Distributed Architectures for Embedded Systems:

Challenges for Real-Time Control

HSCC 2009

Albert Benveniste (INRIA-IRISA, Rennes)

prepared with Paul Caspi, Alberto Sangiovanni-Vincentelli,

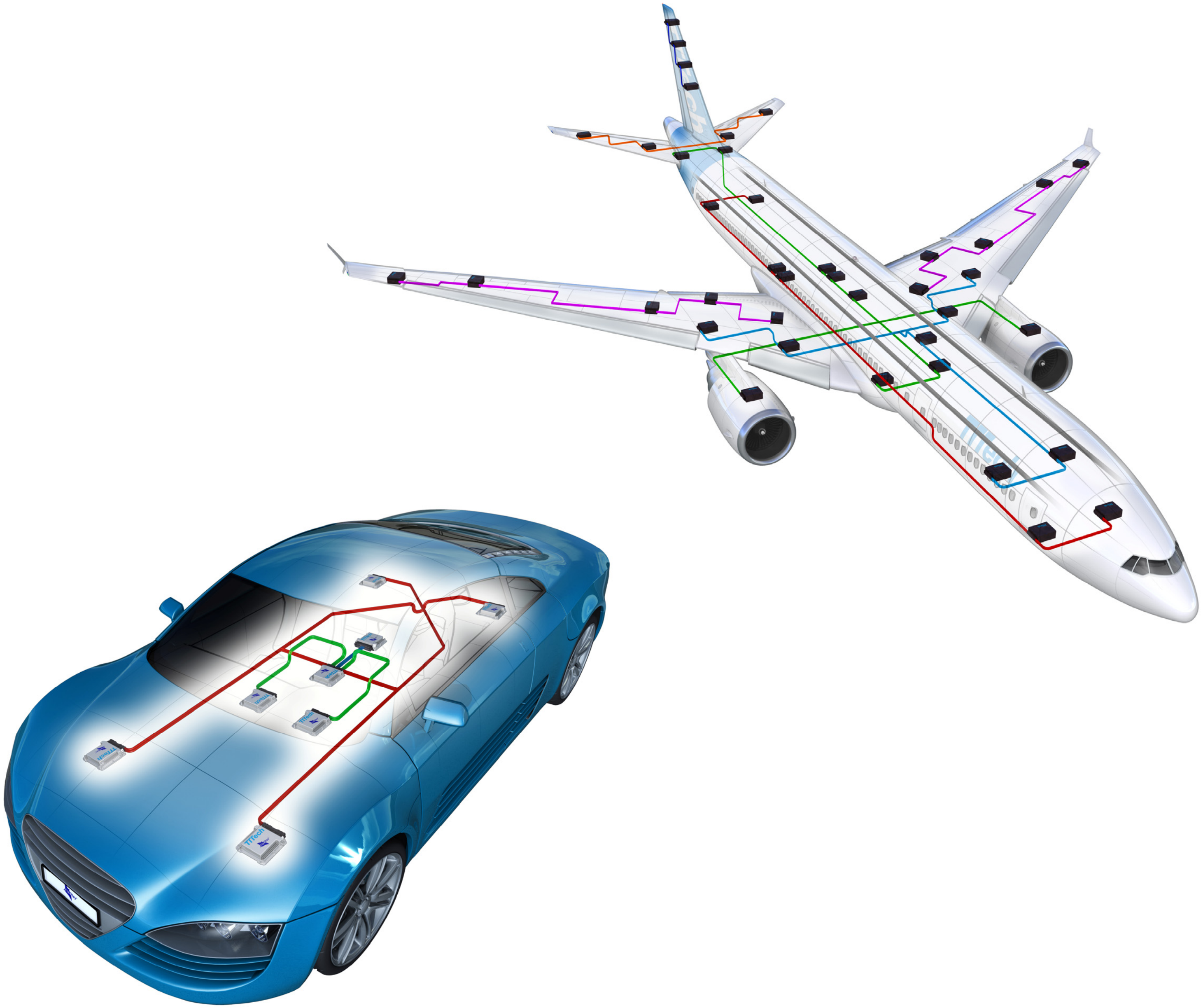
Stavros Tripakis, and Claudio Pinello

Acknowledgement: EU-projects Artist-Design and COMBEST

Preamble: facts from industry

Some facts from actual architectures in use in industry:

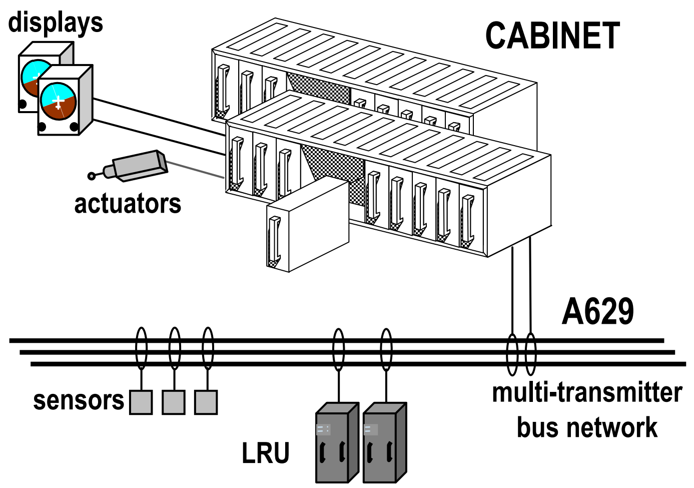
- aeronautics
  - IMA-AFDX from Airbus
  - Boeing-Honeywell IMA architecture
  - Link application-architecture, Rockwell-Collins
- automobile
  - AUTOSAR principles and its RTE
  - TT-Ethernet from Hermann Kopetz
- automatic subways

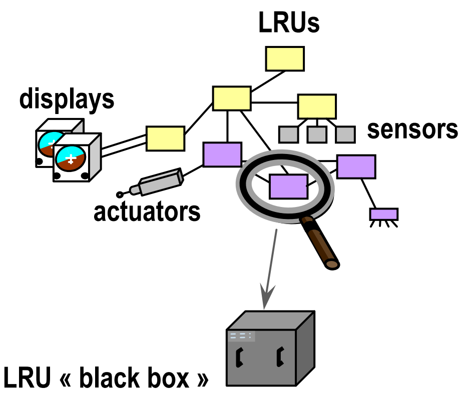

TTTech

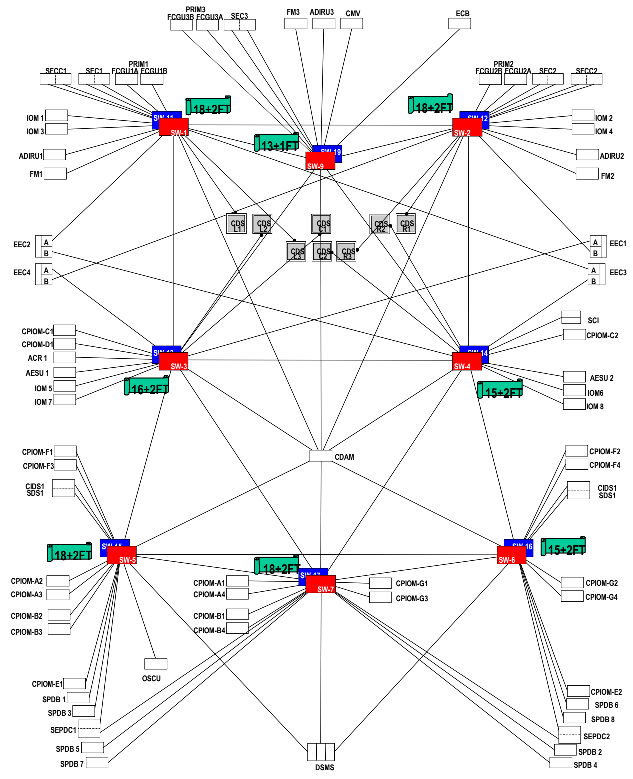
What is A380 IMA?

