Overview
Overview

Local control
Overview

Global Application

Local control
Overview

How to preserve the semantics of the global application?
Overview

Global Application

Clock synchronization
eg. TTA [Kopetz, Bauer 2003]

Introduces control dependencies between nodes.
Overview

Global Application

Unsynchronized nodes + Middleware = LTTA

[Tripakis et al. 2008]
[Caspi, Benveniste 2008]

Less constraining protocols. Limited control dependencies.
Outline

1. What is an LT TA?
   1. Quasi-Periodic Architecture
   2. Synchronous Applications

2. General Framework

3. The two protocols
   1. Back-Pressure LT TA
   2. Time-Based LT TA

4. What About Clock Synchronisation?
Outline

1. What is an LTTA?
   1. Quasi-Periodic Architecture
   2. Synchronous Applications

2. General Framework

3. The two protocols
   1. Back-Pressure LTTA
   2. Time-Based LTTA

4. What About Clock Synchronisation?
Quasi-Periodic Architecture

- A set of “quasi-periodic” processes with local clocks and nominal period $T^n$ (jitter $\varepsilon$)

\[ 0 < T_{\min} \leq T^n \leq T_{\max} \quad \text{or} \]
\[ T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon \]

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss

- Bounded communication delay

\[ \tau_{\min} \leq \tau \leq \tau_{\max} \]
Quasi-Periodic Architecture

• A set of “quasi-periodic” processes with local clocks and nominal period $T^n$ (jitter $\varepsilon$)

$$0 < T_{\text{min}} \leq T^n \leq T_{\text{max}} \quad \text{or}$$

$$T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

• Buffered communication without message inversion or loss

• Bounded communication delay

$$\tau_{\text{min}} \leq \tau \leq \tau_{\text{max}}$$
Quasi-Periodic Architecture

- A set of “quasi-periodic” processes with local clocks and nominal period $T^n$ (jitter $\varepsilon$)

\[
0 < T_{\text{min}} \leq T^n \leq T_{\text{max}} \quad \text{or} \\
T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon
\]

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss

- Bounded communication delay

\[
\tau_{\text{min}} \leq \tau \leq \tau_{\text{max}}
\]
Problems

- **Overwriting**: Loss of values
- **Oversampling**: Duplication of values
- **Combination of signals**
Problems

- **Overwriting**: Loss of values
- **Oversampling**: Duplication of values
- **Combination of signals**
Problems

- **Overwriting**: Loss of values
- **Oversampling**: Duplication of values
- **Combination of signals**
Problems

- **Overwriting:** Loss of values
- **Oversampling:** Duplication of values
- **Combination of signals**
Problems

- **Overwriting**: Loss of values
- **Oversampling**: Duplication of values
- **Combination of signals**
Synchronous Applications
Network of Communicating Mealy Machines

• Initial state $S_{\text{init}}$, Inputs $I$, Output $O$

• Transition function $F : S \times V^I \rightarrow S \times V^O$

Semantics

Synchronous $[m]^S : (V^{n_i})^\infty \rightarrow (V^{n_o})^\infty$

Kahn $[m]^K : (V^\infty)^{n_i} \rightarrow (V^\infty)^{n_o}$
Synchronous Applications
Network of Communicating Mealy Machines

- ‘Moore-style’ Composition: machines communicate through a unit delay

- No instantaneous dependency between nodes (avoid ‘microschedule’ and ‘causality’ issues)

- A composition is also a Mealy machine

Basically, classic synchronous programs ‘mono-clock’.
What are LTTAs?

