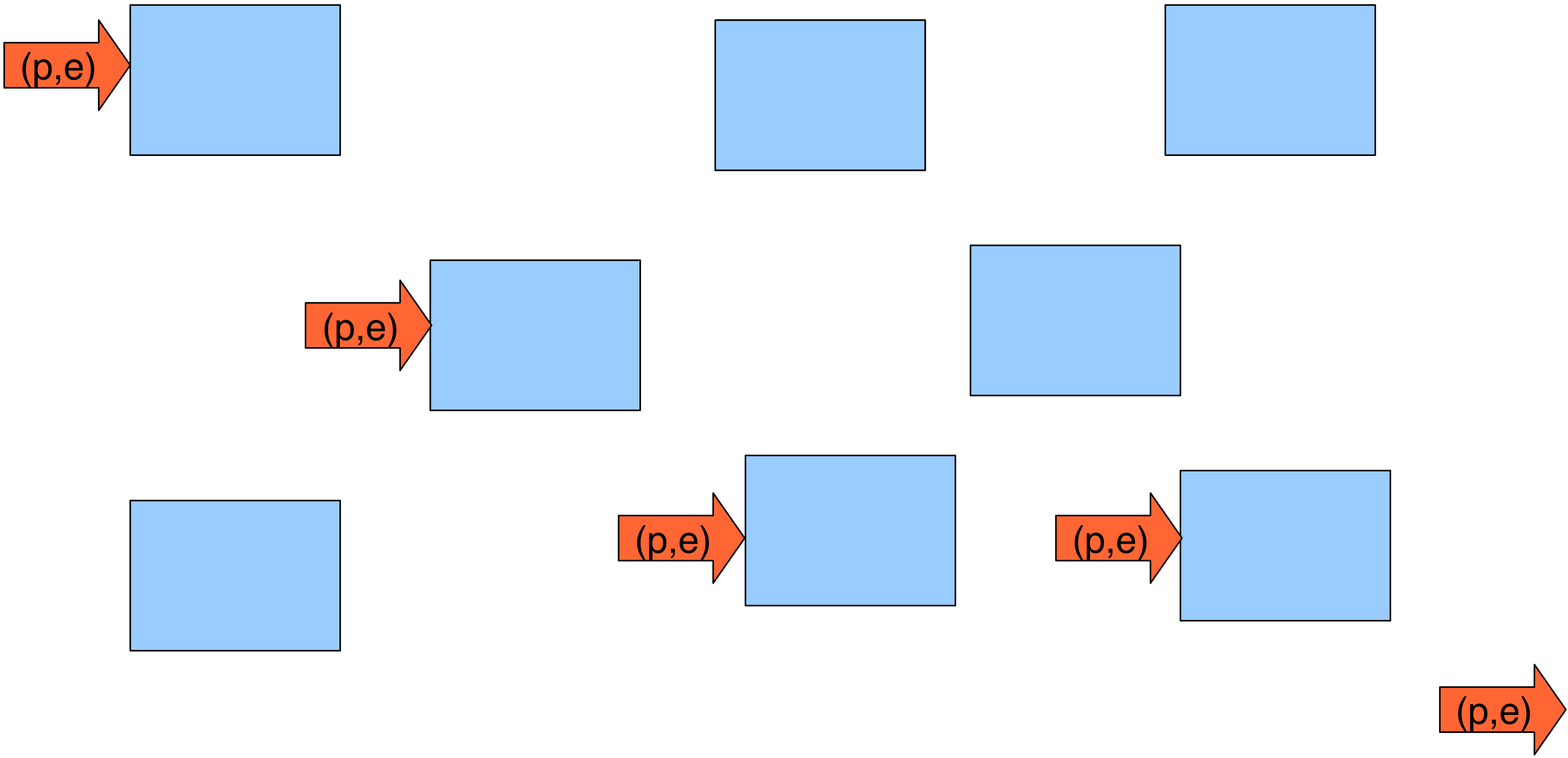
how to guarantee

- Bandwidth (p_i,e_i) and
- end-to-end delay

Scheduling example: reservation

Jane Liu

(p,e)



Weighted Round Robin

Jane Liu

Nodes:

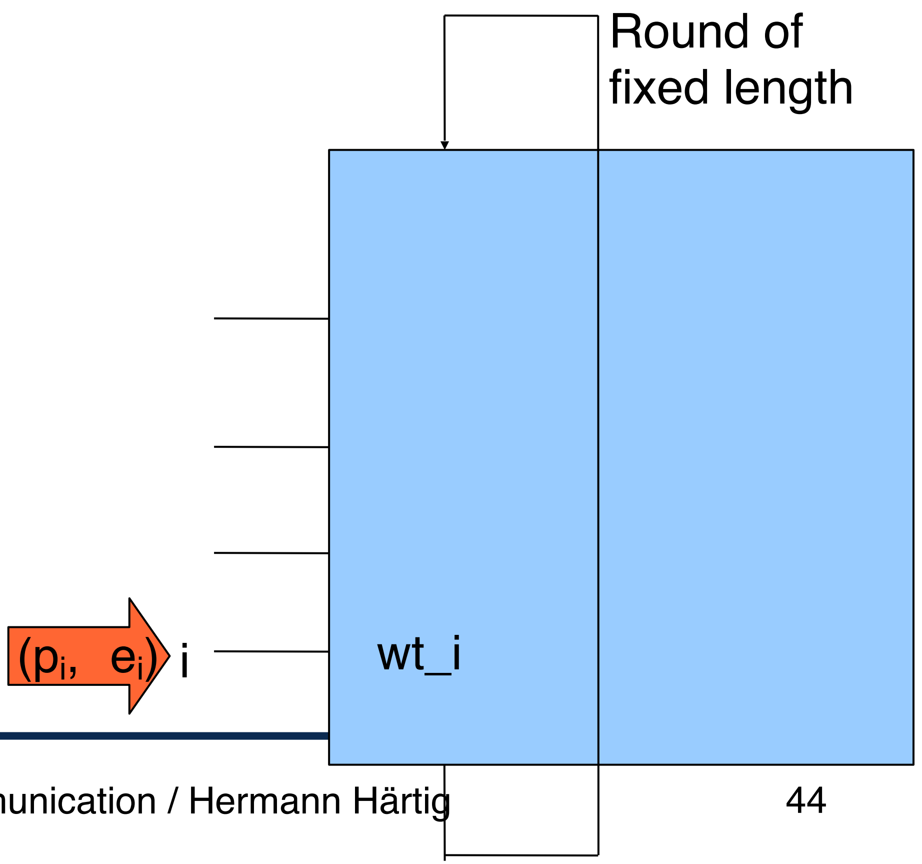
wt_i	each connection is given wt_i slots per round.
RL	round length (switch design parameter)
e_i	max „length“ of message in units
	if wt_i=1 → e_i rounds at the switch are needed

$\Sigma\, wt_i <= RL$

Round lengths ?

Weights?

Delay? Per Node, end to end?



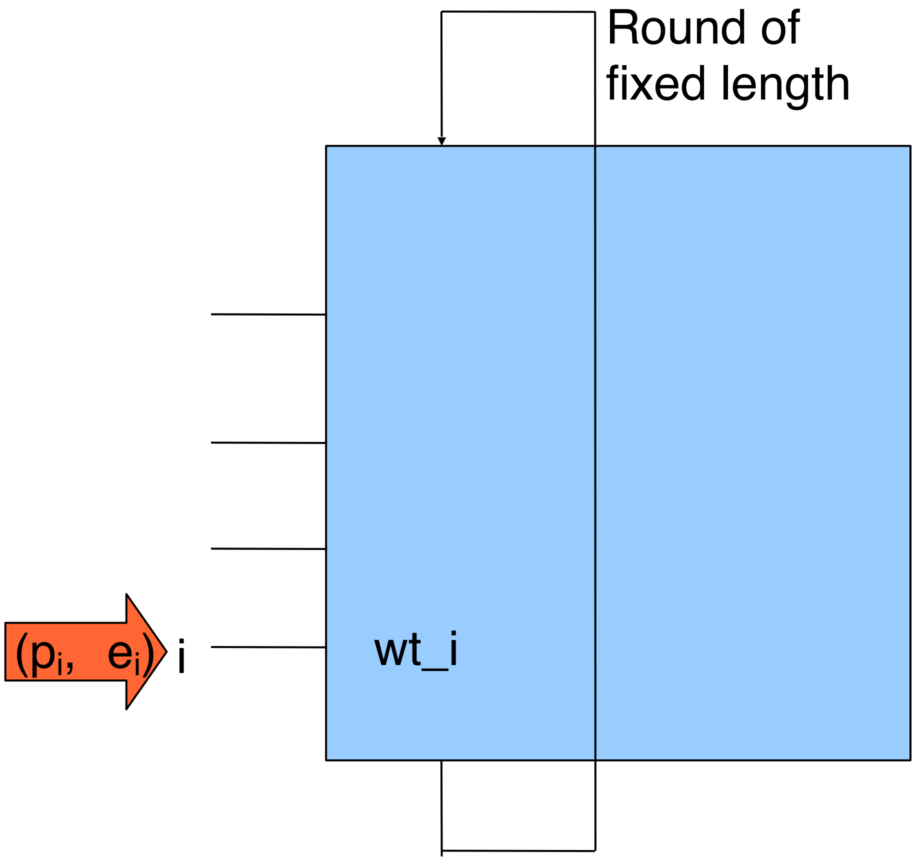
Scheduling example: Weighted Round Robin

Jane Liu

In each round, wt_i message units are taken from input i

Assign wt sufficient for (p,e)

Examples RL==16 :
(p,e)
- (08, 1) not possible
- (16, 4) → NoR: 1 → wt = 4
- (32, 4) → NoR: 2 → wt = 2
- (32, 5) → NoR: 2 → wt = 3
- (31, 5) → NoR: 1 → wt = 5



General:
- RL <= min p_i
- Σ wt_i <= RL
- $wt_i >= \lceil e_i / \lfloor p_i / RL \rfloor \rceil$

Scheduling example: weighted round robin

Jane Liu

Some Calculations:

Delay per Node:

$RL * (\lceil e_i / wt_i \rceil)$

RL is bound by delay guarantees of Node for example, to guarantee that messages are transmitted within a period:

$RL <= \min p_i$

$wt_i >= \lceil e_i / \lfloor p_i / RL \rfloor \rceil$

end to end delay (pipelined "execution"):

$(\lceil e_i / wt_i \rceil + \rho - 1) * RL$

Summary

- Field busses
  - Scheduling analysis analog to processors
    - Use time driven / static priority / table based scheduling

- Switched LANs

- WANs
  - Dynamic
  - Use local scheduling disciplines in switches/routes to enforce system wide behaviour
  - Admission for connections required