To Integrated Modular Avionics

From Federated Architectures

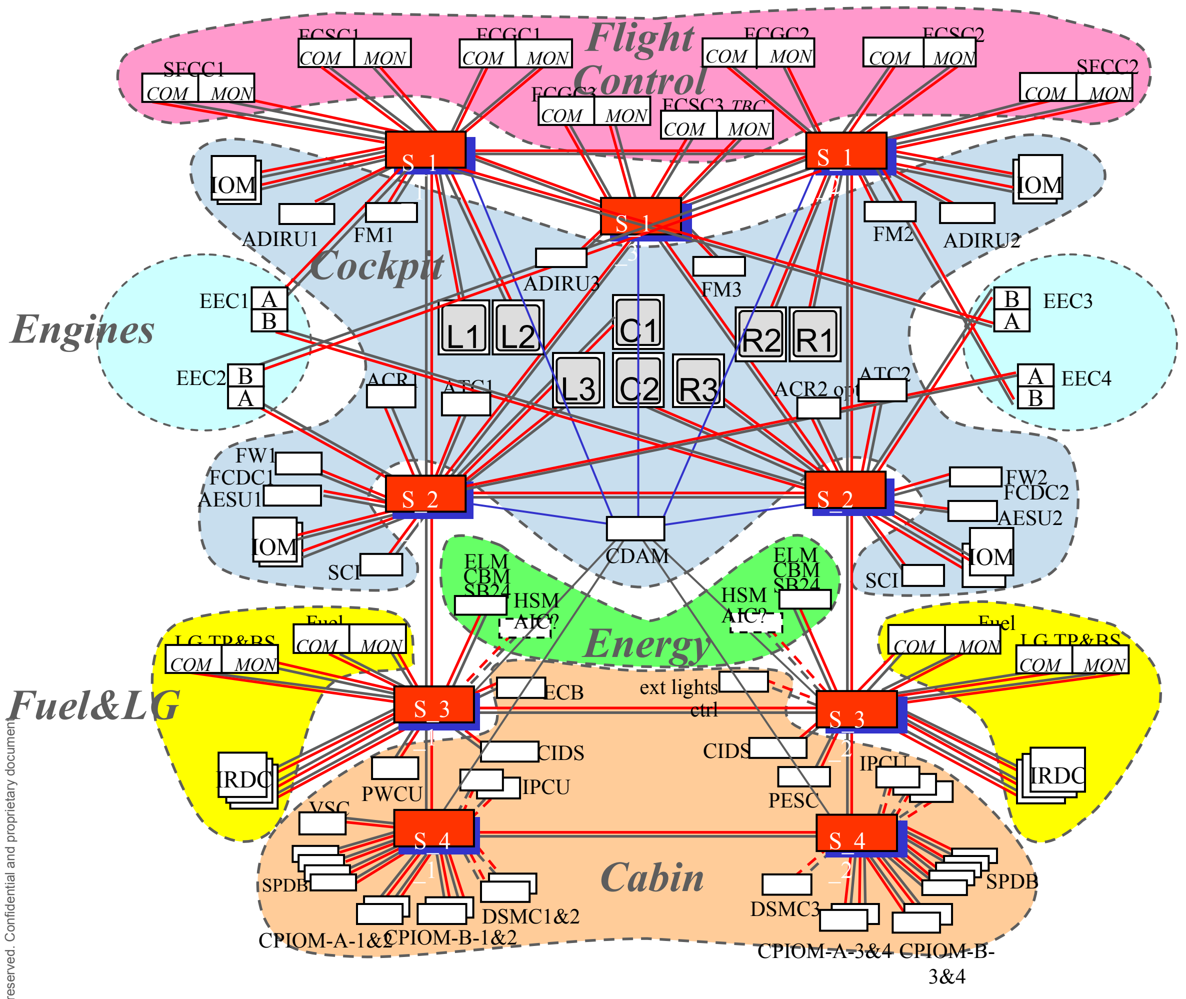






AFDX NETWORK

What is A380 IMA? - ADCN Network & Topology

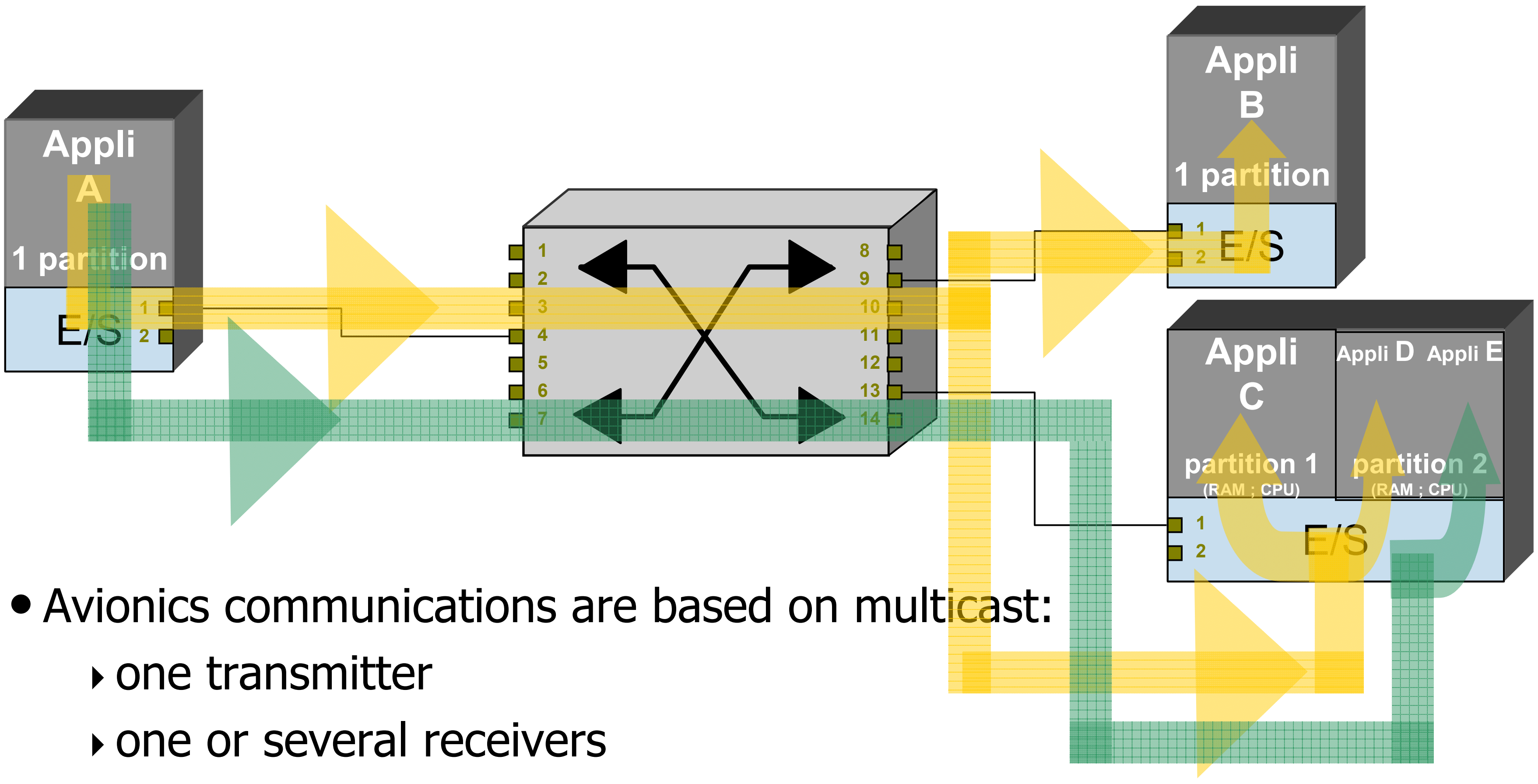


AFDX Network:

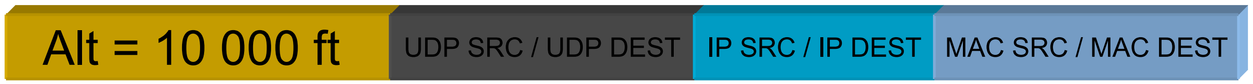
- 100 Mbits
- Redundant Network (A&B) with independent alimentation
- AFDX switches = 2 x 8
- NB of ports (connections) possible on each switch (20-24)
- MTBF of the switch is very high (100 000 hours expected)
- Up 80 AFDX subscriber

AIRBUS

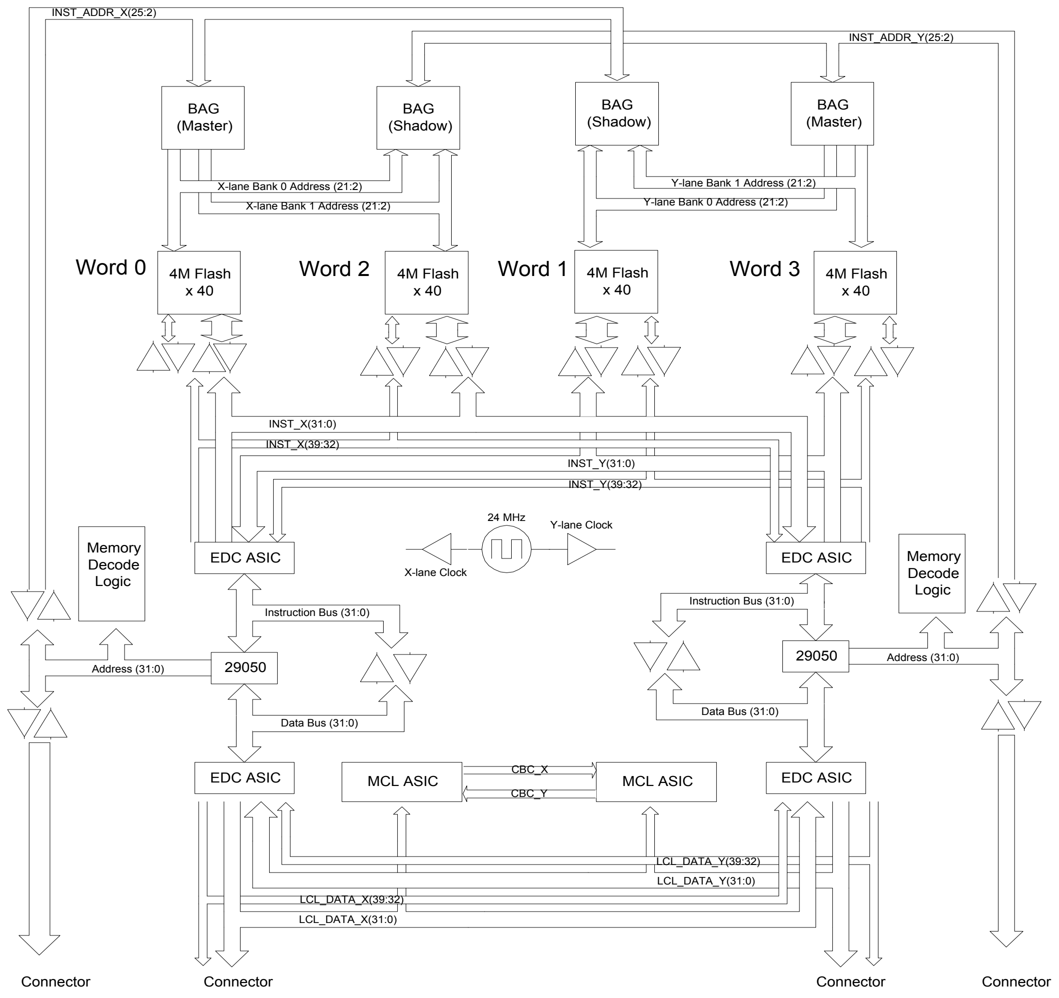
AFDX technology – Addressing : MAC,IP,UDP



- Avionics communications are based on multicast:
  - one transmitter
  - one or several receivers
- Asynchrony of individual clocks
- NO reconfiguration capability in the AFDX network

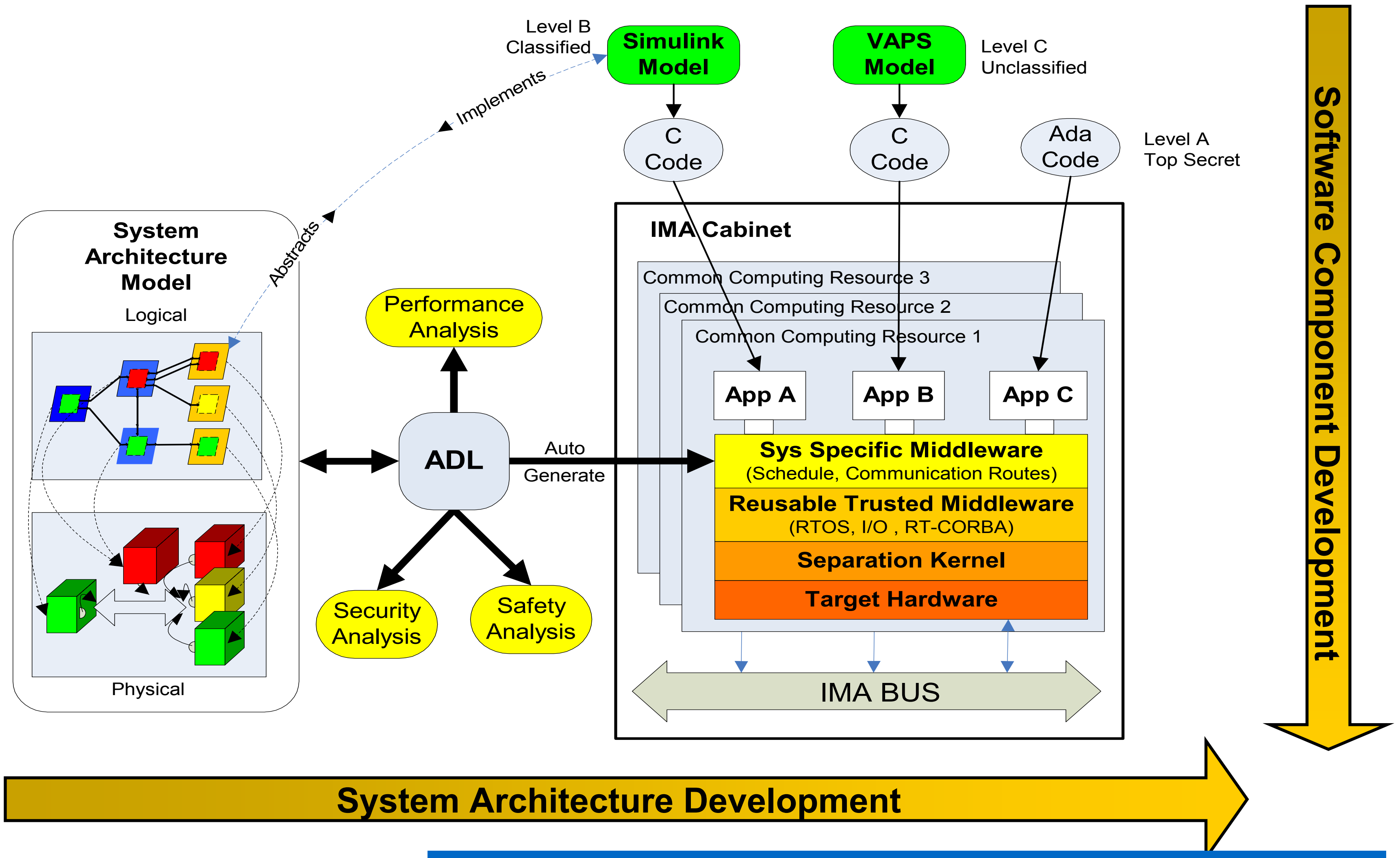


Lock-Step Processor Architecture



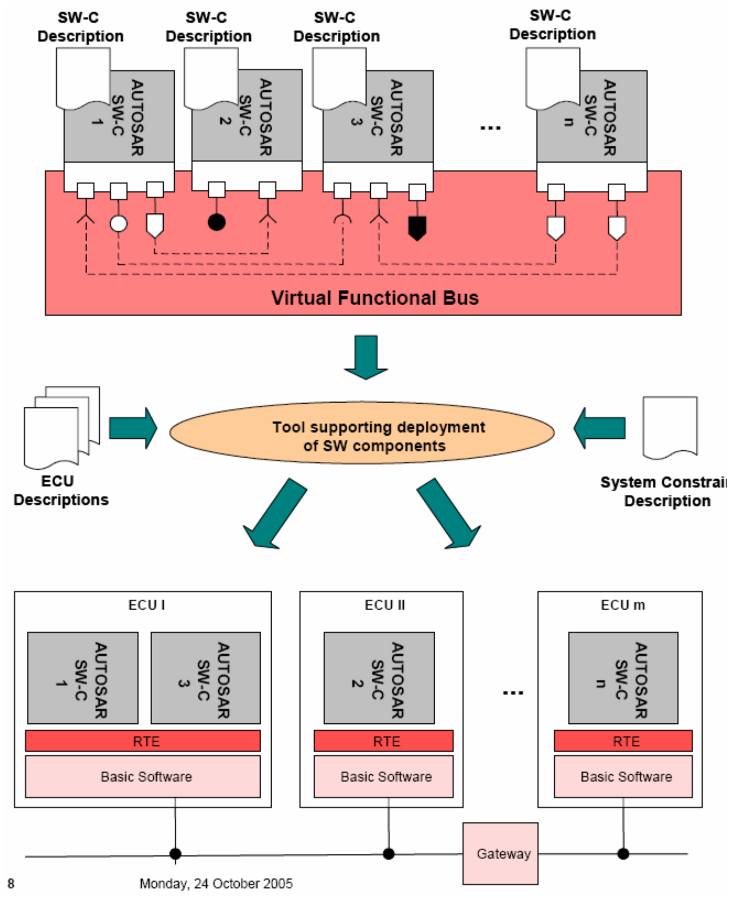
- **Space partitioning**
  - Protected system page tables
    - Constructed at build time
  - Validated MMU
- **Time partitioning**
  - Non-user maskable SAFEbus interrupt drives OS schedule
  - No other interrupts allowed
- **Fault effects containment**
  - Lock-step checking of all memory accesses (I and D)
  - EDC on all memory R/W
  - Monitored clock, power
  - Power up BIT, CBIT