- **Base**: A quasi-periodic architecture
- **Goal**: Safely deploy a synchronous application
- **Idea**: Add a layer of middleware
Different Approaches

• **Discrete abstractions**, e.g., quasi-synchrony [Caspi 2000]
  - Allows verifications
  - State explosion, incomplete model

• **Petri nets**, [Benveniste et al. 2010]
  - Unify LTTA protocols
  - Complex model, no implementation

• **Zélus:**
  - A single language for discrete- and real-time,
  - Implementation, simulation using numerical solvers
  - But no verification at this point
1. What is an LT TA?
   1. Quasi-Periodic Architecture
   2. Synchronous Applications

2. General Framework

3. The two protocols
   1. Back-Pressure LT TA
   2. Time-Based LT TA

4. What About Clock Synchronisation?
General Framework
Modelling nodes

LT TA Controller

Mealy Machine
General Framework
Modelling nodes

LT TA Controller
Mealy Machine

Synchronous application
General Framework
Modelling nodes

Logical clock model node activation

Synchronous application
General Framework
Modelling nodes

Control the execution of the application

Logical clock model node activation

Synchronous application

LTTA Controller

Mealy Machine
General Framework
Modelling nodes

Control the execution of the application
Input sampled from memories (links)
Logical clock model node activation
Synchronous application

LT TA Controller
Mealy Machine
General Framework
Modelling nodes

Control the execution of the application

Input sampled from memories (links)

Logical clock model node activation

Communication via signals (v, b)

Synchronous application

LTTA Controller

Mealy Machine

Input sampled from memories (links)
General Framework
Modelling nodes

Control the execution of the application

Input sampled from memories (links)

Memory store the last computed value

Logical clock model node activation

Communication via signals (v, b)

Synchronous application
General Framework
Modelling Links
General Framework
Modelling Links

Input: Signals

i → [Process] → o

dc
General Framework
Modelling Links

Input: Signals

Channel: delay a signal
General Framework
Modelling Links

Input: Signals

Channel:
delay a signal

Delayed version of the sender clock
model the transmission delay
General Framework

Modelling Links

Input: Signals

Channel:
delay a signal

Memory:
store the last received value

Delayed version of the sender clock
model the transmission delay
What’s next?

Design controllers that ensure a synchronous execution of embedded machines

- **Back-Pressure LTTA**  
  [Tripakis et al. 2008]

- **Time-Based LTTA**  
  [Caspi, Benveniste 2008]
Outline

1. What is an LT TA?
   1. Quasi-Periodic Architecture
   2. Synchronous Applications

2. General Framework

3. The two protocols
   1. Back-Pressure LT TA
   2. Time-Based LT TA

4. What About Clock Synchronisation?
Back-Pressure Kahn Network

Buffer of Size 1

- Reading from a buffer is acknowledged to the writer
- Nodes alternate between `exec` and `send`
Back-Pressure Kahn Network

Buffer of Size 1

- Reading from a buffer is acknowledged to the writer
- Nodes alternate between `exec` and `send`
Back-Pressure Kahn Network
Buffer of Size 1

• Reading from a buffer is acknowledged to the writer
• Nodes alternate between exec and send
Back-Pressure LTDA

- **Difference:** nodes are triggered by their local clock
- **Idea:** adding skipping mechanism
• **Difference:** nodes are triggered by their local clock
• **Idea:** adding skipping mechanism
Back-Pressure LT TA

- **Difference:** nodes are triggered by their local clock
- **Idea:** adding skipping mechanism
Back-Pressure LTTHA

• **Difference:** nodes are triggered by their local clock
• **Idea:** adding skipping mechanism
Back-Pressure LTDA

- **Difference**: nodes are triggered by their local clock
- **Idea**: adding skipping mechanism
Back-Pressure LT TA

all_inputs_fresh / emit im = data(i) and emit a = ()

all_acks_fresh / emit o = m
Back-Pressure LTTA

- **Theorem 1:** Composition of the controller and the embedded machine is always well-defined (no cycle)

- **Theorem 2:** Back-pressure LTTA preserves the Kahn semantics of the embedded application (forget the skips)

- **Theorem 3:** The worst case throughput is: \( \frac{1}{\lambda_{BP}} = 2(T_{\text{max}} + \tau_{\text{max}}) \)
Time-Based LTTA

- **Problem:** Back-pressure multiplies the number of messages and memories, and *blocks if a node crashes*

