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
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.
Motivations

From synchronous programs to 1-safe nets

Loosely Timed-Triggered Architecture

Back Pressure LT TA

Time-based LT TA

Performances and comparison

Extensions

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)
Motivation: model based design process

Model-based design processes: IMA in aeronautics

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
Motivation: model based design process

Model-based design processes: AUTOSAR in automobile

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.

The AUTOSAR Builder Real-Time Executable Generator (RTExG) 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 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 a simulation based on the AUTOSAR standard.

**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 environments).

**REAL-TIME EXECUTABLE GENERATION**

To support several embedded systems it is essential to be able to compose and simulate a number of sub-systems in a single environment. New 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 Mathworks/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 test interface standard that provides advanced communication and interoperability features. AUTOSAR Builder provides accurate system model compositions to be created by allowing several pre-compiled simulation models to be combined into one functional simulation model 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**

- Easy-to-use integration with current development processes.
- Open and customizable solutions.
- Check of model-based systems early in the development process.
- Systematic generation of AUTOSAR models.
- Consistency assurance of the system model against the AUTOSAR meta-model.
- Consistency checks against AUTOSAR configuration management.
- Interoperability with external tools.

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

Dassault Systèmes, the 3D EXPERIENCE Company, provides businesses and people with virtual power to imagine sustainable innovation. We work to bring ideas to life, to make products, services, and processes more sustainable and incremental, and to transform the way our world is to inspire the real world. The group’s revenue was over 7.7 billion of customers of all sizes in all industries in more than 140 countries. For more information, visit www.3ds.com.
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
Motivations

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)
Motives

Motivation: resistances to TTA

*Avionics communications are based on multicast:*
- one transmitter
- one or several receivers

*Asynchrony of individual clocks*

*NO reconfiguration capability in the AFDX network*

**AFDX technology – Addressing: MAC, IP, UDP**

**Alt = 10 000 ft**

**UDP SRC / UDP DEST**

**IP SRC / IP DEST**

**MAC SRC / MAC DEST**
Motivation: resistances to TTA

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

down: 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.
Loosely Timed-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
Loosely Timed-Triggered Architecture

Communication by Sampling

1. Communication medium ~ 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
Problems when writing/sensing with non synchronized clocks:

- duplications
- losses
Problems when writing/sensing with non synchronized clocks:

no harm so far for continuous feedback control

RT-Builder [Geensys→DS]
JitterBug/TrueTime [Arzen]
Problems when writing/sensing multiple discrete signals:

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

Benveniste et al. (2008) Loosely Time-Triggered Architectures for Distributed Control Applications
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 $\mu$-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

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

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

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 LT TA**

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

![Diagram](attachment:image.png)
Back-Pressure LT TA

top: Node as a Net
reads and writes alternate

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

\[ \tilde{N}_i \]

\[ N_{ji} : \text{directed link } j \rightarrow i \]
dashed: back-pressure

\[ \left( \prod_i \tilde{N}_i \right) \times \left( \prod_{j \rightarrow i} N_{ji} \right) \]

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

\[ \tilde{N}_i = r_i \uparrow w_i \]

\[ N_{ji} = \begin{array}{c}
  w_j \\
  r_i 
\end{array} \]

Benveniste et al. (2010) Loosely Time-Triggered Architectures for Distributed Control Applications
Back-Pressure LT TA

top: Node as a Net
reads and writes alternate

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

\[ \tilde{N}_i \]
\[ r_i \quad w_i \]

\[ N_{ji} = \]
\[ w_j \quad r_i \]

\[ r_1 \quad w_1 \quad r_2 \quad w_2 \]

1-buffer
Back-Pressure LTTA

Problem:
fail-stop of a node blocks the entire net

top: Node as a Net
reads and writes alternate

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

$\tilde{N}_i$

$\tilde{N}_j = \begin{array}{c}
\text{1-buffer on each link}
\end{array}$
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.

Benveniste et al. () Loosely Time-Triggered Architectures for Distributed Control Applications 20 / 38
Back-Pressure LTTA

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

$N_i = \{r_i, w_i\}$

$\tilde{N}_i = \{skip_i\}$

$N_{ji} = \{r_i, w_{ji}\}$

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

BP-EC (elastic circuit)

$$\left( \prod_i \tilde{N}_i \right) \times \left( \prod_{j \rightarrow i} N_{ji} \right)$$

BP-LTTA

$$\left( \prod_i N_i \right) \times \left( \prod_{j \rightarrow i} N_{ji} \right)$$
Back-Pressure LTATA

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

\[ \tilde{N}_i = N_i \times \text{skip}_i \]

\[ N_{ji} = \text{skip} \]

\[ 1 \text{-buffer} \]
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

\[
\hat{\mathcal{N}}_i = \mathcal{N}_i = \mathcal{N}_{ji} = \mathcal{N}_{ji} \times \mathcal{N}_{ji}
\]
Back-Pressure LT TA