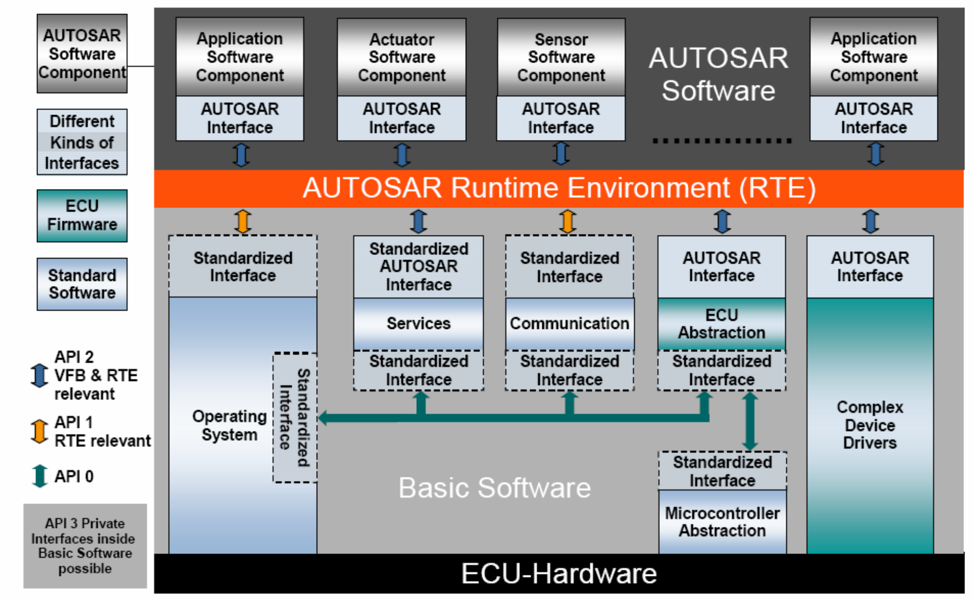
System Architectural Modeling & Analysis



Key AUTOSAR "Methodology and RTE"

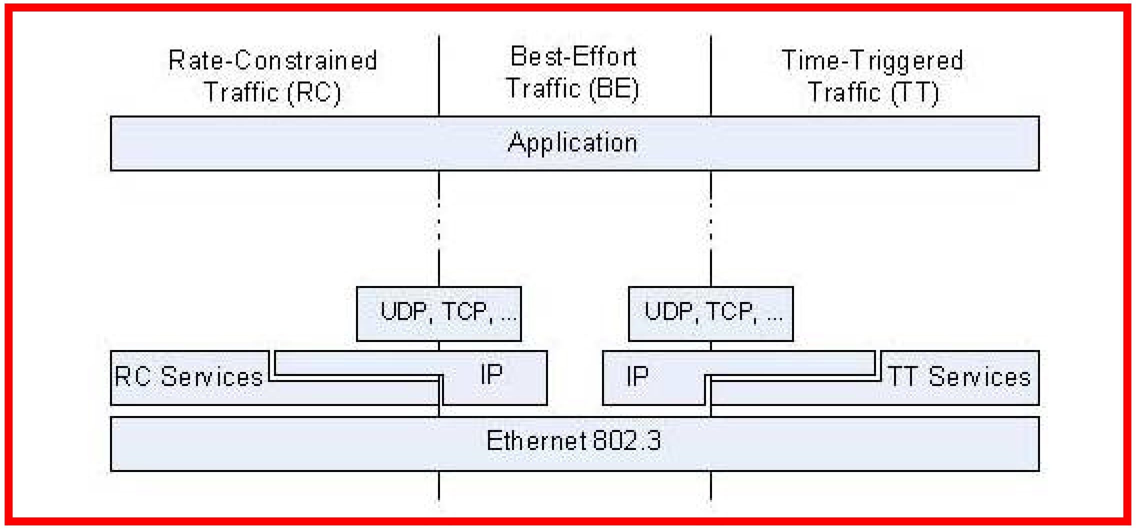
- Flexible mapping of software components ...
- ... enabled by standardized run-time environment (RTE)



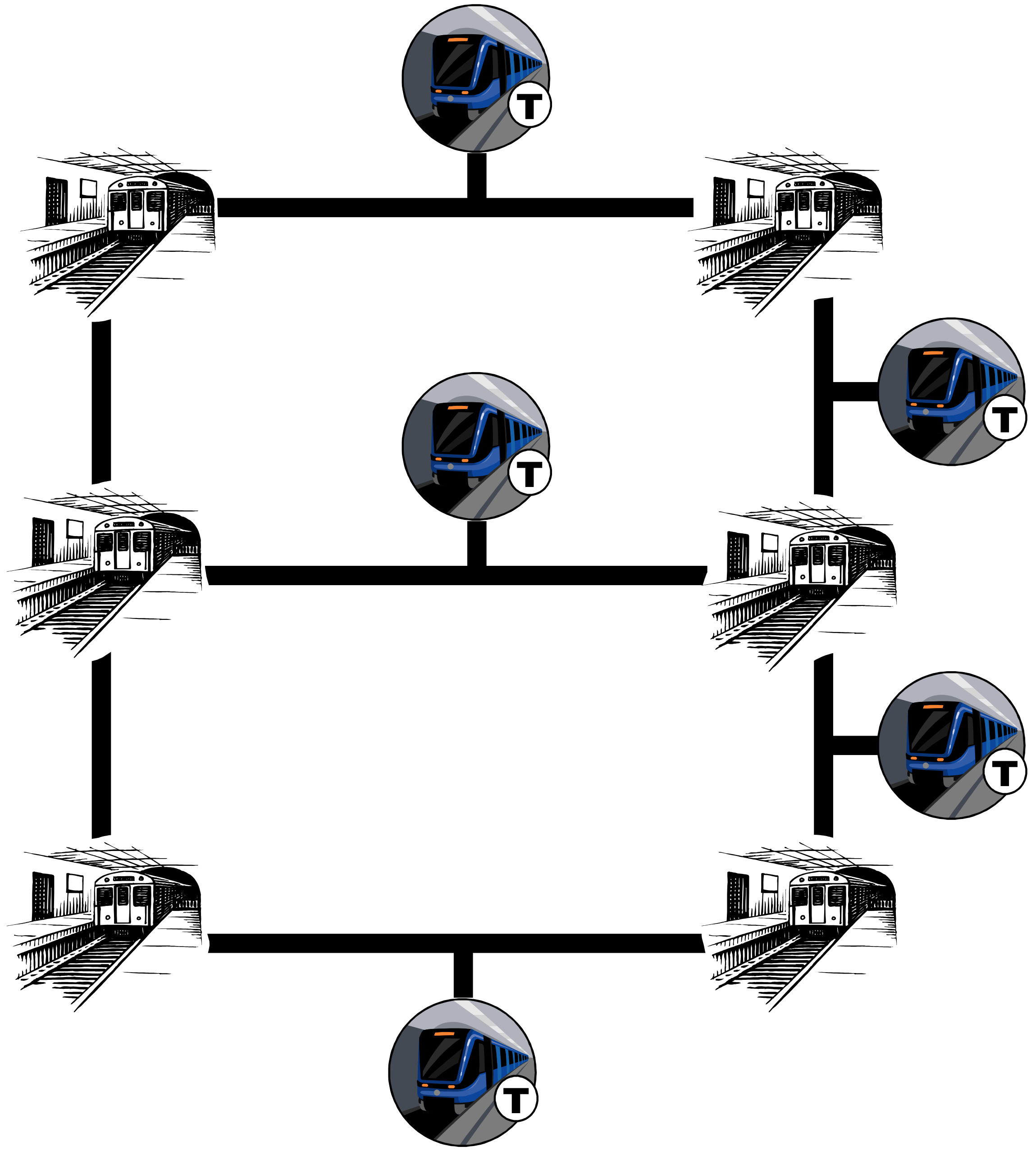


TTTech

As TTEthernet supports communication among applications with various real-time and safety requirements over a network, three different traffic types are provided: time-triggered (TT) traffic, rate-constrained (RC) traffic, and best-effort (BE) traffic. If required, the corresponding traffic type of a message can be identified based on a message's Ethernet Destination address. The relation of the TTEthernet traffic types to existing standards is depicted in Figure 2.



Messages from higher layer protocols, like IP or UDP, can be "made" time-triggered without modifications of the messages' contents itself. The TTEthernet protocol overhead is transmitted in dedicated messages termed protocol control frames, which are used to establish system-wide synchronization. In short, TTEthernet is only concerned with "when" a data message is sent, not with specific contents within in a message.



Computers on trains for speed control

Computers on tracks for collision avoidance and to avoid losing a train (ghost train!!)

MBPC

Wired communications for fixed computers

For computers on trains: use wheels or wireless

Communication by Sampling (LTTA)

Preamble: facts from industry

Reasons for chosing a particular distributed architecture:

- distributing intelligence over system architecture, physical reasons
- resilience, fault tolerance, segregation & compartmentalization
- distributed development process with OEM & suppliers

Preamble: facts from industry

Reasons for chosing a particular distributed architecture:

- distributing intelligence over system architecture, physical reasons
- resilience, fault tolerance, segregation & compartmentalization
- distributed development process with OEM & suppliers
- some less important issues:
    - power (of increasing importance, however)
    - memory, bandwidth

Preamble: facts from industry

Reasons for chosing a particular distributed architecture:

- distributing intelligence over system architecture, physical reasons
- resilience, fault tolerance, segregation & compartmentalization
- distributed development process with OEM & suppliers
- some less important issues:
  - power (of increasing importance, however)
  - memory, bandwidth

Control design must cope with these constraints

Problems for control design

- Embedded systems distributed architectures raise other issues than
  - limited Shannon budget and
  - cost-to-communicate

Problems for control design

- Embedded systems distributed architectures raise other issues than
  - limited Shannon budget and
  - cost-to-communicate
- Distributed control architectures cause artifacts that can be problematic for feedback control
- Systems architectures such as IMA and AUTOSAR aim at enabling modular development of systems in complex supplier chains. Are modular design techniques available for control algorithms?

Problems for control design

Distributed sensing & computing & actuating architecture $+$ communication media cause the following artifacts:

- clock jitter & drift
- $\Rightarrow$ duplications, losses

Are these artifacts covered by state-of-the-art robust control design techniques?

Problems for control design

Are modular design techniques available for control algorithms?

Stability and performance not compositional $\Rightarrow$ HARD!

Can modular design techniques be developed based on

- passivity,
- Lyapunov,
- LMI,
- new concepts?