- **Idea:** Replace back-pressure by waiting, using timing characteristics of the architecture

- **First solution:** [Caspi, Benveniste 2008]  
  Slow down the nodes to mimic a synchronous architecture, global synchronisation

- **Our proposal:** Relax broadcast assumption, localise synchronisations
Time-Based LTTP

- Nodes alternate between **exec** and **send**
- Sender sees publication of all receivers
- **Idea:** At some point, a node can be sure that:
  - the last sent data has been read
  - a fresh value is available in the memory
Time-Based LT TA

\[
\text{init } n = 1 \\
\text{last } n = 1 / \text{emit } im = \text{data}(i)
\]

\[
\text{Wait} \\
p \rightarrow \text{pre } n - 1
\]

\[
\text{Ready} \\
q \rightarrow \text{pre } n - 1
\]

\[
\text{last } n = 1 \text{ or preempted } / \text{emit } o = m
\]

TB-LTTA

\[
i \\
ro \\
m \\
o \\
im
\]
TB-LTTA

\[
\text{init } n = 1 \\
\text{last } n = 1 \quad / \quad \text{emit } im = \text{data}(i)
\]

\[
\text{Wait} \quad p \rightarrow \text{pre } n - 1
\]

\[
\text{Ready} \quad q \rightarrow \text{pre } n - 1
\]

\[
\text{last } n = 1 \quad \text{or preempted} \quad / \quad \text{emit } o = m
\]

Sender

Receiver
init n = 1
\[\text{last } n = 1 \rightarrow \text{emit } im = \text{data}(i)\]

Wait
\[p \rightarrow (\text{pre } n-1)\]

Ready
\[q \rightarrow (\text{pre } n-1)\]

last n = 1 or preempted \rightarrow \text{emit } o = m

Sender

Receiver
Sender

Receiver

init n = 1

last n = 1 / emit im = data(i)

Wait

p→(pre n−1)

Ready

q→(pre n−1)

last n = 1 or preempted / emit o = m

exec

send

3

im

TB-LTTA

Send

Wait

Ready

send

exec

Wait

Ready

send

0/3

2
Ready

Wait

send

Receiver

Sender

init n = 1
last n = 1 / emit im = data(i)

Wait
p→(pre n−1)

Ready
q→(pre n−1)

last n = 1 or preempted / emit o = m

i

ro

m

TB-LTTA

init n = 1
last n = 1 / emit im = data(i)

Wait
p→(pre n−1)

Ready
q→(pre n−1)

last n = 1 or preempted / emit o = m

i

ro

m

Sender

exec

send

3

2

send

Receiver

exec

send

0/3

2
TB-LTTA

\[
\text{init } n = 1 \\
\text{last } n = 1 \lor \text{emit im = data(i)}
\]

\[
\text{Wait p \rightarrow (pre n-1)} \\
\text{Ready q \rightarrow (pre n-1)}
\]

\[
\text{last } n = 1 \lor \text{preempted} \lor \text{emit o = m}
\]

**Sender**

**Receiver**
Ready

Wait

25

2

exec

0/2

send

3

2

1

Receiver

Sender
init n = 1

Wait

p→(pre n−1)

ready

q→(pre n−1)

last n = 1 / emit im = data(i)

last n = 1 or preempted / emit o = m

Sender

Receiver
init $n = 1$

last $n = 1$ / emit $im = data(i)$

Wait $p \rightarrow (pre \ n - 1)$

Ready $q \rightarrow (pre \ n - 1)$

last $n = 1$ or preempted / emit $o = m$

Sender

Receiver
init n = 1
last n = 1 / emit im = data(i)

Wait

p→(pre n−1)

Ready

q→(pre n−1)

last n = 1 or preempted / emit o = m

sender

Receiver

Send

exec

Wait

Ready

send

Send

exec

Wait

Ready

send

Send

exec

Wait

Ready

send

Send
init n = 1
last n = 1 / emit im = data(i)

Wait
p→(pre n−1)

