Loosely Time-Triggered Architectures for Distributed Control Applications

Albert Benveniste (Inria-Rennes)
Paul Caspi † (formerly Verimag)

also contributions from
Anne Bouillard, Alberto Sangiovanni-Vincentelli
Stavros Tripakis, Claudio Pinello, Benoit Caillaud
and **Guillaume Baudart**

Collège de France — March 5, 2014

Paul Caspi

The father of this work is Paul Caspi. In the 1990's he was consulting for Airbus, Toulouse, flight control department. He noticed that Airbus was using a time-triggered but asynchronous computing and communication infrastructure for distributed control. A sophisticated discipline was used for Scade programming was used to compensate for the resulting artifacts. Paul launched PhDs to analyze and formalize this.

With a number of colleagues, we subsequently discovered that the right formalization was a new middleware that we decided to call LTTA.

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

5 Time-based LTTA

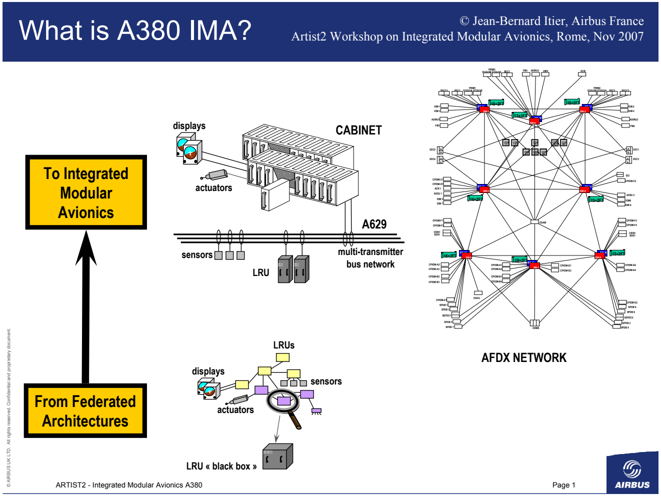
6 Performances and comparison

7 Extensions

8 Conclusion

Motivation: model based design process

From Federated to Integrated Architectures: IMA in aeronautics



What is A380 IMA?

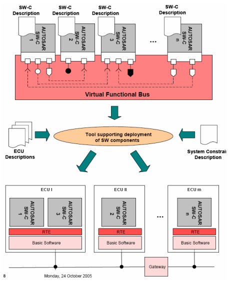
© Jean-Bernard Itier, Airbus France
Artist2 Workshop on Integrated Modular Avionics, Rome, Nov 2007

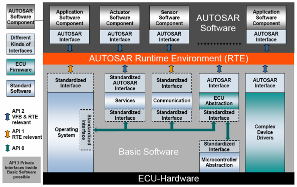
Motivation: model based design process

From Federated to Integrated Architectures: AUTOSAR in automobile

Key AUTOSAR "Methodology and RTE"

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




AUTOSAR Software Component
Different Kinds of Interfaces
ECU Firmware
Standard Software
API 2 VFB & RTE relevant
API 1 RTE relevant
API 0
API 3 Private Interfaces inside Basic Software possible
Application Software Component
Actuator Software Component
Sensor Software Component
AUTOSAR Software
Application Software Component
AUTOSAR Interface
AUTOSAR Runtime Environment (RTE)
Standardized Interface
Standardized AUTOSAR Interface
Services
Communication
ECU Abstraction
Complex Device Drivers
Operating System
Basic Software
Microcontroller Abstraction
ECU-Hardware

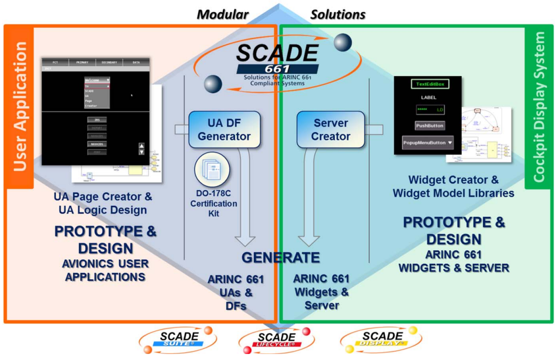
SYMTA VISION

© Dr. Kai Richter, Symtavision GmbH
The AUTOSAR Timing Model - Status and Challenges
ARTIST2 Workshop, 23.3.06 Innsbruck
8

Motivation: model based design process

Model-based design processes: IMA in aeronautics

SCADE Solutions for ARINC 661 Compliant Systems | Esterel Technologies http://www.esterel-technologies.com/products/scade-arinc-661/



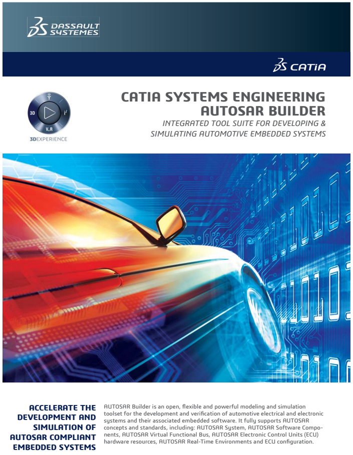
A fully-integrated COTS solution for the specification, development and certification of avionics displays following the ARINC 661 standard

The SCADE Solutions for ARINC 661 Compliant Systems are a tool suite for creating and simulating

2 sur 5 03/01/2014 10:07

Motivation: model based design process

Model-based design processes: AUTOSAR in automobile


DASSAULT SYSTEMES

CATIA

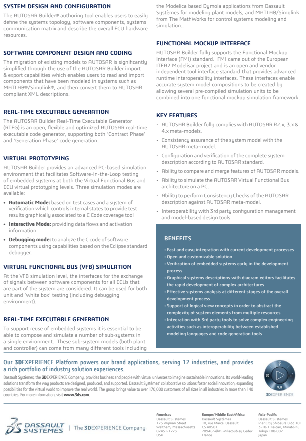
3DEXPERIENCE

CATIA SYSTEMS ENGINEERING
AUTOSAR BUILDER

INTEGRATED TOOL SUITE FOR DEVELOPING & SIMULATING AUTOMOTIVE EMBEDDED SYSTEMS

ACCELERATE THE DEVELOPMENT AND SIMULATION OF AUTOSAR COMPLIANT EMBEDDED SYSTEMS

AUTOSAR Builder is an open, flexible and powerful modeling and simulation toolset for the development and verification of automotive electrical and electronic systems and their associated embedded software. It fully supports AUTOSAR concepts and standards, including: AUTOSAR System, AUTOSAR Software Components, AUTOSAR Virtual Functional Bus, AUTOSAR Electronic Control Units (ECU) hardware resources, AUTOSAR Real-Time Environments and ECU configuration.


SYSTEM DESIGN AND CONFIGURATION

The AUTOSAR Builder® authoring tool enables users to easily define the systems topology, software components, systems communication matrix and describe the overall ECU hardware resources.

SOFTWARE COMPONENT DESIGN AND CODING

The migration of existing models to AUTOSAR is significantly simplified through the use of the AUTOSAR Builder import & export capabilities which enables users to read and import components that have been modeled in systems such as MATLAB®/Simulink®, and then convert them to AUTOSAR compliant XML descriptions.