This talk: communication artifacts

- Loosely Time-Triggered Architecture (LTTA)
  - used in many industrial plants
- some remarks regarding continuous control
- focus on discrete systems
- dealing with hybrid systems? still open

An often used architecture: LTTA

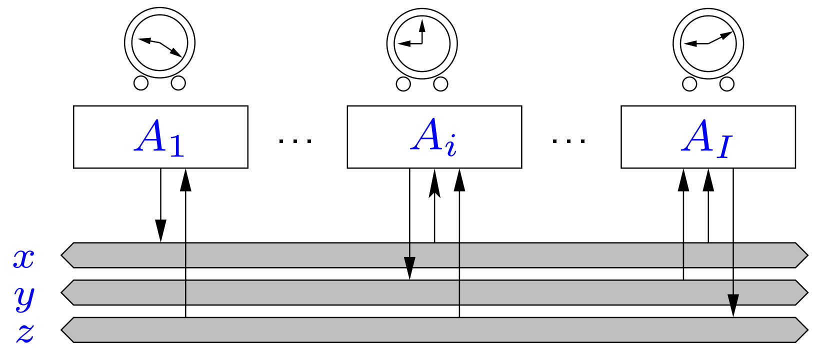
- Synchronous models used for both:
  - *application modeling* — synchronous languages
  - *architectures* — Kopetz' Time-Triggered Architecture

  Advantages: safe programming, safe architecture

- Its use for architectures can be too constraining:
  - oversizing
  - high cost at redesign (budgeting time is global)
  - not even feasible with wireless communications

- $\Longrightarrow$ Loosely Time-Triggered Architecture (LTTA)

An often used architecture: LTTA

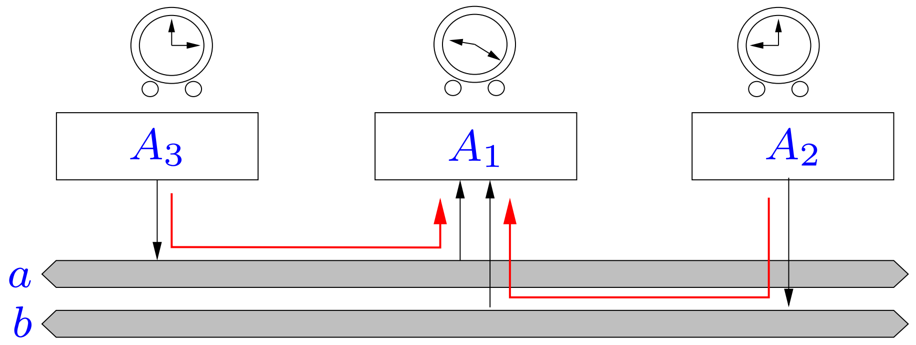


Communication
by Sampling (CbS)

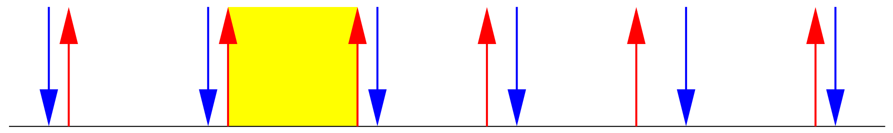
1. The communication medium behaves like a collection of shared memories, one for each variable

2. Each computer periodically samples its external world: physical world and other computers; and so does the communication medium itself. Advantages:
   - communication medium off-the-shelf;
   - autonomy, no deadlock, no livelock;
   - but misses and duplications.

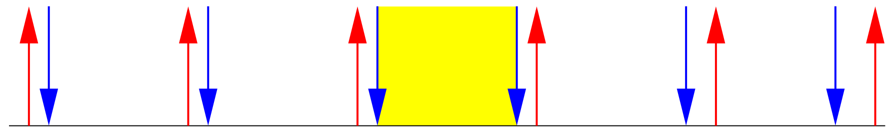
An often used architecture: LTTA



Problems when writing/sensing with non synchronized clocks:

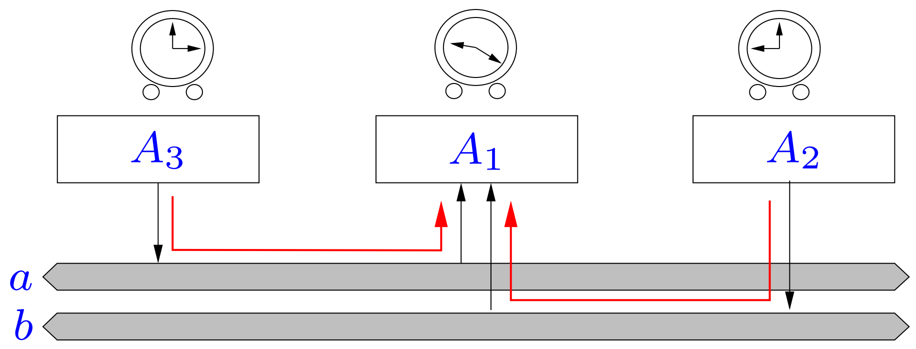


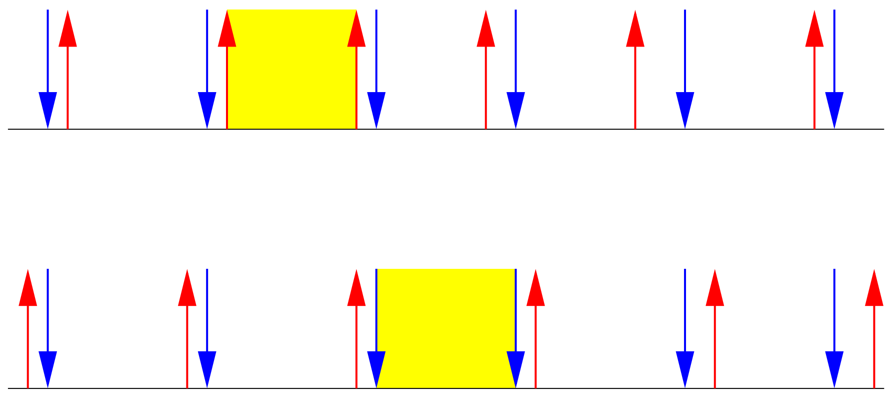
duplications



losses

An often used architecture: LTTA





No harm for continuous feedback control: smoothness should do

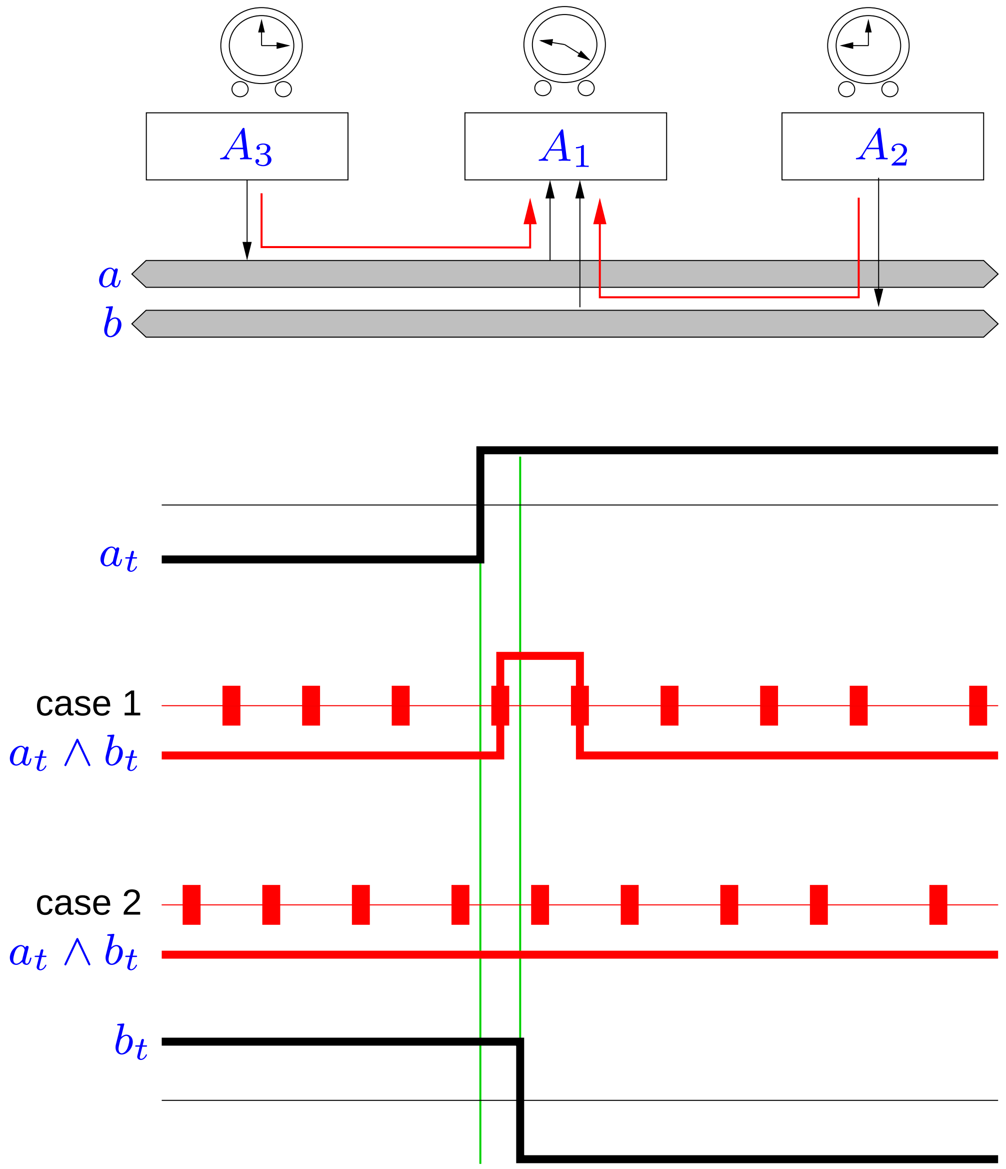
Is it, however, within the scope of H_2/H_∞ or other theories? [Kao-Lincoln 2004]

Exploration tools:

RT-Builder [Geensys]

JitterBug/TrueTime [Arzen]

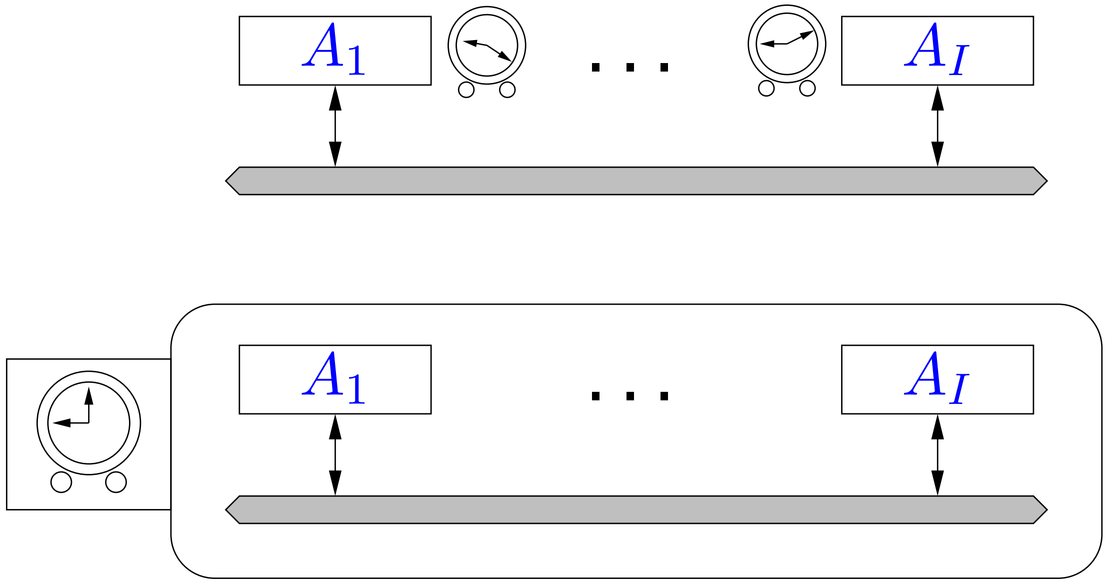
An often used architecture: LTTA



More problems when sensing multiple discrete signals:

Cases 1 and 2 correspond to two different outcomes for the local clock of A_1

Problem setting



The problem: ensuring flow equivalence between LTTA design (top) and strictly synchronous design (bottom).