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

- Nodes are triggered by their own local clocks

  $\Rightarrow$ communication does not survive node failure

- Still, abstracting away the (lower priority) skipping mechanism yields again the BP architecture ($\sim$ elastic circuit):

  $\Rightarrow$ 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 LT TA

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

\[ M_{ji} = w_j \quad r_i \]

\[ N_{ji} = w_j \quad r_i \]

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

Observe the lack of synchronization that results
Time-based LTTA: approach

\[ M_{ji} = w_j \]  
\[ N_{ji} = w_j \]

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

publications by the other nodes

set

bool

bool

guard

$w_i^1$

$w_i^2$

$v_i^1$

$w_i^1$

$r_i$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$

$w_i^1$

$w_i^2$
Time-based LTTA: distributed protocol

CASE 1
this node is 1\textsuperscript{st} to publish

publications by the other nodes
**Time-based LTTA: distributed protocol**

CASE 1 this node is 1st to publish

publications by the other nodes
Time-based LTTA: distributed protocol

CASE 1
this node is
1\textsuperscript{st} to publish
Time-based LTTA: distributed protocol

CASE 1
this node is
1st to publish

publications by
the other nodes
**Time-based LT TA: distributed protocol**

CASE 1
this node is 1\textsuperscript{st} to publish

meanwhile some early node has possibly published

publications by the other nodes
Time-based LTTE: distributed protocol

CASE 1
this node is
1\textsuperscript{st} to publish

publications by
the other nodes
Time-based LT TA: distributed protocol

CASE 1
this node is 1st to publish

publications by the other nodes
**Time-based LT TA: distributed protocol**

CASE 2
- this node synchronizes on an earlier publication

Publications by the other nodes
Time-based LTSA: distributed protocol

CASE 2
this node synchronizes on an earlier publication

publications by the other nodes
Time-based LT TA: distributed protocol

CASE 2
this node synchronizes on an earlier publication

publications by the other nodes
Time-based LTTA: distributed protocol

cutting the net here

publications by the other nodes
**Time-based LTTA: distributed protocol**

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

**Thm:** with adequate choices of \( p \) and \( q \) the TB-LTTA net ensures a clean alternation of global read and write periods.
Performances and comparison

Motivations

From synchronous programs to 1-safe nets

Loosely Timed-Triggered Architecture

Back Pressure LTTA

Time-based LTTA

Performances and comparison

Extensions

Conclusion
Performances and comparison

**Back-pressure LTTA performance**

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

---

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

**Elastic circuit (no periodic clock):**

$$\lambda_{\tilde{N}} = \frac{1}{2 \max(T_{\text{max}}, \tau_{\text{max}})}$$
Performances and comparison

Back-pressure LT TA performance

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

Performance of Back-Pressure LT TA (using \( \max/+ \))

Elastic circuit (no periodic clock):

\[ \lambda_{\tilde{N}} = \frac{1}{2 \max(T_{\text{max}},\tau_{\text{max}})} \]

With clock and skipping mechanism:

\( T_{\text{max}} \leftarrow T_{\text{max}} \) and \( \tau_{\text{max}} \leftarrow \tau_{\text{max}} + T_{\text{max}} \)

\[ \lambda_N = \frac{1}{2(T_{\text{max}} + \tau_{\text{max}})} \]
Performances and comparison

**Time-based LT TTA performance**

Correctness of time-based LT TTA

These conditions on $p, q$ ensure $\hat{L}_M = L_{\tilde{N}}$:

\[
p > 2\frac{\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}}
\]

\[
q > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p \left( \frac{T_{\text{max}}}{T_{\text{min}}} - 1 \right)
\]

Performance of time-based LT TTA

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

\[
\lambda_M = \frac{1}{(p_\star + q_\star)T_{\text{max}}}
\]
Comparison regarding throughput

- BP-LTTA: lower bound for throughput is
  \[ \lambda_N = \frac{1}{2(T_{\text{max}} + \tau_{\text{max}})} \]

- TB-LTTA: when delay and jitter small relative to nominal period, \( p_\star = q_\star = 2 \) and the lower bound for the throughput is
  \[ \lambda_M \approx \frac{1}{2} \lambda_N \]

- TB-LTTA for distant communications or when the clocks are precise, i.e., \( \tau_{\text{max}} \gg T_{\text{max}} \approx T_{\text{min}} \), we have \( p_\star = 2 \frac{\tau_{\text{max}}}{T_{\text{min}}} \), \( q_\star = 1 \) and the lower bound for the throughput becomes
  \[ \lambda_M \approx \lambda_N \]
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: \( N_{ji} = \)

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} = \begin{array}{c}
  \text{w}_j \\
  \text{r}_i 
\end{array}
$

  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: $N_{ji} = \begin{array}{c}
\text{w}_j \\
\text{N} \\
\text{r}_i
\end{array}$

  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 $\mu$sec
  - 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 $\mu$sec
  - 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 $\mu$sec
  - 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)
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