Wait

Ready
q→(pre n−1)

last n = 1 or preempted / emit o = m

Sender

Receiver
Time-Based LTGA

• **Theorem 1:** Composition of the controller and the embedded machine is always well-defined (no cycle)

• **Theorem 2:** Time-based LTGA preserves the Kahn semantics of the embedded application

• **Theorem 3:** The worst case throughput is: $\frac{1}{\lambda_{TB}} = (p + q)T_{max}$
The Time-Based Protocol

**Theorem 2:** The following initial counter values ensure the preservation of the Kahn semantics

\[
p > \frac{2\tau_{\text{max}} + T_{\text{max}}}{T_{\text{min}}} \\
q > \frac{\tau_{\text{max}} - \tau_{\text{min}} + (p + 1)T_{\text{max}}}{T_{\text{min}}} - p
\]
The Time-Based Protocol

Proof sketch

• Worst case reasoning

• Tuning constants $p$ and $q$ (counter initial values)

• Ensure that the receiver always reads the proper data
The Time-Based Protocol

Property 1: \( W_{k-1}^a < E_k^b \)

\[
\begin{align*}
p & > \frac{2\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} \\
q & > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p \left( \frac{T_{\text{max}}}{T_{\text{min}}} - 1 \right)
\end{align*}
\]
The Time-Based Protocol

Property 1: $W_{k-1}^a < E_k^b$

\[ p > \frac{2\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} \]
\[ q > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p \left(\frac{T_{\text{max}}}{T_{\text{min}}} - 1\right) \]
The Time-Based Protocol

Property 1: \( W_{k-1}^a < E_k^b \)

\[
p > \frac{2\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}}
\]
\[
q > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p \left( \frac{T_{\text{max}}}{T_{\text{min}}} - 1 \right)
\]
The Time-Based Protocol

Property 1: $W_{k-1}^a < E_k^b$

\[ p > \frac{2\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} \]
\[ q > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p \left( \frac{T_{\text{max}}}{T_{\text{min}}} - 1 \right) \]
The Time-Based Protocol

Property 1: $W_{k-1}^a < E_k^b$

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

$q > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p\left(\frac{T_{\text{max}}}{T_{\text{min}}} - 1\right)$
The Time-Based Protocol

Property 1: \( W^a_{k-1} < E^b_k \)

Property 2: \( E^b_k < W^a_k \)
The Time-Based Protocol

Property 1: \( W_{k-1}^a \prec E_k^b \)

Property 2: \( E_k^b \prec W_k^a \)

\[
\begin{align*}
p &> \frac{2\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} \\
q &> \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p \left( \frac{T_{\text{max}}}{T_{\text{min}}} - 1 \right)
\end{align*}
\]
The Time-Based Protocol

Property 1: \( W_{k-1}^a \prec E_k^b \)

Property 2: \( E_k^b \prec W_k^a \)

\[
\begin{align*}
p & > \frac{2\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} \\
q & > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p\left(\frac{T_{\text{max}}}{T_{\text{min}}} - 1\right)
\end{align*}
\]
The Time-Based Protocol

Property 1: $W_{k-1}^a < E_k^b$

Property 2: $E_k^b < W_k^a$

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

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

$W_{k-1}^a \xrightarrow{pT_{\text{min}}} E_k^a \xrightarrow{qT_{\text{min}}} W_k^a$
The Time-Based Protocol

Property 1: \( W_{k-1}^a \prec E_k^b \)

Property 2: \( E_k^b \prec W_k^a \)

\[
p > \frac{2\tau_{\text{max}} + T_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p\left(\frac{T_{\text{max}}}{T_{\text{min}}} - 1\right)
\]

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

Property 1: \( W_{k-1}^a < E_k^b \)

Property 2: \( E_k^b < W_k^a \)

\[
p > \frac{2\tau_{\text{max}} + T_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p \left( \frac{T_{\text{max}}}{T_{\text{min}}} - 1 \right)
\]

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

Property 1: $W_{k-1}^a < E_k^b$

Property 2: $E_k^b < W_k^a$

\[ p > \frac{2\tau_{\text{max}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} \]
\[ q > \frac{\tau_{\text{max}} - \tau_{\text{min}}}{T_{\text{min}}} + \frac{T_{\text{max}}}{T_{\text{min}}} + p\left(\frac{T_{\text{max}}}{T_{\text{min}}} - 1\right) \]
The Time-Based Protocol

**Corollary:** The protocol ensures alternation between **exec** and **send** phases for each pair of communicating nodes.