$$A = \underbrace{N_1 \parallel \ldots \parallel N_n}_{\text{no zero--delay circuit}} \text{ , where } N : \left\{ \begin{array}{rcl} X_k & = & f(X_{k-1}, u_k^1, \ldots, u_k^p) \\ y_k & = & g(X_{k-1}, v_k^1, \ldots, v_k^q) \end{array} \right.$$

where $\parallel$ denotes input-to-output connection; multi-clock encompassed by having a special symbol $\perp$ (stuttering)

Two approaches

1. Similar to *elastic circuits* [Cortadella] or *latency insensitive designs* [Carloni] in circuits:

2. Time-based approach, TTA with relaxed constraints.

Two approaches

1. Similar to *elastic circuits* [Cortadella] or *latency insensitive designs* [Carloni] in circuits:

   (a) See synchronous designs as Kahn Process Networks $\Rightarrow$ blocking reads and infinite buffers

2. Time-based approach, TTA with relaxed constraints.

Two approaches

1. Similar to *elastic circuits* [Cortadella] or *latency insensitive designs* [Carloni] in circuits:

   (a) See synchronous designs as Kahn Process Networks $\Rightarrow$ blocking reads and infinite buffers

   (b) Since buffers must be finite, writes must be controlled $\Rightarrow$ block writes when buffers filled $\Rightarrow$ use *back-pressure*

2. Time-based approach, TTA with relaxed constraints.

Two approaches

1. Similar to *elastic circuits* [Cortadella] or *latency insensitive designs* [Carloni] in circuits:
   (a) See synchronous designs as Kahn Process Networks $\Rightarrow$ blocking reads and infinite buffers
   (b) Since buffers must be finite, writes must be controlled $\Rightarrow$ block writes when buffers filled $\Rightarrow$ use *back-pressure*
   (c) Replace blocking by skipping

2. Time-based approach, TTA with relaxed constraints.

Two approaches

1. Similar to *elastic circuits* [Cortadella] or *latency insensitive designs* [Carloni] in circuits:
   (a) See synchronous designs as Kahn Process Networks $\Rightarrow$ blocking reads and infinite buffers
   (b) Since buffers must be finite, writes must be controlled $\Rightarrow$ block writes when buffers filled $\Rightarrow$ use *back-pressure*
   (c) Replace blocking by skipping

   In this approach, time is logical. No assumption on local clocks. Performance studies come as a second step.
2. Time-based approach, TTA with relaxed constraints.

Two approaches

1. Similar to *elastic circuits* [Cortadella] or *latency insensitive designs* [Carloni] in circuits: In this approach, time is logical. No assumption on local clocks. Performance studies come as a second step.
2. Time-based approach, TTA with relaxed constraints:
   - relative drift between clocks must be bounded
   - communication delays must be bounded
   - control (skipping) is purely local, based on counters
   - no blocking read/write, no back-pressure

Time-based approach [Caspi]

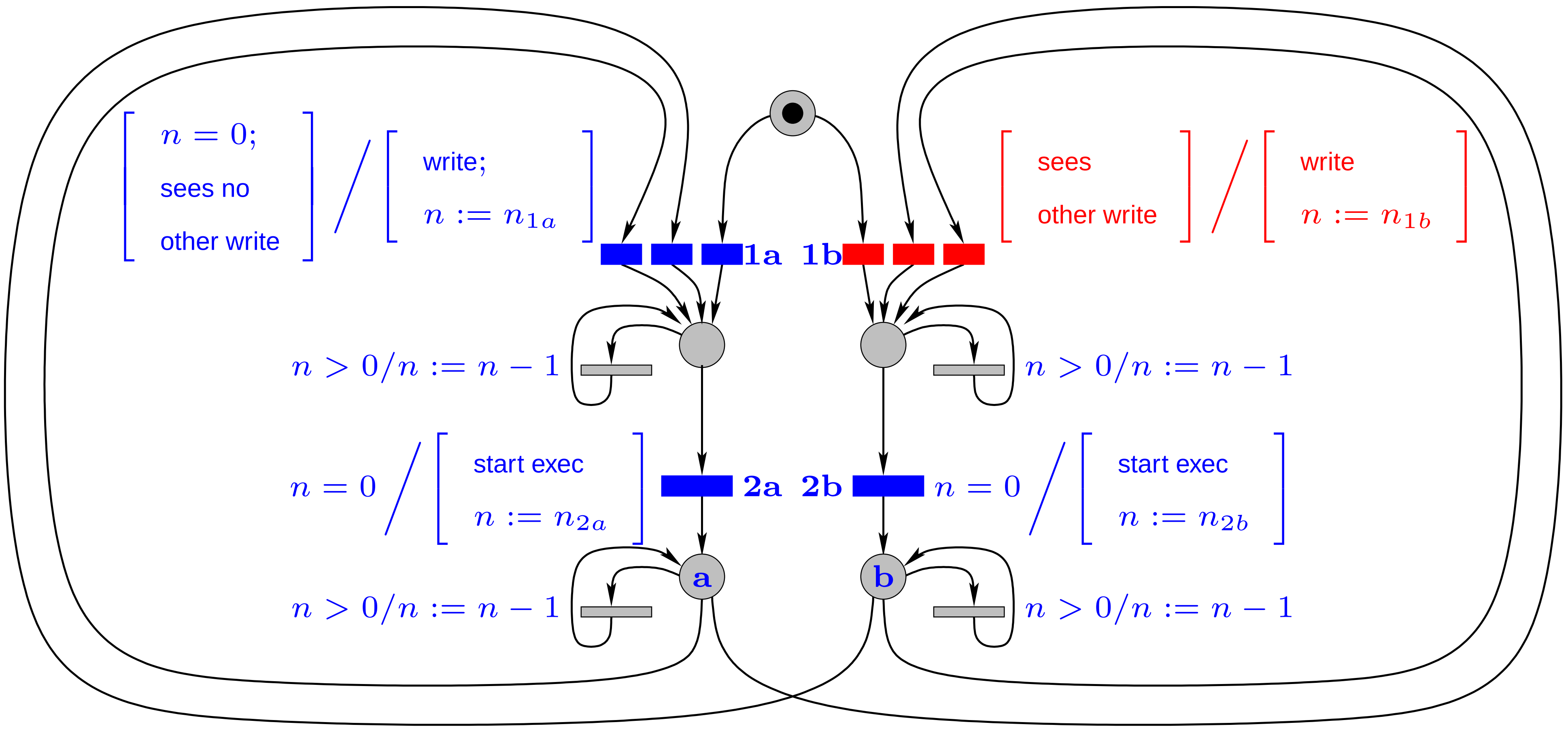
Further Assumptions:

1. Updates of every variable are visible to every node (updates may not occur at each reaction: multi-clock)
2. Communications between different sites occur through state variables and are thus subject to a unit delay.
3. For each computing unit, executions take at most one clock cycle and a computing unit which starts executing freezes its input data.
4. The inter-tick time is uniformly bounded from below and from above; communication delays are uniformly bounded.

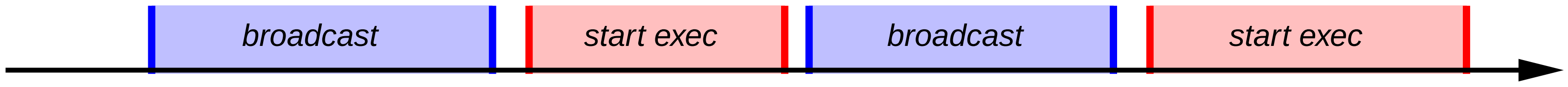
Time-based approach [Caspi]

- The following protocol is run at each node
- This protocol is entirely local and timed based, using the local (non synchronized) clock of the node.

Time-based approach [Caspi]

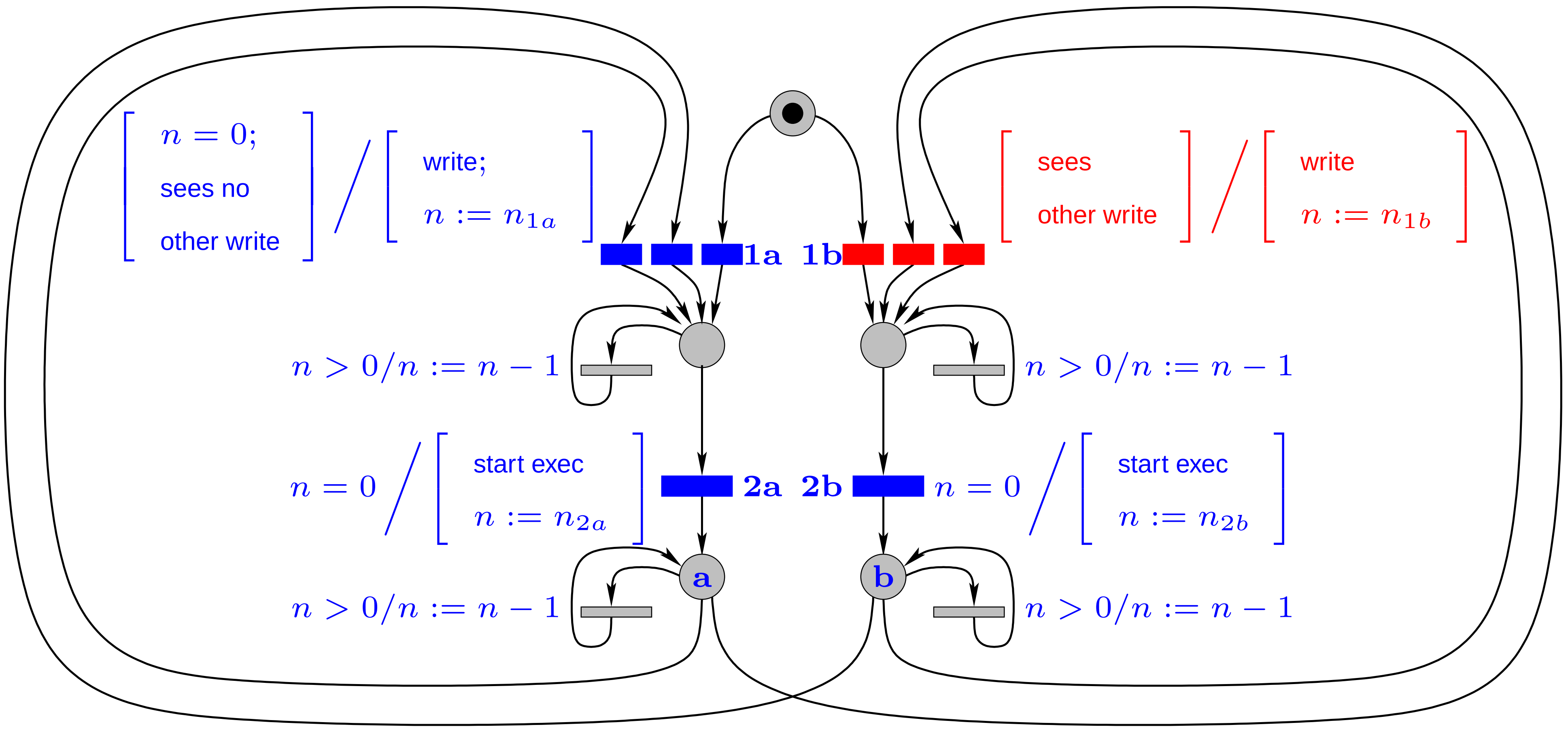


$\left[\begin{array}{l} n = 0; \\ \text{sees no} \\ \text{other write} \end{array}\right] / \left[\begin{array}{l} \text{write;} \\ n := n_{1a} \end{array}\right]$
$\left[\begin{array}{l} \text{sees} \\ \text{other write} \end{array}\right] / \left[\begin{array}{l} \text{write} \\ n := n_{1b} \end{array}\right]$
1a
1b
$n > 0 / n := n - 1$
$n > 0 / n := n - 1$
$n = 0 / \left[\begin{array}{l} \text{start exec} \\ n := n_{2a} \end{array}\right]$
2a
2b
$n = 0 / \left[\begin{array}{l} \text{start exec} \\ n := n_{2b} \end{array}\right]$
a
b
$n > 0 / n := n - 1$
$n > 0 / n := n - 1$