REAL-TIME EXECUTABLE GENERATION

The AUTOSAR Builder Real-Time Executable Generator (RTEG) is an open, flexible and optimized AUTOSAR real-time executable code generator, supporting both 'Contract Phase' and 'Generation Phase' code generation.

VIRTUAL PROTOTYPING

AUTOSAR Builder provides an advanced PC-based simulation environment that facilitates Software-In-the-Loop testing of embedded systems at both the Virtual Functional Bus and ECU virtual prototyping levels. Three simulation modes are available:

- **Automatic Mode:** based on test cases and a system of verification which controls internal states to provide test results graphically associated to a C Code coverage tool
- **Interactive Mode:** providing data flows and activation information
- **Debugging mode:** to analyze the C code of software components using capabilities based on the Eclipse standard debugger.

VIRTUAL FUNCTIONAL BUS (VFB) SIMULATION

At the VFB simulation level, the interfaces for the exchange of signals between software components for all ECUs that are part of the system are considered. It can be used for both unit and 'white box' testing (including debugging environment).

REAL-TIME EXECUTABLE GENERATION

To support reuse of embedded systems it is essential to be able to compose and simulate a number of sub-systems in a single environment. These sub-system models (both plant and controller) can come from many different tools including the Modelica based Dymola applications from Dassault Systèmes for modeling plant models, and MATLAB/Simulink from The MathWorks for control systems modeling and simulation..

FUNCTIONAL MOCKUP INTERFACE

AUTOSAR Builder fully supports the Functional Mockup Interface (FMI) standard. FMI came out of the European ITEA2 Modelisar project and is an open and vendor independent tool interface standard that provides advanced runtime interoperability interfaces. These interfaces enable accurate system model compositions to be created by allowing several pre-compiled simulation units to be combined into one functional mockup simulation framework.

KEY FEATURES

- AUTOSAR Builder fully complies with AUTOSAR R2.x, 3.x & 4.x meta-models.
- Consistency assurance of the system model with the AUTOSAR meta-model.
- Configuration and verification of the complete system description according to AUTOSAR standard.
- Ability to compare and merge features of AUTOSAR models.
- Ability to simulate the AUTOSAR Virtual Functional Bus architecture on a PC.
- Ability to perform Consistency Checks of the AUTOSAR description against AUTOSAR meta-model.
- Interoperability with 3rd party configuration management and model-based design tools

BENEFITS

- Fast and easy integration with current development processes
- Open and customizable solution
- Verification of embedded systems early in the development process
- Graphical systems descriptions with diagram editors facilitates the rapid development of complex architectures
- Effective systems analysis at different stages of the overall development process
- Support of logical view concepts in order to abstract the complexity of system elements from multiple resources
- Integration with 3rd party tools to solve complex engineering activities such as interoperability between established modeling languages and code generation tools

Our 3DEXPERIENCE Platform powers our brand applications, serving 12 industries, and provides a rich portfolio of industry solution experiences.

Dassault Systèmes, the **3D**EXPERIENCE Company, provides business and people with virtual universes to imagine sustainable innovations. Its world-leading solutions transform the way products are designed, produced, and supported. Dassault Systèmes' collaborative solutions foster social innovation, expanding possibilities for the virtual world to improve the real world. The group brings value to over 170,000 customers of all sizes in all industries in more than 140 countries. For more information, visit **www.3ds.com**.

3DEXPERIENCE

DASSAULT SYSTEMES | The 3DEXPERIENCE Company

Americas	Europe/Middle East/Africa	Asia-Pacific
Dassault Systèmes	Dassault Systèmes	Dassault Systèmes
175 Wyman Street	10, rue Marcel Dassault	Pier City Shibaura Bldg 10F
Waltham, Massachusetts	CS 40501	3-18-1 Kaigan, Minato-Ku
02451-1223	78946 Vélizy-Villacoublay Cedex	Tokyo 108-002
USA	France	Japan

Motivation: model based design process

Model-based design processes:

- Models of Computation and Communication (MoCC)
    - For the functions (synchronous programming, Kahn Networks. . .) with corresponding formalisms
    - For the architectures: this talk

Motivation: TTA

Architecture MoCCs:

- TTA (Time-Triggered Architecture) [Hermann Kopetz 1987, 1991]
  A comprehensive MoCC-based architecture:
  - strong synchrony
  - global discrete notion of time
  - time-based fault-tolerance
  - time-based scheduling (TDMA)
  - time-based interfaces

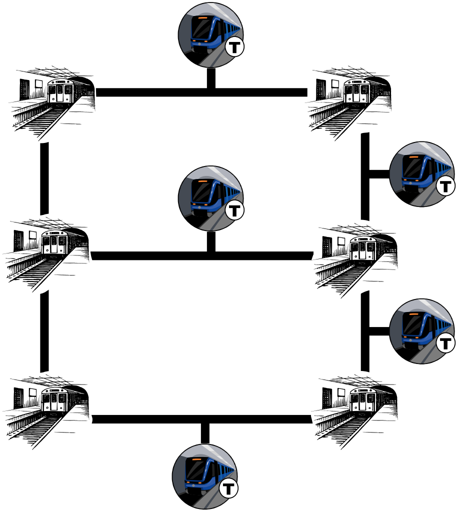
Motivation: TTA

Architecture MoCCs:

- TTA (Time-Triggered Architecture) [Hermann Kopetz 1987, 1991]
  A comprehensive MoCC-based architecture:
  - strong synchrony
  - global discrete notion of time
  - time-based fault-tolerance
  - time-based scheduling (TDMA)
  - time-based interfaces

- Resistances to TTA:
  - cost of synchronization
  - rigidity of TDMA
  - cost of re-design (adaptations & upgrades)

Motivation: resistances to TTA



Computers on trains for speed control

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

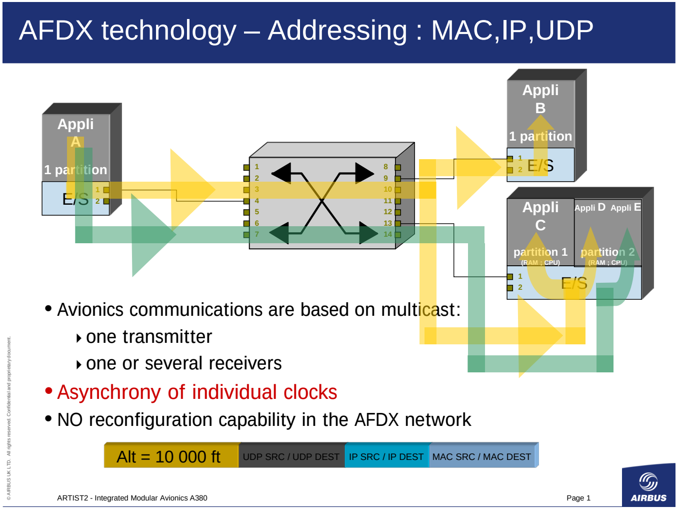
MBPC

Wired communications for fixed computers

For computers on trains: use wheels or wireless

Communication by Sampling (LTTA)