**Broadcast communication:** ensure clean alternation throughout the entire architecture (idem for back-pressure LTTA)
Comparison

Back-pressure

Time-based
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Time-Based</th>
<th>Back-Pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flexibility</strong></td>
<td>Require architecture specifications</td>
<td>Very flexible</td>
</tr>
<tr>
<td><strong>Robustness</strong></td>
<td>Can run in a degraded mode</td>
<td>Stuck if a node crash</td>
</tr>
<tr>
<td><strong>Fault Tolerance</strong></td>
<td>Can be programmed in the application</td>
<td>Implemented in the Middleware</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td><strong>Pipelining</strong></td>
<td>Limited</td>
<td>Optimal</td>
</tr>
<tr>
<td></td>
<td>Time-Based</td>
<td>Back-Pressure</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td>Require architecture specifications</td>
<td>Very flexible</td>
</tr>
<tr>
<td><strong>Robustness</strong></td>
<td>Can run in a degraded mode</td>
<td>Stuck if a node crash</td>
</tr>
<tr>
<td><strong>Fault Tolerance</strong></td>
<td>Can be programmed in the application</td>
<td>Implemented in the Middleware</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td><strong>Pipelining</strong></td>
<td>Limited</td>
<td>Optimal</td>
</tr>
</tbody>
</table>
Outline

1. What is an LT TA?
   1. Quasi-Periodic Architecture
   2. Synchronous Applications

2. General Framework

3. The two protocols
   1. Back-Pressure LT TA
   2. Time-Based LT TA

4. What About Clock Synchronisation?
Global Clock Protocol
Central Master Synchronisation

• **Goal:** Implement clock synchronisation on a quasi-periodic architecture

• We use the most efficient protocol for comparison purposes

• One node is used as a time reference for all other nodes: the **central master clock**
Global Clock Protocol

The Big Picture

- **Local time**
- **Reference time**
- **Maximum divergence**
- **Offset after resynchronisation**
- **Reference time (Master)**
- **Synchronisation interval**
## Global Clock Protocol

### Central Master Synchronisation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Formula</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronisation interval</td>
<td>$R$</td>
<td></td>
</tr>
<tr>
<td>Offset after resynchronisation</td>
<td>$\Phi = \tau_{\text{max}} + T_{\text{max}} - \tau_{\text{min}}$</td>
<td></td>
</tr>
<tr>
<td>Drift rate</td>
<td>$\rho = \frac{T_{\text{max}}}{T_n} - 1 = \frac{T_{\text{max}} - T_{\text{min}}}{T_{\text{max}} + T_{\text{min}}}$</td>
<td></td>
</tr>
<tr>
<td>Maximum divergence</td>
<td>$\Gamma = 2\rho R$</td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>$\Pi = \Phi + \Gamma$</td>
<td></td>
</tr>
</tbody>
</table>
Global Clock Protocol

Lock-Step Execution

• Given the precision of the synchronisation one can build a global notion of time or macroticks.

• Activating nodes on each macrotick imposes a synchronous execution.

• But we need to wait for the transmission delay between execution steps.
Comparative Evaluation

Compared to synchronous execution*

* The smaller, the better.
Conclusion

• Our new model simplifies and clarifies those of previous papers

• A new proposition for the time-based protocol that does not require broadcast communication and does allow pipelining

• Model and Simulation of the protocols in Zélus Discrete model + link with continuous time

• Comparison with clock synchronisation deployed on the same architecture