Dynamic FTSS in Asynchronous Systems: the Case of Unison

S. Dubois\textsuperscript{1}  M. Gradinariu Potop-Butucaru\textsuperscript{2}  S. Tixeuil\textsuperscript{3}

ANR project ALADDIN – 12/06/2009

\textsuperscript{1}Université Pierre & Marie Curie, LIP6-CNRS & INRIA Regal
\textsuperscript{2}Université Pierre & Marie Curie, LIP6-CNRS & INRIA Regal
\textsuperscript{3}Université Pierre & Marie Curie, LIP6-CNRS & INRIA Grand Large
1. Fault-Tolerance
   - Self-Stabilization
   - Fault-Containment
   - FTSS

2. Unison
   - Strong vs. Weak Clock Synchronization
   - Related works
   - Unison – Specification
   - Remarks
   - Sub-classes of unison

3. Model
   - Shared Memory
   - Scheduling/Daemon

4. Impossibilities results
   - Overview
   - With Two Crashes (or more)
   - With Unfair Daemon

5. On the remaining cases...
   - Which cases?
   - Principle
   - Example

6. Conclusion
   - Summary
   - Perspectives
1. **Fault-Tolerance**
   - Self-Stabilization
   - Fault-Containment
   - FTSS

2. **Unison**

3. **Model**

4. **Impossibilities results**

5. **On the remaining cases...**

6. **Conclusion**
Self-Stabilization

- Characterisation of states
Self-Stabilization

- Convergence

Diagram:

- States of the system
- Legitimates states
Self-Stabilization

- Closure

States of the system

Legitimates states
Fault-Containment
Fault-Containment
Fault-Containment
Definition

\( \mathcal{A} \) is a \((f, r)\)-ftss algorithm \iff \begin{cases} \mathcal{A} \text{ is self-stabilizing.} \\ \text{and} \\ \mathcal{A} \text{ is } (f, r) - \text{fault-tolerant.} \end{cases} \)
1. Fault-Tolerance

2. Unison
   - Strong vs. Weak Clock Synchronization
   - Related works
   - Unison – Specification
   - Remarks
   - Sub-classes of unison

3. Model

4. Impossibilities results

5. On the remaining cases...

6. Conclusion
Strong Clock Synchronization

Step 0
Strong Clock Synchronization

Step 1

![Diagram](attachment:image.png)
Strong Clock Synchronization

Step 2

\[ \begin{array}{cccccccc}
  & & & X & X & X & X & X \\
\end{array} \]

\[ \begin{array}{cccccccc}
  N & & & & & & & \\
\end{array} \]
Strong Clock Synchronization

Step 3

X X X X X X X X

N

Clock Value

0 1 2 3 4 5
Weak Clock Synchronization

Step 0
Weak Clock Synchronization

Step 1
Weak Clock Synchronization

Step 2
Weak Clock Synchronization

Step 3

Clock Value

X X X X
X X X

N
Weak Clock Synchronization

Step 4

Clock Value

0
1
2
3
4
5

N
FTSS Strong Clock Synchronization

Proposition\(^1\)

There exists no FTSS algorithm for strong clock synchronization in asynchronous system.

\(^1\)from J. Beauquier, S. Kekkonen-Moneta. *Fault-tolerance and self-stabilization: impossibility results and solutions using self-stabilizing failures detectors*, in Int. J. Systems Science
Weak Clock Synchronization

- There exists self-stabilizing solutions to weak clock synchronization in \textit{asynchronous} systems.\textsuperscript{1}
- There exists FTSS solutions to weak clock synchronization in \textit{synchronous} systems.\textsuperscript{2}
- There exists self-stabilizing solutions to weak clock synchronization in \textit{synchronous} systems which copes with byzantine failures.\textsuperscript{3}

\textsuperscript{1}e.g. M. Gouda, T. Herman. \textit{Stabilizing unison}, in Inf. Process. Letter
\textsuperscript{2}e.g. S. Dolev. \textit{Possible and impossible self-stabilizing digital clock synchronization in general graphs}, in Real-Time Systems
\textsuperscript{3}e.g. M. Ben-Or, D. Dolev, E. Hoch. \textit{Fast self-stabilizing byzantine tolerant clock synchronization} in PODC’08
Weak Clock Synchronization