Motivation: resistances to TTA



AFDX technology – Addressing : MAC,IP,UDP

Appli A
1 partition
E/S
1
2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Appli B
1 partition
E/S
1
2
Appli C
partition 1
(RAM ; CPU)
Appli D
Appli E
partition 2
(RAM ; CPU)
E/S
1
2

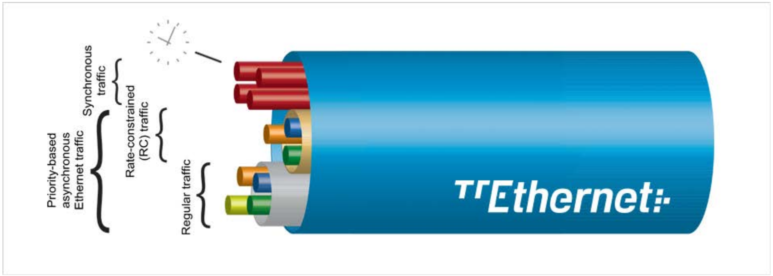
- Avionics communications are based on multicast:
  - one transmitter
  - one or several receivers
- Asynchrony of individual clocks
- NO reconfiguration capability in the AFDX network

Alt = 10 000 ft	UDP SRC / UDP DEST	IP SRC / IP DEST	MAC SRC / MAC DEST

ARTIST2 - Integrated Modular Avionics A380

Page 1

Motivation: resistances to TTA


Priority-based asynchronous Ethernet traffic
Synchronous traffic
Rate-constrained (RC) traffic
Regular traffic
TTEthernet

Quote from TTTech

TTEthernet is a fault-tolerant real-time communication protocol for safety-related systems. It integrates data flows of time-triggered, rate-constrained, and standard Ethernet in one physical infrastructure. The TTEthernet switches provide the means for robust partitioning between these three traffic classes.

Motivations and Overall Objectives

When no TTA infrastructure can be offered by the medium itself, e.g.:

- wide area distributed system
- wireless
- other. . . (available asynchronous infrastructure)

but it is still wanted to have a

- coherent *logical* synchronous basis
- with, preferably, controlled timing behavior, then:

Motivations and Overall Objectives

When no TTA infrastructure can be offered by the medium itself, e.g.:

- wide area distributed system
- wireless
- other. . . (available asynchronous infrastructure)

but it is still wanted to have a

- coherent *logical* synchronous basis
- with, preferably, controlled timing behavior, then:

Relax TTA to LTTA

(Loosely Time-Triggered Architecture)

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

5 Time-based LTTA

6 Performances and comparison

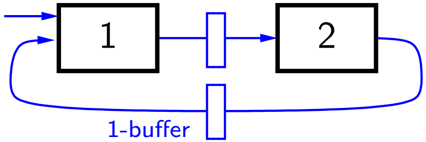
7 Extensions

8 Conclusion

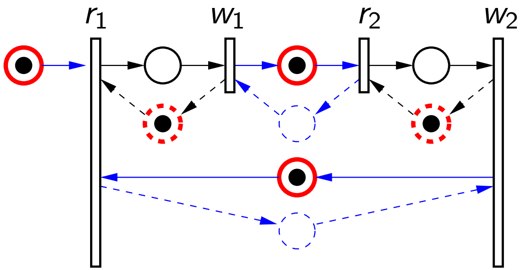
From synchronous programs to 1-safe nets

A synchronous machine with two computers and two 1-buffers

top: a data-flow representation of the synchronous machine



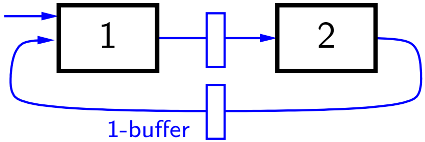
bottom: a net form;
the subset of red places represents the end of each reaction;
dashed: back-pressure



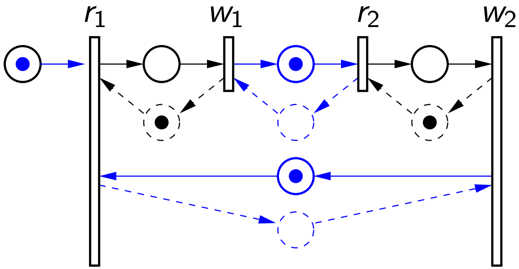
From synchronous programs to 1-safe nets

A synchronous machine with two computers and two 1-buffers

top: a data-flow representation of the synchronous machine



bottom: a net form; no special places are distinguished; yields a net model of a 1-buffer Kahn network



From synchronous programs to nets

- This shows that 1-clocked synchronous programs having no delay-free circuit can be implemented on 1-buffered nets
- For multi-clocked synchronous programs, tokens hold a $\perp$ to indicate absence of data;

From synchronous programs to nets

- This shows that 1-clocked synchronous programs having no delay-free circuit can be implemented on 1-buffered nets
- For multi-clocked synchronous programs, tokens hold a $\perp$ to indicate absence of data; is it possible to get rid of this signalling overhead?
- Yes it is! Assuming tokens carry data:
    - if, by only reading its present input tokens, a transition can infer which tokens will be absent in the next firing, then *this transition does not need the $\perp$ signaling*; such a transition is called endochronous

From synchronous programs to nets

- This shows that 1-clocked synchronous programs having no delay-free circuit can be implemented on 1-buffered nets
- For multi-clocked synchronous programs, tokens hold a $\perp$ to indicate absence of data; is it possible to get rid of this signalling overhead?
- Yes it is! Assuming tokens carry data:
  - if, by only reading its present input tokens, a transition can infer which tokens will be absent in the next firing, then *this transition does not need the $\perp$ signaling*; such a transition is called endochronous
- If a net is such that all its transitions are endochronous, then *the $\perp$-labeled tokens need not be circulated* and the resulting net provides a model of the asynchronous execution of the synchronous program

This is the subject of extensive research in the synchronous languages community (Caillaud, Potop. . .) and also related to asynchronous circuits

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

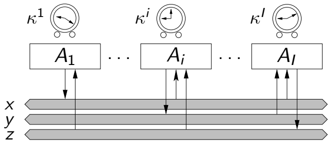
5 Time-based LTTA

6 Performances and comparison

7 Extensions

8 Conclusion

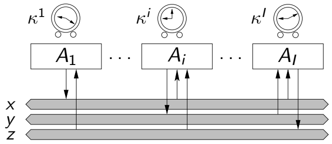
Loosely Time-Triggered Architecture



Communication by Sampling

1. Communication medium $\sim$ set of shared memories, 1 per variable
2. Each computer periodically samples its external world
   And so does the communication medium itself

Loosely Time-Triggered Architecture



Communication by Sampling

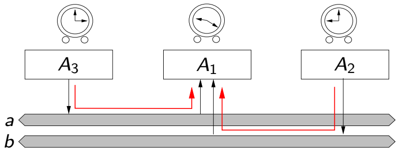
1. Communication medium $\sim$ set of shared memories, 1 per variable
2. Each computer periodically samples its external world
   And so does the communication medium itself

Advantages:

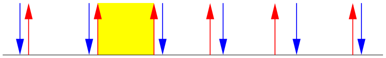
- communication medium off-the-shelf
- autonomy, no deadlock, no livelock

Results, however, in losses and duplications

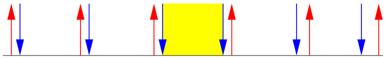
Loosely Time-Triggered Architecture



Problems when writing/sensing with non synchronized clocks:

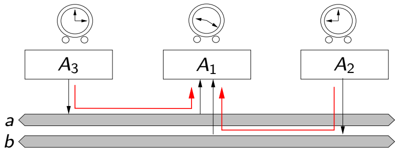


duplications

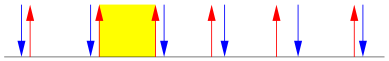


losses

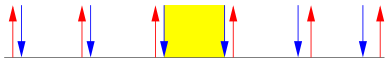
Loosely Time-Triggered Architecture



Problems when writing/sensing with non synchronized clocks:

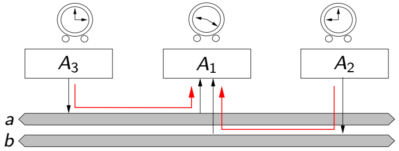


no harm so far for continuous feedback control

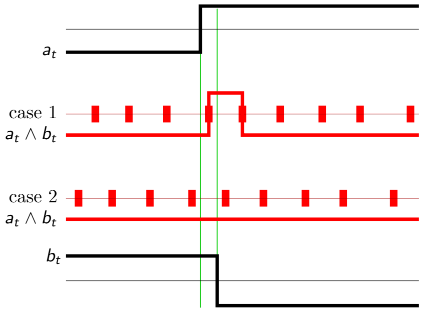


RT-Builder [Geensys→DS]
JitterBug/TrueTime [Arzen]

Loosely Time-Triggered Architecture



Problems when writing/sensing multiple discrete signals:



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

Loosely Time-Triggered Architecture

A two-level architecture:

- **Low-level high-speed computing layer**
  - for use in continuous feedback control
  - Communication by Sampling used as such; no protocol, no middleware
  - robustness to artifacts ensured thanks to
    1. continuity properties of physical system for control, and
    2. advanced techniques of robust control design

Loosely Time-Triggered Architecture

A two-level architecture:

- **Low-level high-speed computing layer**
  - for use in continuous feedback control
  - Communication by Sampling used as such; no protocol, no middleware
  - robustness to artifacts ensured thanks to
    1. continuity properties of physical system for control, and
    2. advanced techniques of robust control design
- **Top-level lower-speed computing layer**
  - for use in discrete control (protection handling, mode management)
  - **middleware ensuring strict preservation of semantics**

Loosely Time-Triggered Architecture

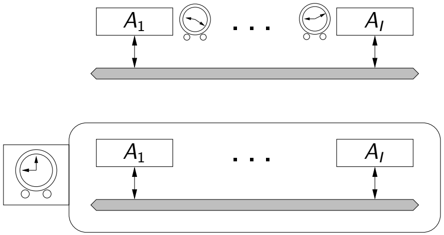
A two-level architecture:

- **Low-level high-speed computing layer**
  - for use in continuous feedback control
  - Communication by Sampling used as such; no protocol, no middleware
  - robustness to artifacts ensured thanks to
    1. continuity properties of physical system for control, and
    2. advanced techniques of robust control design
- **Top-level lower-speed computing layer**
  - for use in discrete control (protection handling, mode management)
  - **middleware ensuring strict preservation of semantics**

Case of interest: all-electric aircraft

- feedback control of electric motors with a μ-sec time scale (AFDX and ARINC technologies not fast enough)
- flight control and flight management with a m-sec time scale

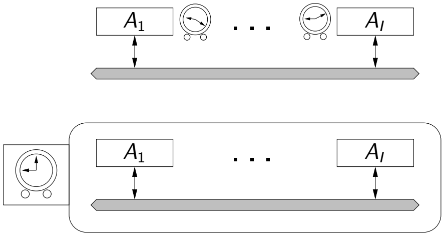
Preservation of the semantics: the problem



ensuring *flow equivalence*
between
LTTA Design (top)
and
Synchronous Design (bottom).

Flow equivalence ensured by a special LTTA protocol on top of Communication by Sampling

Preservation of the semantics: the problem



ensuring *flow equivalence*
between
LTTA Design (top)
and
Synchronous Design (bottom).

Flow equivalence ensured by a special LTTA protocol on top of Communication by Sampling

Two approaches:

- building on *back-pressure* and *elastic circuits*
- building on *time* by "making events thick"

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

5 Time-based LTTA

6 Performances and comparison

7 Extensions

8 Conclusion

Back-Pressure LTTA

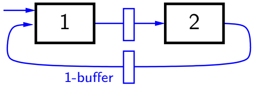
Principle:

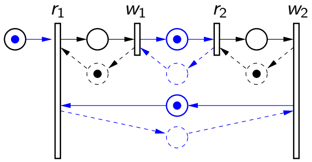
1. start with a target architecture that is a 1-safe, conflict-free Petri net (an *event graph*):
   - nodes alternate input-reads and output-writes
   - links are
     - FIFO of finite size, modeled by a series of place-transitions in sequence
     - together with a mirroring back-pressure virtual link with the same amount of successive place-transitions
   - this is called an *elastic circuit* in asynchronous hardware
   - it can implement a Kahn network with bounded buffer size

Back-Pressure LTTA

Principle:

1. start with a target architecture that is a 1-safe, conflict-free Petri net (an *event graph*):
   - nodes alternate input-reads and output-writes
   - links are
     - FIFO of finite size, modeled by a series of place-transitions in sequence
     - together with a mirroring back-pressure virtual link with the same amount of successive place-transitions
   - this is called an *elastic circuit* in asynchronous hardware
   - it can implement a Kahn network with bounded buffer size





Back-Pressure LTTA

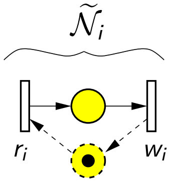
top: Node as a Net
reads and writes alternate

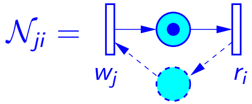
bottom: Link as a Net
1-buffer on each link

$\mathcal{N}_{ji}$: directed link $j \to i$
dashed: back-pressure

$$\left(\prod_i \widetilde{\mathcal{N}}_i\right) \times \left(\prod_{j \to i} \mathcal{N}_{ji}\right)$$

BP-EC (elastic circuit)