Red transition is synchronizing, other ones let time pass

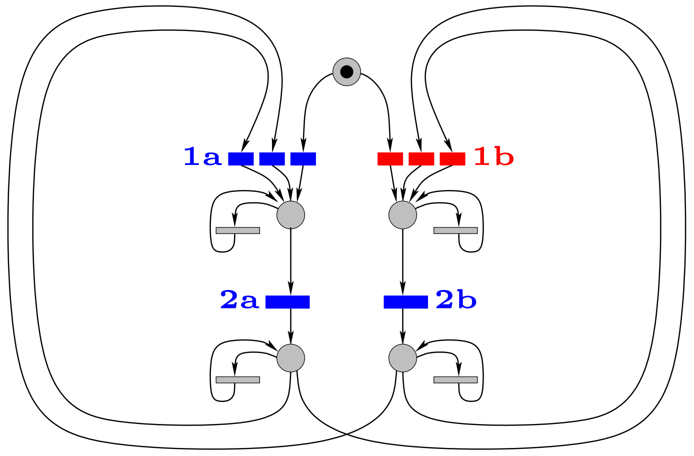
Time-based approach [Caspi]

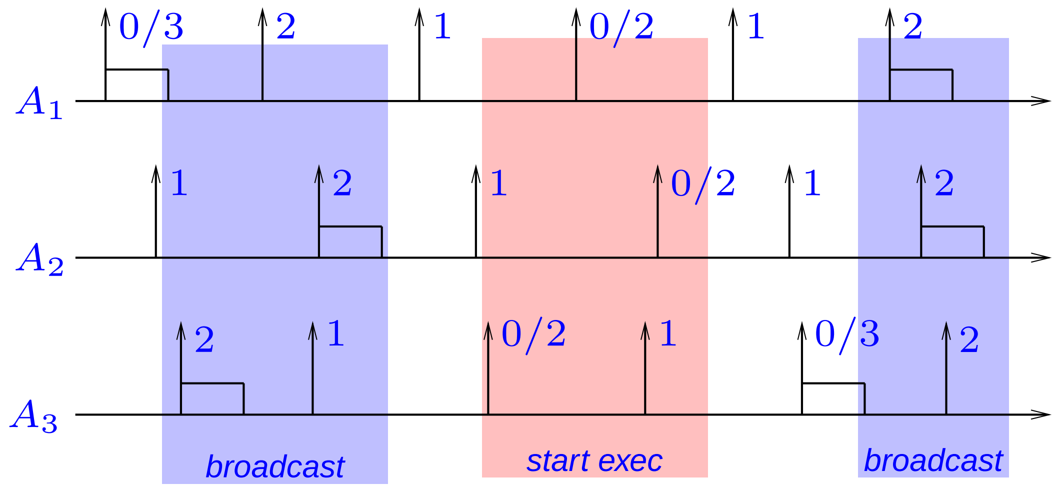


$\left[n = 0; \text{ sees no other write} \right] / \left[\text{write; } n := n_{1a} \right]$
$\left[\text{sees other write} \right] / \left[\text{write } n := n_{1b} \right]$
1a
1b
$n > 0 / n := n - 1$
$n > 0 / n := n - 1$
$n = 0 / \left[\text{start exec } n := n_{2a} \right]$
$n = 0 / \left[\text{start exec } n := n_{2b} \right]$
2a
2b
a
b
$n > 0 / n := n - 1$
$n > 0 / n := n - 1$

Theorem: for suitable choices of $n_{1a}, n_{1b}, n_{2a}, n_{2b}$, *broadcast* and *start exec* phases globally alternate: flow semantics is preserved.

Time-based approach [Caspi]





Illustrating the protocol with

$$T_{max} = 1.5, T_{min} = 1, \tau_{max} = \tau_{min} = 0.5$$

which yields

$$n_{1a} = 3, n_{1b} = n_{2a} = n_{2b} = 2 \Rightarrow \text{ slow-down } = 5$$

Time-based approach [Caspi]

Back to the assumptions:

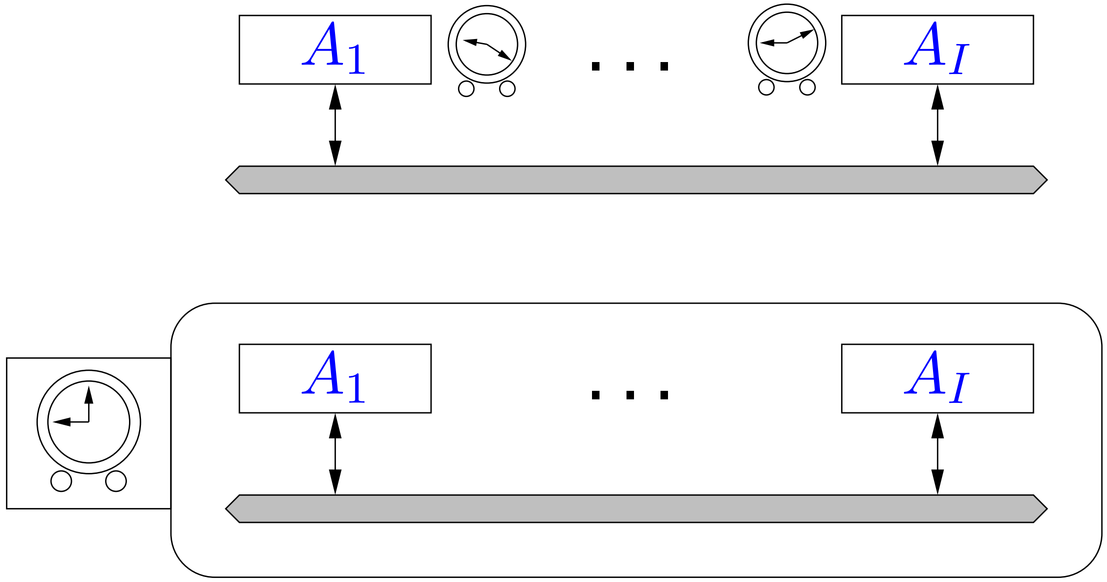
1. Updates of every variable are visible to every node (updates may not occur at each reaction: multi-clock)
2. Communications between different sites occur through state variables and are thus subject to a unit delay.
3. Executions take at most one clock cycle and a computing unit which starts executing freezes its input data.
4. The inter-tick time is bounded; communication delays are bounded.

Time-based approach [Caspi]

Back to the assumptions:

1. Updates of every variable are visible to every node (updates may not occur at each reaction: multi-clock)
2. Communications between different sites occur through state variables and are thus subject to a unit delay. Can be removed, at the price of increasing n_{2a} and n_{2b}. Causes an upsampling $\approx$ proportional to the max length of a communication chain without delays.
3. Executions take at most one clock cycle and a computing unit which starts executing freezes its input data.
4. The inter-tick time is bounded; communication delays are bounded.

Problem setting (recall)



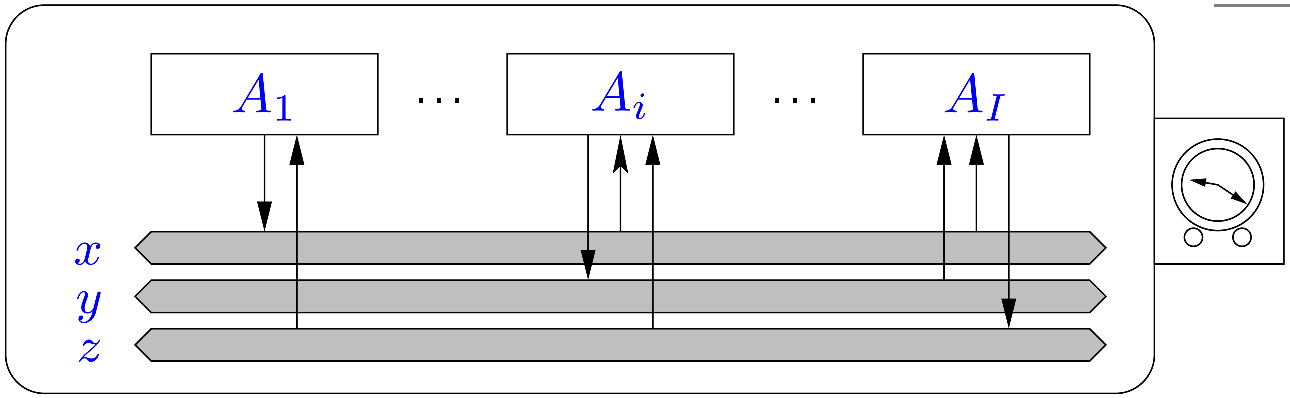
The problem: ensuring flow equivalence between LTTA design (top) and strictly synchronous design (bottom).

$$A = \underbrace{N_1 \parallel \ldots \parallel N_n}_{\text{no zero−delay circuit}}, \text{ where } N : \left\{ \begin{array}{rcl} X_k & = & f(X_{k-1}, u_k^1, \ldots, u_k^p) \\ y_k & = & g(X_{k-1}, v_k^1, \ldots, v_k^q) \end{array} \right.$$

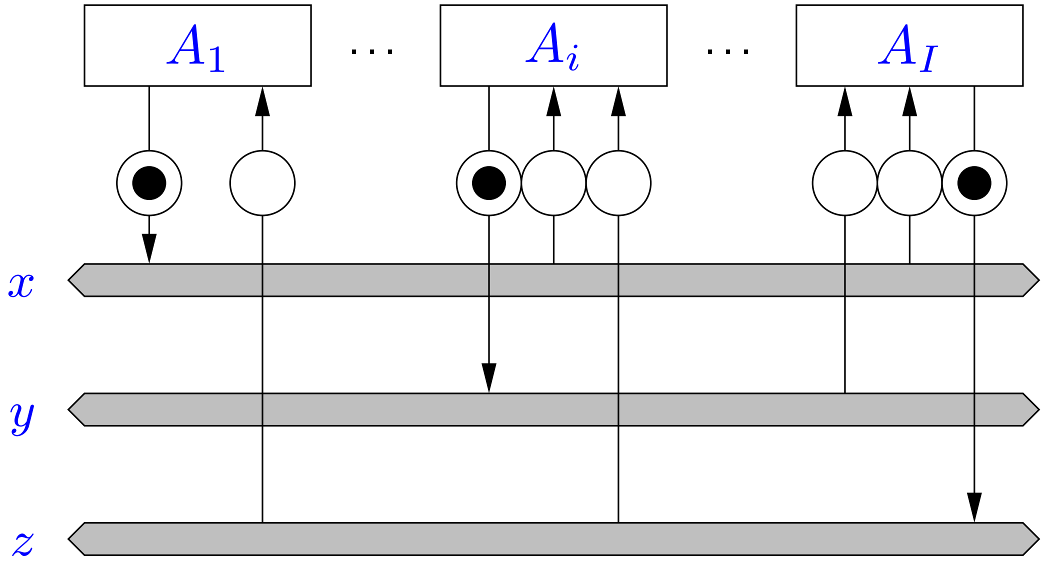
where $\parallel$ denotes input-to-output connection; multi-clock encompassed by having a special symbol $\perp$ (stuttering)