- There exists self-stabilizing solutions to weak clock synchronization in **asynchronous** systems.\(^1\)
- There exists FTSS solutions to weak clock synchronization in **synchronous** systems.\(^2\)
- There exists self-stabilizing solutions to weak clock synchronization in **synchronous** systems which copes with byzantine failures.\(^3\)

---

\(^1\) e.g. M. Gouda, T. Herman. *Stabilizing unison*, in Inf. Process. Letter

\(^2\) e.g. S. Dolev. *Possible and impossible self-stabilizing digital clock synchronization in general graphs*, in Real-Time Systems

\(^3\) e.g. M. Ben-Or, D. Dolev, E. Hoch. *Fast self-stabilizing byzantine tolerant clock synchronization in PODC’08*
Weak Clock Synchronization

- There exists self-stabilizing solutions to weak clock synchronization in \textit{asynchronous} systems.\(^1\)
- There exists FTSS solutions to weak clock synchronization in \textit{synchronous} systems.\(^2\)
- There exists self-stabilizing solutions to weak clock synchronization in \textit{synchronous} systems which copes with byzantine failures.\(^3\)

\(^1\)\textit{e.g.} M. Gouda, T. Herman. \textit{Stabilizing unison}, in Inf. Process. Letter
\(^2\)\textit{e.g.} S. Dolev. \textit{Possible and impossible self-stabilizing digital clock synchronization in general graphs}, in Real-Time Systems
\(^3\)\textit{e.g.} M. Ben-Or, D. Dolev, E. Hoch. \textit{Fast self-stabilizing byzantine tolerant clock synchronization} in PODC’08
Unison – Specification

Intuitively, unison = self-stabilizing weak synchronization.

**Specification of Asynchronous Unison (AU)**

Let be $\gamma_0 \in \Gamma$. An execution $\epsilon = \gamma_0\gamma_1 \ldots$ starting from $\gamma_0$ is a legitimate execution for AU if and only if:

- **Safety**: $\forall i \in \mathbb{N}, \gamma_i$ is weakly synchronised.
- **Liveness**: Each correct processor increments its clock infinitely often in $\epsilon$. 
Unison – Specification

Intuitively, unison = self-stabilizing weak synchronization.

**Specification of Asynchronous Unison (AU)**

Let be $\gamma_0 \in \Gamma$. An execution $\epsilon = \gamma_0\gamma_1 \ldots$ starting from $\gamma_0$ is a legitimate execution for AU if and only if:

- **Safety:** $\forall i \in \mathbb{N}, \gamma_i$ is weakly synchronised.
- **Liveness:** Each correct processor increments its clock infinitely often in $\epsilon$. 
Remarks

- Here, we consider *unbounded* clocks.

- The specification doesn’t forbid decrementation.
Remarks

- Here, we consider unbounded clocks.

- The specification doesn’t forbid decrementation.
Two main properties of unison:

- **Minimality**: nodes maintain no extra variables but the clock.
- **Priority**: whenever incrementing the clock value does not break the local safety predicate between neighbors, the clock value is actually incremented in a finite number of activations, even when no neighbor modifies its clock value.
Sub-classes of unison

Two main properties of unison:

- **Minimality**: nodes maintain no extra variables but the clock.
- **Priority**: whenever incrementing the clock value does not break the local safety predicate between neighbors, the clock value is actually incremented in a finite number of activations, even when no neighbor modifies its clock value.
1 Fault-Tolerance

2 Unison

3 Model
   - Shared Memory
   - Scheduling/Daemon

4 Impossibilities results

5 On the remaining cases...

6 Conclusion
Shared Memory

- Network is modelised by a graph.
- Each processor can read the state of its neighbors...
- ... even if they are crashed.
- But it can only modify its state.
- Processors take action asynchronously.
Shared Memory

- Network is modelised by a graph.
- Each processor can read the state of its neighbors...
  - ... even if they are crashed.
- But it can only modify its state.
- Processors take action asynchronously.
Shared Memory

- Network is modelised by a graph.
- Each processor can read the state of its neighbors...
- ... even if they are crashed.
- But it can only modify its state.
- Processors take action asynchronously.
Shared Memory

- Network is modelised by a graph.
- Each processor can read the state of its neighbors...
- ... even if they are crashed.
- But it can only modify its state.
- Processors take action asynchronously.
Shared Memory

- Network is modelised by a graph.
- Each processor can read the state of its neighbors...
- ... even if they are crashed.
- But it can only modify its state.
- Processors take action asynchronously.
Scheduling

- Distribution:
  - Distributed.
  - Centralized.
  - Locally centralized.