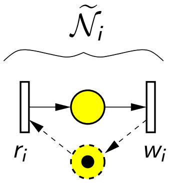


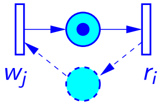


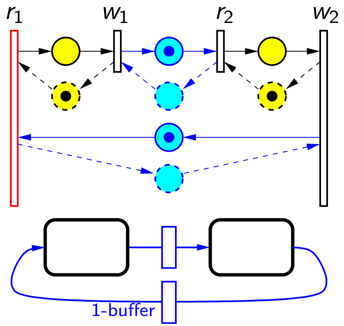
Back-Pressure LTTA

top: Node as a Net
reads and writes alternate

bottom: Link as a Net
1-buffer on each link



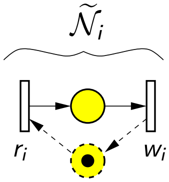
$\mathcal{N}_{ji} =$ 

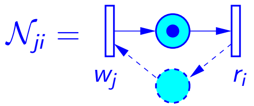


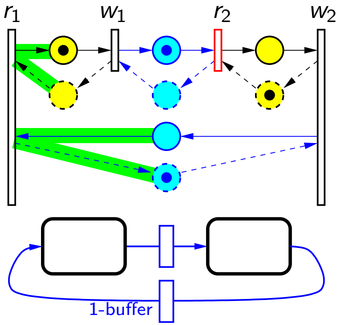
Back-Pressure LTTA

top: Node as a Net
reads and writes alternate

bottom: Link as a Net
1-buffer on each link







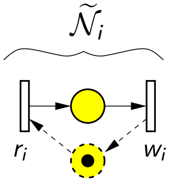
Back-Pressure LTTA

top: Node as a Net
reads and writes alternate

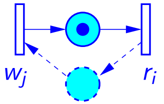
bottom: Link as a Net
1-buffer on each link

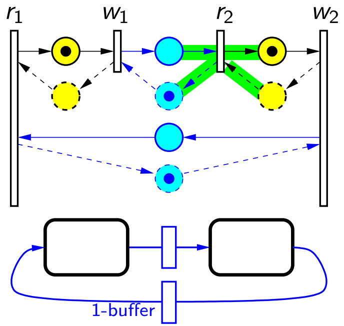
Problem:
fail-stop of a node

blocks the entire net



$\mathcal{N}_{ji} =$





Back-Pressure LTTA

Principle:

1. start with a target architecture that is a 1-safe, conflict-free Petri net (an *event graph*):
   - nodes alternate input-reads and output-writes
   - links are
     - FIFO of finite size, modeled by a series of place-transitions in sequence
     - together with a mirroring back-pressure virtual link with the same amount of successive place-transitions
   - this is called an *elastic circuit* in asynchronous hardware
   - it can implement a Kahn network with bounded buffer size

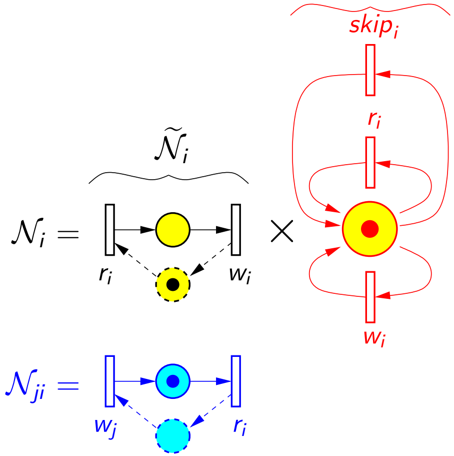
Back-Pressure LTTA

Principle:

1. start with a target architecture that is a 1-safe, conflict-free Petri net (an *event graph*):
    - nodes alternate input-reads and output-writes
    - links are
        - FIFO of finite size, modeled by a series of place-transitions in sequence
        - together with a mirroring back-pressure virtual link with the same amount of successive place-transitions
    - this is called an *elastic circuit* in asynchronous hardware
    - it can implement a Kahn network with bounded buffer size
2. Add a skipping mechanism
    - allowing nodes to fire freely, triggered by their local clocks, without getting blocked by tokens originating from other nodes.

Back-Pressure LTTA

low priority skipping mechanism at node i
triggered by the clock of node i



$\mathcal{N}_{ji}$: directed link $j \to i$
dashed: back-pressure

$$\left(\prod_i \widetilde{\mathcal{N}}_i\right) \times \left(\prod_{j \to i} \mathcal{N}_{ji}\right)$$

BP-EC (elastic circuit)

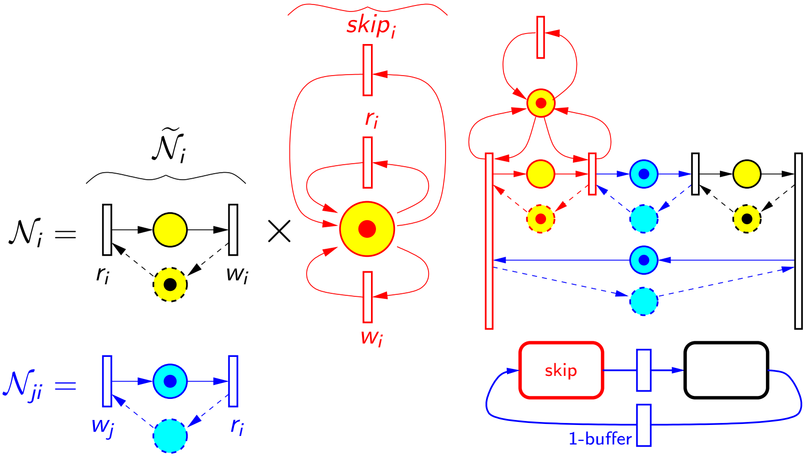
$$\left(\prod_i \mathcal{N}_i\right) \times \left(\prod_{j \to i} \mathcal{N}_{ji}\right)$$

BP-LTTA

Back-Pressure LTTA

low priority skipping mechanism at node i

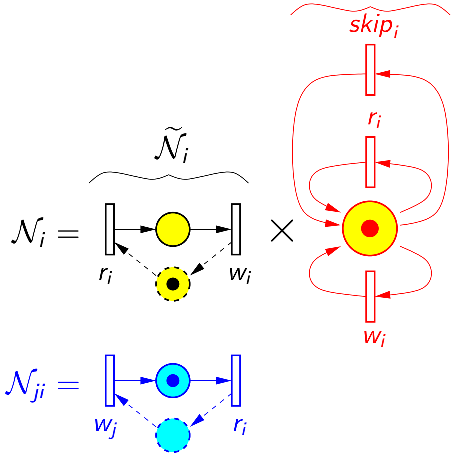
triggered by the clock of node i this node does not block

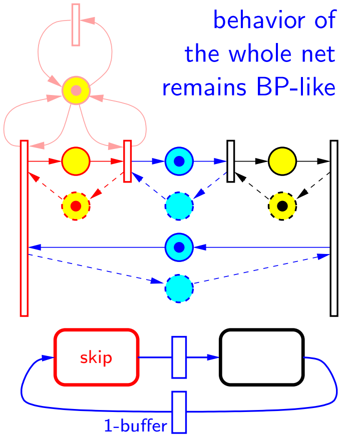


Back-Pressure LTTA

low priority skipping mechanism at node i
triggered by the clock of node i

still, the behavior of the whole net remains BP-like





Back-Pressure LTTA

Features:

- Under absence of failure, logical synchronous pace is provided with no prior assumption regarding local clocks
- Nodes are triggered by their own local clocks

  $\Longrightarrow$ node activation is robust against fail-stop nodes and local computing activities survive node failure

Back-Pressure LTTA

Features:

- Under absence of failure, logical synchronous pace is provided with no prior assumption regarding local clocks
- Nodes are triggered by their own local clocks

  $\Longrightarrow$ node activation is robust against fail-stop nodes and local computing activities survive node failure
- Still, abstracting away the (lower priority) skipping mechanism yields again the BP architecture ($\sim$ elastic circuit):

  $\Longrightarrow$ communication does not survive node failure

  $\Longrightarrow$ the pace of fetching fresh data coincides with that of pure BP

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

5 Time-based LTTA

6 Performances and comparison

7 Extensions

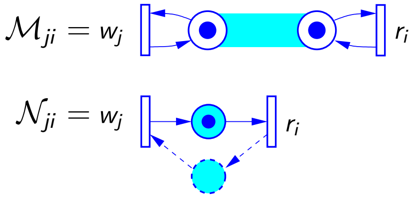
8 Conclusion

Time-based LTTA

Principle:

- Here we ensure robustness against fail-stop nodes for both
  - node activation (as in BP-LTTA)
  - and communication
- Requires prior assumptions regarding local clocks (bounds on intervals between successive ticks)

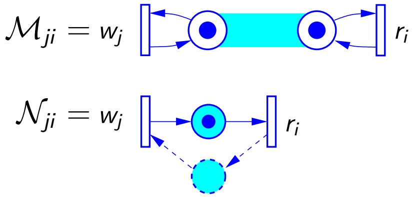
Time-based LTTA: approach



Time-based, pure CbS link (top), compared to BP-link (bottom)

Observe the lack of synchronization that results

Time-based LTTA: approach



Time-based, pure CbS link (top), compared to BP-link (bottom)

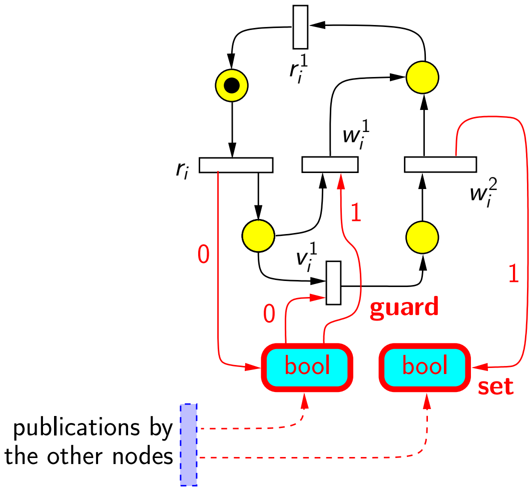
Observe the lack of synchronization that results

Re-synchronization

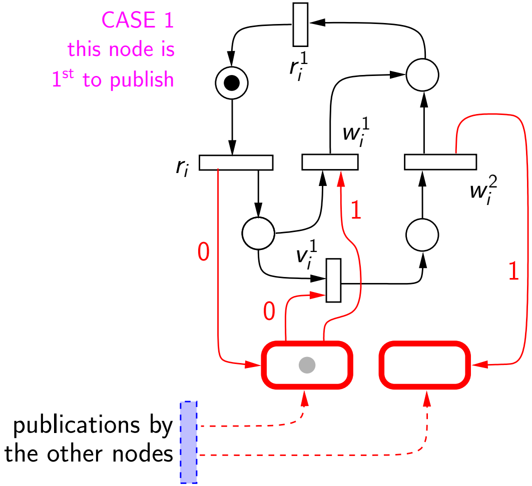
Re-synchronization is by ensuring a clean alternation of writing and reading phases throughout the entire architecture. This is achieved by:

- slowing-down by synchronizing on *local* physical time
- accelerating using a token-based *publication* broadcast