Token-based approach [Pinello, Sangiov., Tripakis]

from synchronous to Kahn networks with bounded FIFOs

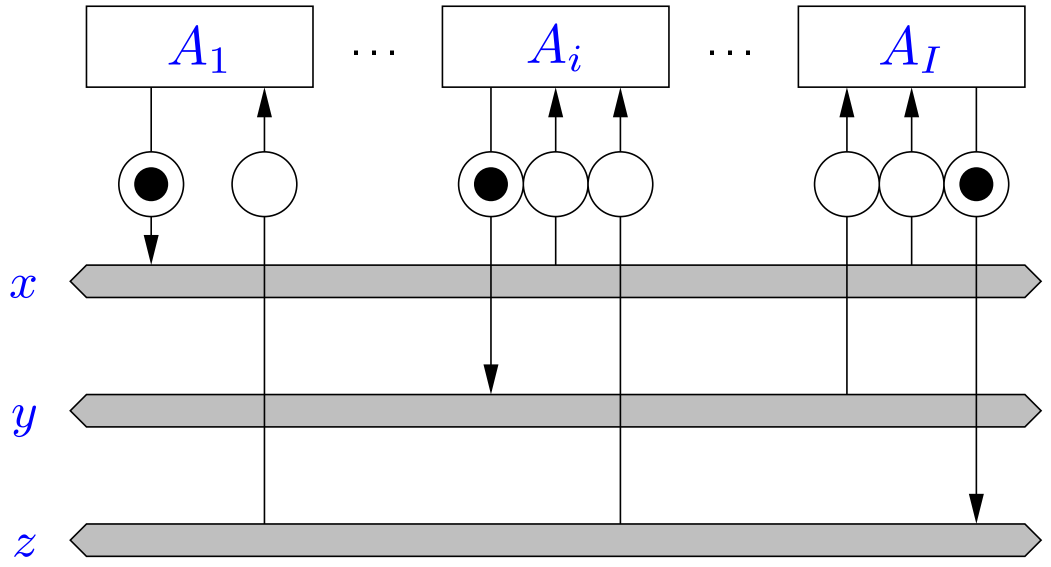


one buffer on each wire, plus one buffer per logical delay

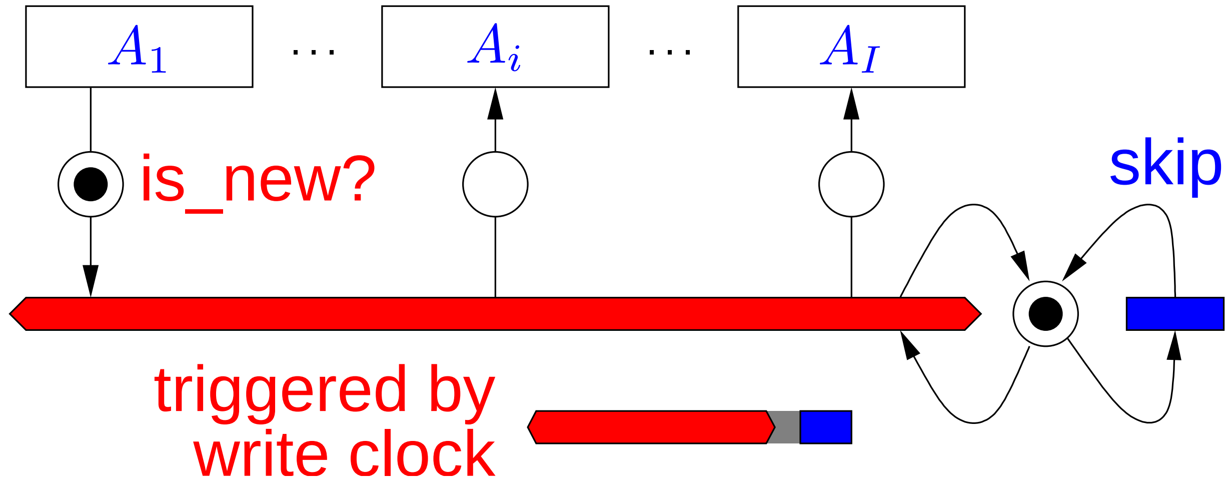


Token-based approach [Pinello, Sangiov., Tripakis]

from n-safe net with blocking reads, to . . .

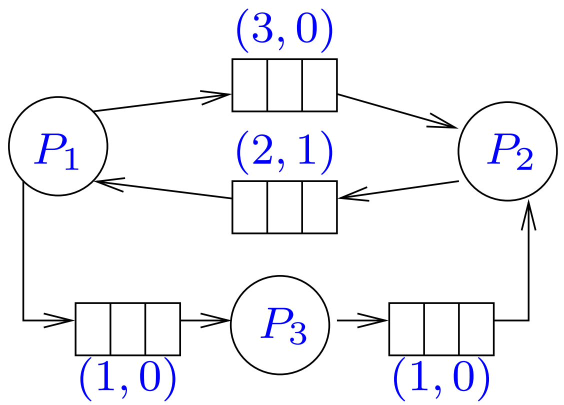


LTTA with CbS; m-FIFOs are implemented by circular tuples of shared memories



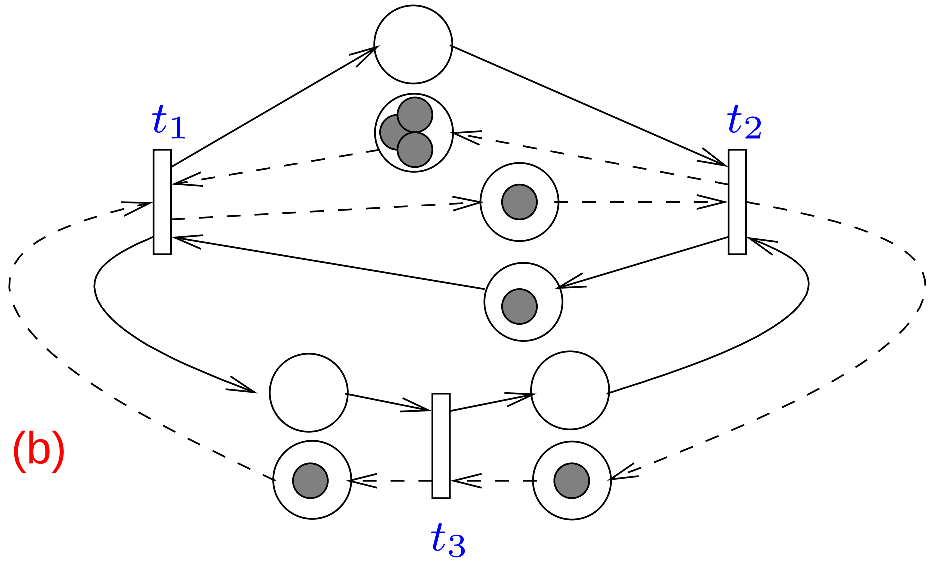
Token-based approach [Pinello, Sangiov., Tripakis]

(a)



Label $(2, 1)$ indicates that queue length is 2, with 1 initial value in the queue.

(b)

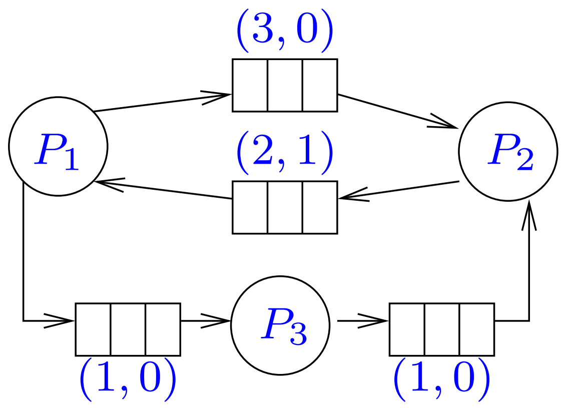


This information is directly derived from synchronous specif.

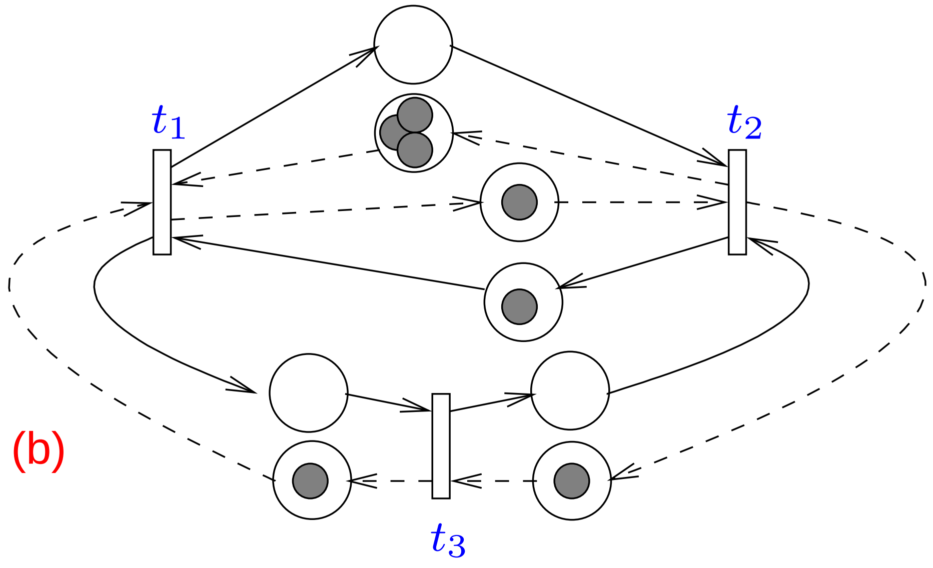
Showing systematic translation, with back-pressure

Token-based approach [Pinello, Sangiov., Tripakis]

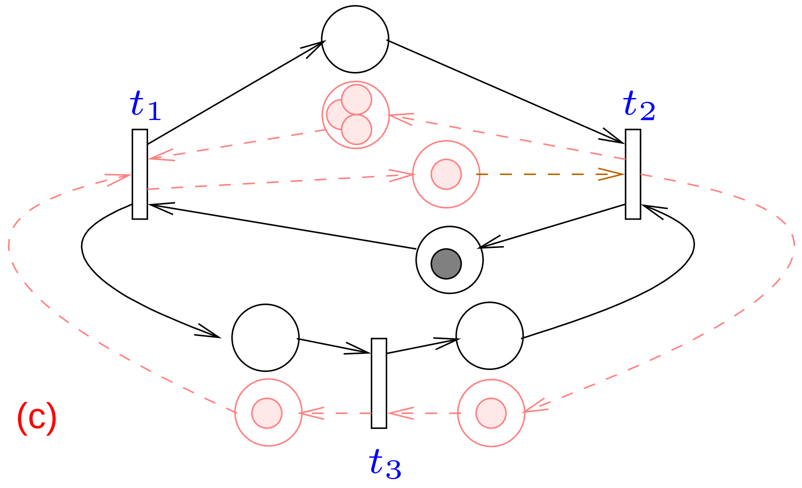
(a)



(b)



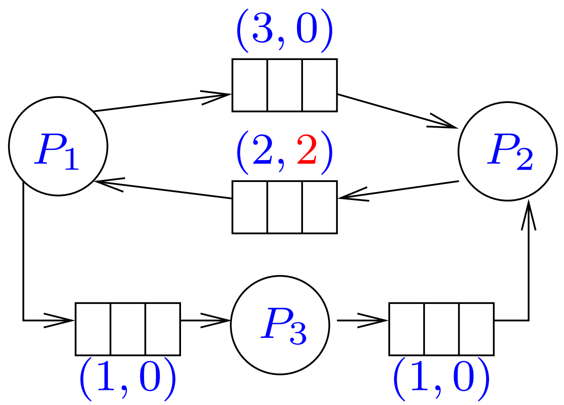
(c)



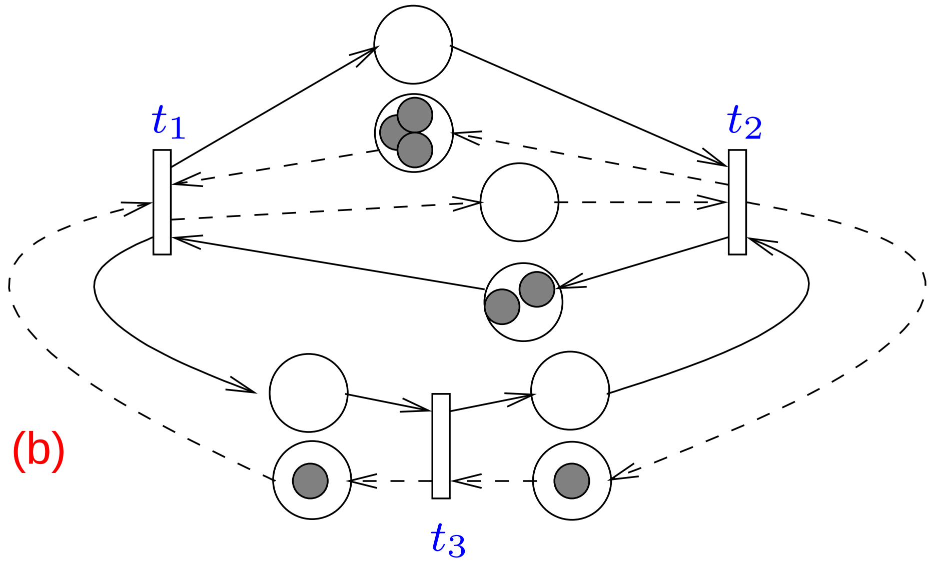
Back-pressure actually not needed here.