- Fairness:
  - Unfairness.
  - Weak fairness
  - Strong fairness
Scheduling

- **Distribution:**
  - Distributed.
  - Centralized.
  - Locally centralized.

- **Fairness:**
  - Unfairness.
  - Weak fairness.
  - Strong fairness.
Scheduling

- **Distribution:**
  - Distributed.
  - Centralized.
  - Locally centralized.

- **Fairness:**
  - Unfairness.
  - Weak fairness
  - Strong fairness
Scheduling

- Distribution:
  - Distributed.
  - Centralized.
  - Locally centralized.

- Fairness:
  - Unfairness.
  - Weak fairness
  - Strong fairness
Scheduling

- Distribution:
  - Distributed.
  - Centralized.
  - Locally centralized.

- Fairness:
  - Unfairness.
  - Weak fairness
  - Strong fairness
Scheduling

- **Distribution:**
  - Distributed.
  - Centralized.
  - Locally centralized.

- **Fairness:**
  - Unfairness.
  - Weak fairness
  - Strong fairness
Scheduling

Distribution:
- Distributed.
- Centralized.
- Locally centralized.

Fairness:
- Unfairness.
  - Weak fairness
  - Strong fairness
Scheduling

- **Distribution:**
  - Distributed.
  - Centralized.
  - Locally centralized.

- **Fairness:**
  - Unfairness.
  - Weak fairness
  - Strong fairness
Scheduling

- **Distribution:**
  - Distributed.
  - Centralized.
  - Locally centralized.

- **Fairness:**
  - Unfairness.
  - Weak fairness
  - Strong fairness
1. Fault-Tolerance
2. Unison
3. Model
4. Impossibilities results
   - Overview
   - With Two Crashes (or more)
   - With Unfair Daemon
5. On the remaining cases...
6. Conclusion
### Overview

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td>$f = 1, \Delta \geq 3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f = 1, \Delta \leq 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f \geq 2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Overview

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td>$f = 1, \Delta \geq 3$</td>
<td>Imp. $(\forall r)$</td>
<td></td>
</tr>
<tr>
<td>$f = 1, \Delta \leq 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f \geq 2$</td>
<td></td>
<td>Imp. $(\forall r)$</td>
</tr>
</tbody>
</table>
### Overview

<table>
<thead>
<tr>
<th></th>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
<td>Neither</td>
</tr>
<tr>
<td>$f = 1$, $\Delta \geq 3$</td>
<td>Imp. ($\forall r$)</td>
<td>Imp. ($\forall r$)</td>
<td></td>
</tr>
<tr>
<td>$f = 1$, $\Delta \leq 2$</td>
<td>Imp. ($\forall r$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f \geq 2$</td>
<td></td>
<td>Imp. ($\forall r$)</td>
<td></td>
</tr>
</tbody>
</table>
Overview

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td>$f = 1$, $\Delta \geq 3$</td>
<td>Imp. ($\forall r$)</td>
<td>Imp. ($\forall r$)</td>
</tr>
<tr>
<td>$f = 1$, $\Delta \leq 2$</td>
<td>Imp. ($\forall r$)</td>
<td>Imp. ($\forall r$)</td>
</tr>
<tr>
<td>$f \geq 2$</td>
<td></td>
<td>Imp. ($\forall r$)</td>
</tr>
</tbody>
</table>
Overview

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td>$f = 1, \Delta \geq 3$</td>
<td>Imp. ($\forall r$)</td>
<td>Imp. ($\forall r$)</td>
</tr>
<tr>
<td>$f = 1, \Delta \leq 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f \geq 2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Overview

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td>$f = 1, \Delta \geq 3$</td>
<td>Imp. ($\forall r$)</td>
<td>Imp. ($\forall r$)</td>
</tr>
<tr>
<td>$f = 1, \Delta \leq 2$</td>
<td>Imp. ($\forall r$)</td>
<td>Imp. ($\forall r$)</td>
</tr>
<tr>
<td>$f \geq 2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
With Two Crashes

Proposition

For any natural number \( r \), there exists no \((f, r)\)–ftss algorithm for \textbf{AU} under an asynchronous daemon if \( f \geq 2 \).