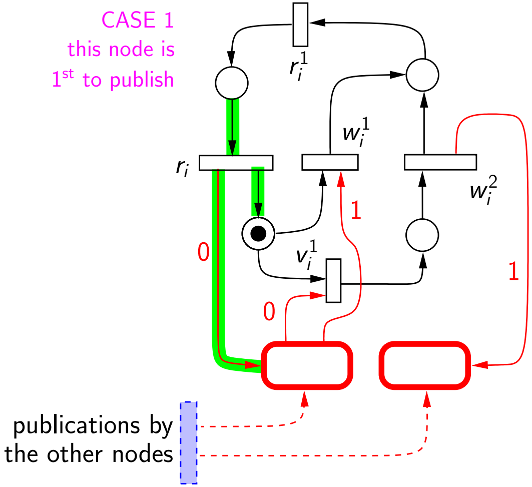
Time-based LTTA: distributed protocol



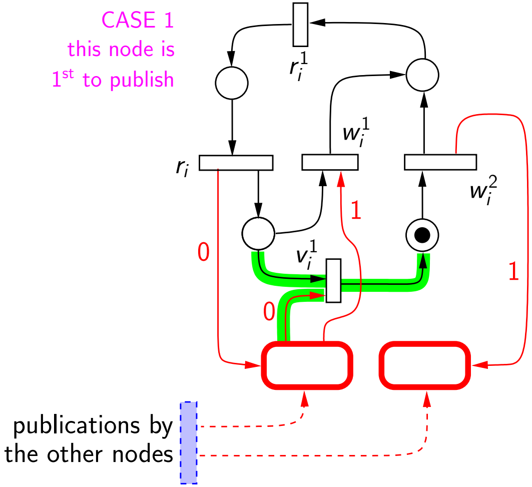
Time-based LTTA: distributed protocol



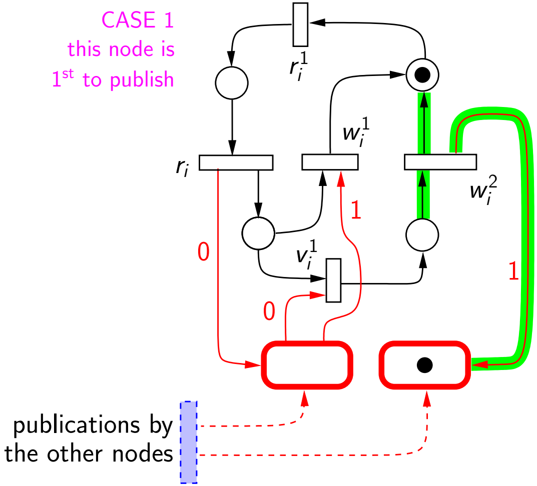
Time-based LTTA: distributed protocol



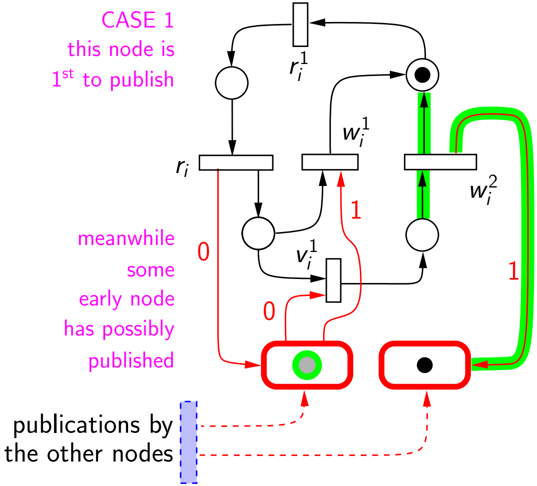
Time-based LTTA: distributed protocol



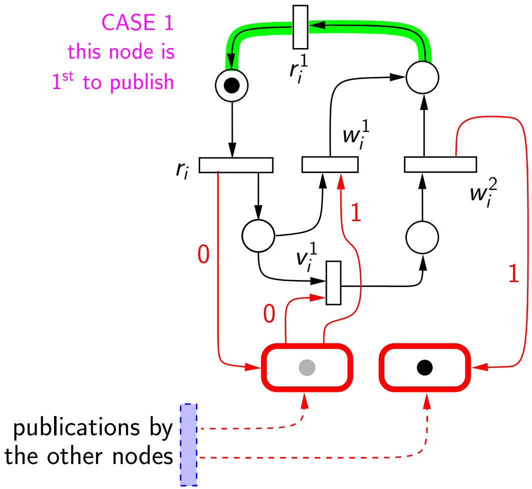
Time-based LTTA: distributed protocol



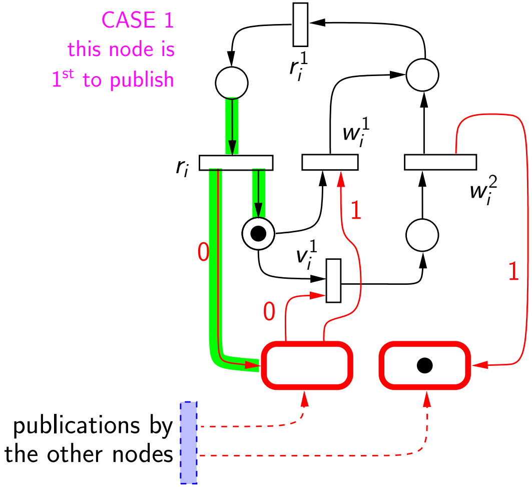
Time-based LTTA: distributed protocol



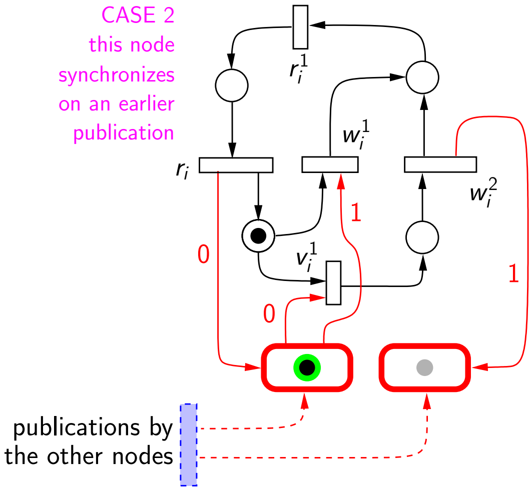
Time-based LTTA: distributed protocol



Time-based LTTA: distributed protocol

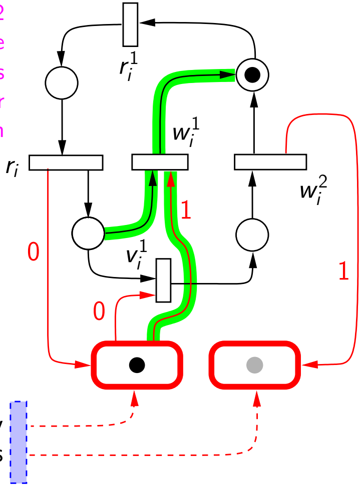


Time-based LTTA: distributed protocol



Time-based LTTA: distributed protocol

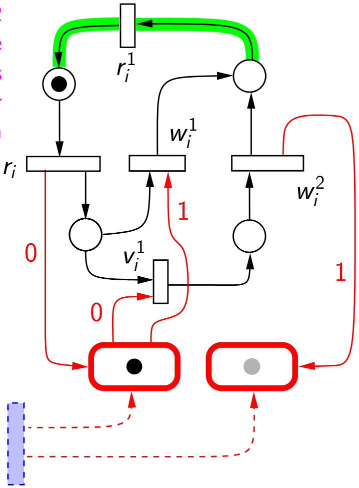
CASE 2
this node
synchronizes
on an earlier
publication



publications by
the other nodes

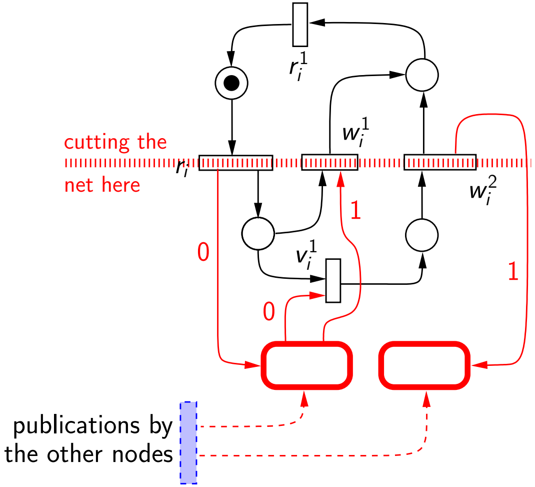
Time-based LTTA: distributed protocol

CASE 2
this node
synchronizes
on an earlier
publication

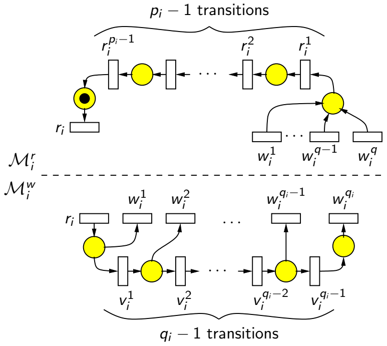


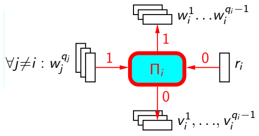
publications by
the other nodes

Time-based LTTA: distributed protocol



Time-based LTTA: distributed protocol





$w_i^1 \ldots w_i^{q_i-1}$
1
$\forall j \neq i : w_j^{q_j}$
1
Π_i
0
r_i
0
$v_i^1, \ldots, v_i^{q_i-1}$

Assumption: Inter-tick time of local clocks:
$T_{\min} \leq \kappa_{k+1}^i - \kappa_k^i \leq T_{\max}$;
Communication delays:
$\tau_{\min} \leq \tau \leq \tau_{\max}$.

Thm: with adequate choices of p and q the TB-LTTA net ensures a clean alternation of global read and write periods

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

5 Time-based LTTA

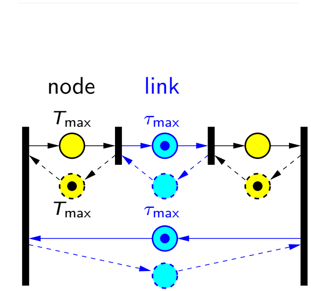
6 Performances and comparison

7 Extensions

8 Conclusion

Back-pressure LTTA performance

- Inter-tick time of local clocks: $T_{\min} \leq \kappa^i_{k+1} - \kappa^i_k \leq T_{\max}$
- Communication delays: $\tau_{\min} \leq \tau \leq \tau_{\max}$.