Token-based approach [Pinello, Sangiov., Tripakis]

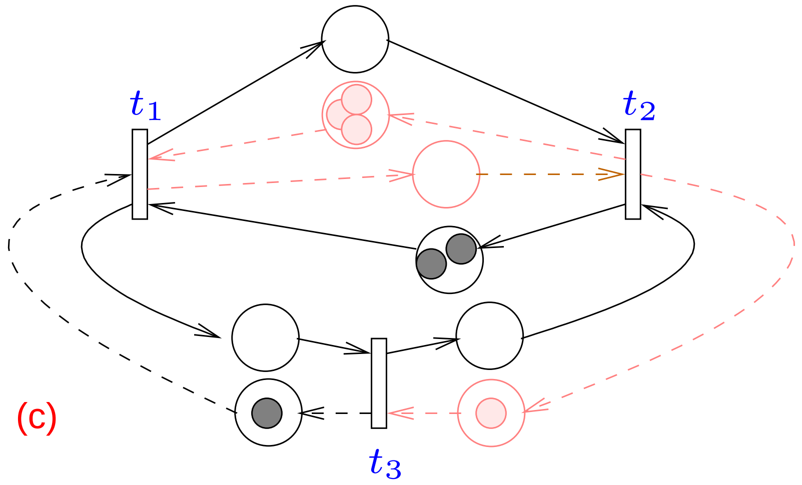
(a)



(b)



(c)



Back-pressure needed in part here.

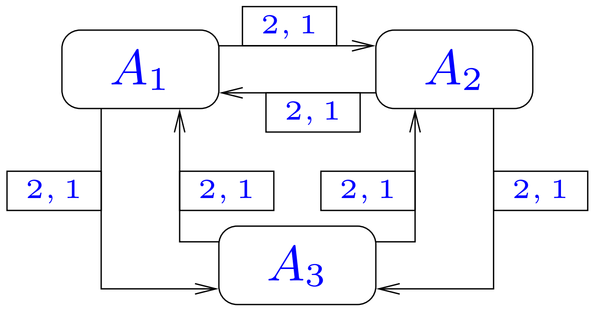
Token-based approach [Pinello, Sangiov., Tripakis]

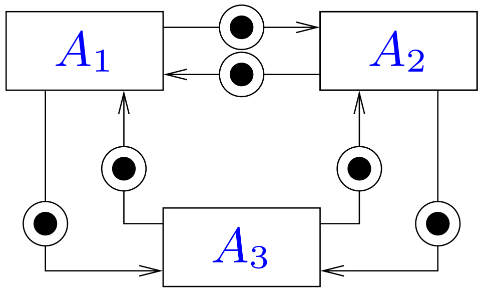
1. From synchronous specification, derive:
   - forward buffer size (1 enough if no delay on channel)
   - # of initial tokens in buffer ($=$ delay on channel)
2. Derive Marked Graph (MG) with back-pressure; optimize MG by removing unnecessary back-pressure

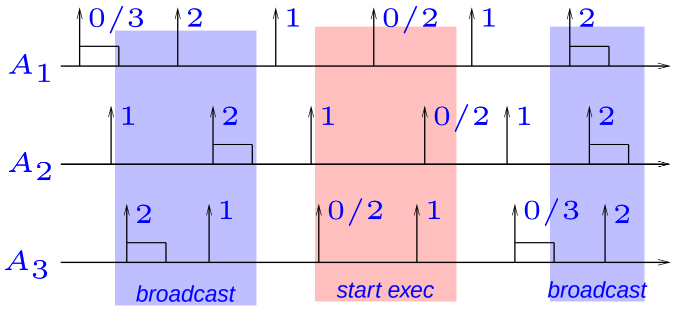
Token-based approach [Pinello, Sangiov., Tripakis]

1. From synchronous specification,
2. Derive Marked Graph (MG) with back-pressure;
3. So far this ensures preservation of flow semantics and absence of buffer overflow.
   - Logical throughput can be statically computed (classical results on MG or Max-+ calculus)
   - Buffer size can be optimized to achieve max logical throughput: integer programming
4. {Logical throughput $+$ bounds on inter-tick periods}
   $\Longrightarrow$ estimate timed throughput (# reactions by time unit)

Comparison of token- and time-based

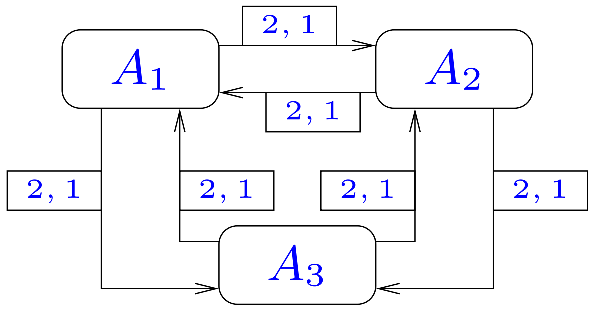


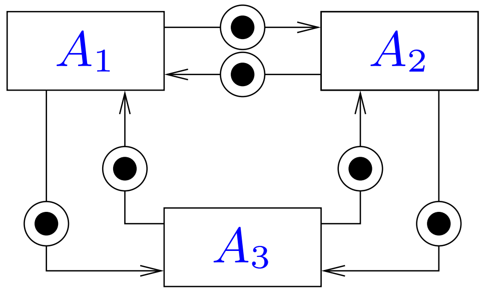


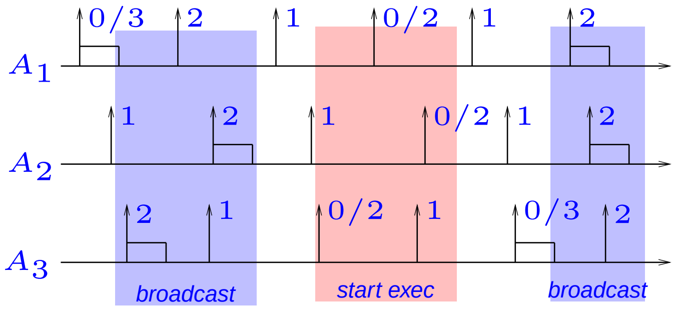


- Complete network (no back-pressure needed)
- Every channel has a delay
- $T_{max} = 1.5, T_{min} = 1, \tau_{max} = \tau_{min} = 0.5$

Comparison of token- and time-based

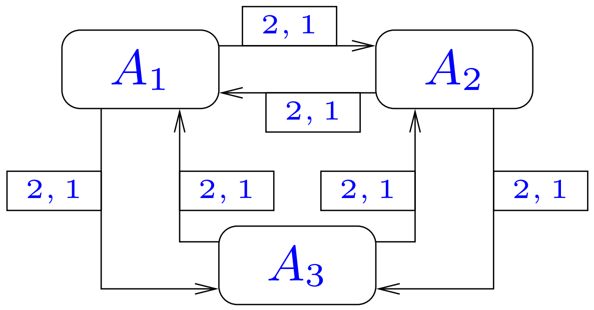


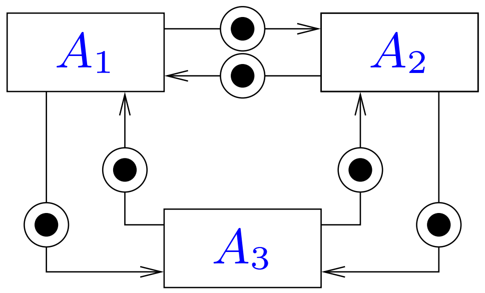


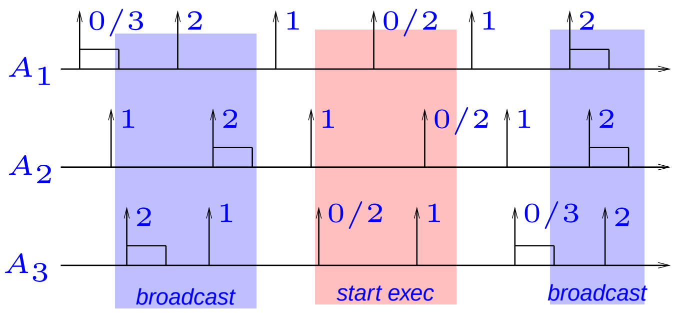


- Complete network (no back-pressure needed)
- Every channel has a delay
- $T_{max} = 1.5, T_{min} = 1, \tau_{max} = \tau_{min} = 0.5$
- token/time-based: no slow-down/slow-down of 5
  cycle duration identical in both cases

Comparison of token- and time-based

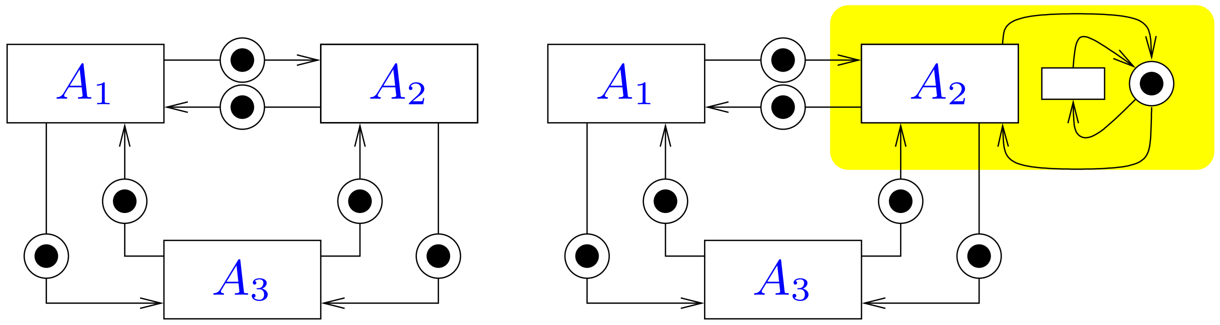






- Complete network (no back-pressure needed)
- Every channel has a delay
- $T_{max} = 1.5, T_{min} = 1, \tau_{max} = \tau_{min} = 0.5$
- token/time-based: no slow-down/slow-down of 5
  cycle duration identical in both cases
- Fault-tolerance considerations:
  token-based: breakdown of node/link freezes entire net
  time-based: breakdown of node/link does not propagate

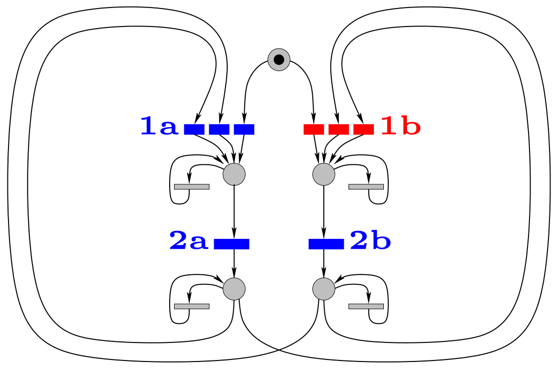
Comparison of token- and time-based

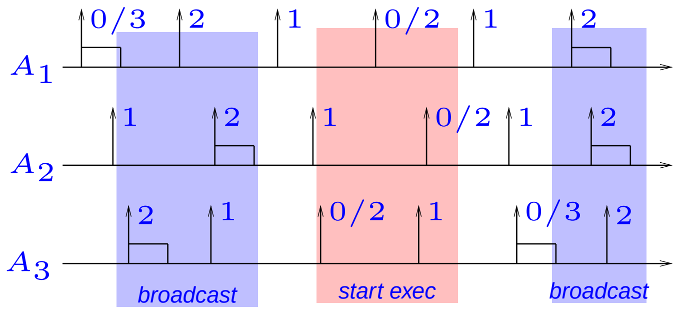


Zooming on fault-tolerance considerations:

- token-based: breakdown of node/link freezes entire net
  - if main tokens are blocked, then control can only be given to the “skip” tokens
  - as a result, no node can update its outputs
  - however, counter-measures exist by using timeouts if bounds are known on relative drifts
- time-based: breakdown of node/link does not propagate

Comparison of token- and time-based





Zooming on fault-tolerance considerations:

- token-based: breakdown of node/link freezes entire net
- time-based: breakdown of node/link does not propagate
  - if a processor or a link fails (assuming fail-stop), then a processor that must be active in a considered reaction will do so when its counter reaches zero
  - it then reads its current (non-updated) inputs and operates as usual

Conclusion

- Distributed architectures for control require careful study
  - artifacts
  - modularity
- We have investigated artifacts to discrete systems caused by LTTA
- Since no smoothness argument can work for discrete systems, we have proposed protocols to preserve specification semantics
- On the other hand, smoothness-robustness arguments should work for continuous systems
- What about hybrid systems?