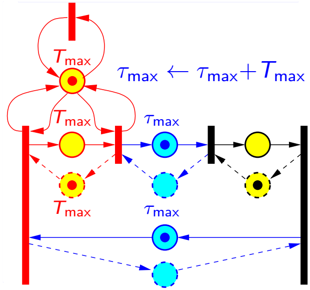
Performance of Back-Pressure LTTA (using max/+)

Elastic circuit (no periodic clock):

$$\lambda_{\widetilde{\mathcal{N}}} = \frac{1}{2\max(T_{max}, \tau_{\max})}$$

Back-pressure LTTA performance

- Inter-tick time of local clocks: $T_{\min} \leq \kappa^i_{k+1} - \kappa^i_k \leq T_{\max}$
- Communication delays: $\tau_{\min} \leq \tau \leq \tau_{\max}$.



Performance of Back-Pressure LTTA (using max/+)

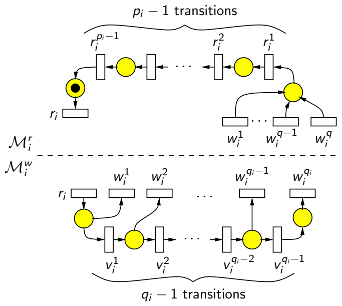
Elastic circuit (no periodic clock):

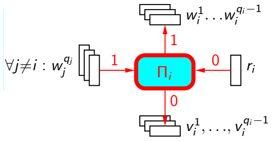
$$\lambda_{\widetilde{\mathcal{N}}} = \frac{1}{2\max(T_{max}, \tau_{\max})}$$

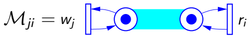
With clock and skipping mechanism:
$T_{max} \leftarrow T_{\max}$ and $\tau_{\max} \leftarrow \tau_{\max} + T_{\max}$

$$\lambda_{\mathcal{N}} = \frac{1}{2(T_{max} + \tau_{\max})}$$

Time-based LTTA performance




$w_i^1 \dots w_i^{q_i-1}$
1
$\forall j \neq i : w_j^{q_j}$
1
Π_i
0
r_i
0
$v_i^1, \dots, v_i^{q_i-1}$



Correctness of time-based LTTA

These conditions on p, q ensure $\hat{\mathcal{L}}_{\mathcal{M}} = \mathcal{L}_{\widetilde{\mathcal{N}}}$:

$$\begin{array}{lll} p & > & \frac{2\tau_{\max}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} \\ q & > & \frac{\tau_{\max}-\tau_{\min}}{T_{\min}} + \frac{T_{\max}}{T_{\min}} + p\left(\frac{T_{\max}}{T_{\min}} - 1\right) \end{array}$$

Performance of time-based LTTA

Worst case throughput ($p_\star$ and $q_\star$ optimal) :

$$\lambda_{\mathcal{M}} = \frac{1}{(p_\star + q_\star)\, T_{\max}}$$

Comparison regarding throughput

- BP-LTTA: lower bound for throughput is

$$\lambda_{\mathcal{N}} = \frac{1}{2(T_{\max} + \tau_{\max})}$$

- TB-LTTA: when delay and jitter small relative to nominal period, $p_\star = q_\star = 2$ and the lower bound for the throughput is

$$\lambda_{\mathcal{M}} \approx \frac{1}{2}\lambda_{\mathcal{N}}$$

- TB-LTTA for distant communications or when the clocks are precise, i.e., $\tau_{\max} \gg T_{\max} \approx T_{\min}$, we have $p_\star = 2\frac{\tau_{\max}}{T_{\min}}$, $q_\star = 1$ and the lower bound for the throughput becomes

$$\lambda_{\mathcal{M}} \approx \lambda_{\mathcal{N}}$$

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

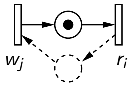
5 Time-based LTTA

6 Performances and comparison

7 Extensions

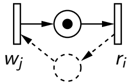
8 Conclusion

Extensions

- So far communications have a 1-delay: $\mathcal{N}_{ji} =$ 

  This can be relaxed: zero-delay communications are allowed, assuming that no zero-delay circuit exists (see paper)

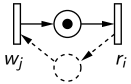
Extensions

- So far communications have a 1-delay: $\mathcal{N}_{ji} =$ 

  This can be relaxed: zero-delay communications are allowed, assuming that no zero-delay circuit exists (see paper)

- Back-pressure and time-based LTTA can be blended (see paper)

Extensions

- So far communications have a 1-delay: $\mathcal{N}_{ji} =$ 

  This can be relaxed: zero-delay communications are allowed, assuming that no zero-delay circuit exists (see paper)

- Back-pressure and time-based LTTA can be blended (see paper)

- For time-based LTTA we required broadcast of publication events
  We conjecture that this can be relaxed (but not simply removed)

1 Motivations

2 From synchronous programs to 1-safe nets

3 Loosely Timed-Triggered Architecture

4 Back Pressure LTTA

5 Time-based LTTA

6 Performances and comparison

7 Extensions

8 Conclusion

Conclusion

- Relaxing TTA to LTTA, a software based middleware
  - providing a logical synchronous time basis
  - with time bounds under additional assumptions
- Back-Pressure LTTA & Time-Based LTTA
  - similar performances w.r.t. throughput
  - BP-LTTA more flexible
  - TB-LTTA more robust against node failures
  - blending the two is easy and natural (see paper)
- The following services can be borrowed from TTA with no changes:
  - fault tolerance
  - scheduling
  - interfacing

However. . .

- Notice from Hermann Kopetz: low cost precise clocks now exist, for synchronization-free distributed control in the range of the μsec
  - does this imply that unsynchronized, precise clock based triggering is enough to ensure full TTA?
  - what about the various OS artifacts?

However. . .

- Notice from Hermann Kopetz: low cost precise clocks now exist, for synchronization-free distributed control in the range of the μsec
  - does this imply that unsynchronized, precise clock based triggering is enough to ensure full TTA?
  - what about the various OS artifacts?

- Variations about the assumptions for TB-LTTA:
  - studies of real-life constraints for distributed real-time architectures are needed to avoid considering irrelevant assumptions

However. . .

- Notice from Hermann Kopetz: low cost precise clocks now exist, for synchronization-free distributed control in the range of the μsec
  - does this imply that unsynchronized, precise clock based triggering is enough to ensure full TTA?
  - what about the various OS artifacts?
- Variations about the assumptions for TB-LTTA:
  - studies of real-life constraints for distributed real-time architectures are needed to avoid considering irrelevant assumptions
- It is very welcome that Guillaume Baudart, Timothy Bourke, and Marc Pouzet, reconsider this theory seriously and develop a simulation platform (see Synchron'13 at Dagstuhl)

Conclusion^{Conclusion}

- Formal executable models of computing infrastructures are useful for virtual modeling
- Mathematical models are more essential
  - support math reasoning
  - correct-by-construction deployment
  - with no need for extensive virtual model exploration
- MoCCs as important as MOOCs albeit less buzzy