With \( r = 2 \):
With Two Crashes

**Proposition**

For any natural number \( r \), there exists no \((f, r)\)-ftss algorithm for \( \text{AU} \) under an asynchronous daemon if \( f \geq 2 \).

With \( r = 2 \):
With Unfair Daemon

Proposition
For any natural number $r$, there exists no $(1, r)$-ftss algorithm for AU under an unfair daemon.

Idea:
- $\gamma$ with no crash such that only $p$ is enabled.
- $\gamma'$ the same configuration in which $p$ is crashed $\Rightarrow$ starvation.
- Contradiction $\Rightarrow \gamma$ doesn’t exist.
- Conclusion: always at least two enabled processors.
- Unfair daemon $\Rightarrow$ a processor can be starved if no crash.
With Unfair Daemon

**Proposition**

For any natural number $r$, there exists no $(1, r)$–ftss algorithm for AU under an unfair daemon.

**Idea:**

- $\gamma$ with no crash such that only $p$ is enabled.
- $\gamma'$ the same configuration in which $p$ is crashed $\Rightarrow$ starvation.
- Contradiction $\Rightarrow \gamma$ doesn’t exist.
- Conclusion: always at least two enabled processors.
- Unfair daemon $\Rightarrow$ a processor can be starved if no crash.
With Unfair Daemon

**Proposition**
For any natural number \( r \), there exists no \((1, r)\)–ftss algorithm for AU under an unfair daemon.

Idea:
- \( \gamma \) with no crash such that only \( p \) is enabled.
- \( \gamma' \) the same configuration in which \( p \) is crashed \( \Rightarrow \) starvation.
- Contradiction \( \Rightarrow \) \( \gamma \) doesn’t exist.
- Conclusion: always at least two enabled processors.
- Unfair daemon \( \Rightarrow \) a processor can be starved if no crash.
With Unfair Daemon

Proposition

For any natural number \( r \), there exists no \((1, r)\)–ftss algorithm for \( \text{AU} \) under an unfair daemon.

Idea:

- \( \gamma \) with no crash such that only \( p \) is enabled.
- \( \gamma' \) the same configuration in which \( p \) is crashed \( \Rightarrow \) starvation.
- Contradiction \( \Rightarrow \gamma \) doesn’t exist.
- Conclusion: always at least two enabled processors.
- Unfair daemon \( \Rightarrow \) a processor can be starved if no crash.
With Unfair Daemon

**Proposition**

For any natural number \( r \), there exists no \((1, r)\)-ftss algorithm for \( \text{AU} \) under an unfair daemon.

Idea:

- \( \gamma \) with no crash such that only \( p \) is enabled.
- \( \gamma' \) the same configuration in which \( p \) is crashed \( \Rightarrow \) starvation.
- Contradiction \( \Rightarrow \gamma \) doesn’t exist.
- Conclusion: always at least two enabled processors.
- Unfair daemon \( \Rightarrow \) a processor can be starved if no crash.
With Unfair Daemon

Proposition

For any natural number $r$, there exists no $(1, r)$-ftss algorithm for AU under an unfair daemon.

Idea:

- $\gamma$ with no crash such that only $p$ is enabled.
- $\gamma'$ the same configuration in which $p$ is crashed $\Rightarrow$ starvation.
- Contradiction $\Rightarrow \gamma$ doesn’t exist.
- Conclusion: always at least two enabled processors.
- Unfair daemon $\Rightarrow$ a processor can be starved if no crash.
1. Fault-Tolerance

2. Unison

3. Model

4. Impossibilities results

5. **On the remaining cases...**
   - Which cases?
   - Principle
   - Example

6. Conclusion
### Which cases?

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td>$f = 1, \Delta \geq 3$</td>
<td>Imp.</td>
<td>Imp.</td>
</tr>
<tr>
<td>$f = 1, \Delta \leq 2$</td>
<td>Imp.</td>
<td>Imp.</td>
</tr>
<tr>
<td>$f \geq 2$</td>
<td>Imp.</td>
<td></td>
</tr>
</tbody>
</table>
## Which cases ?

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal</td>
<td>Priority</td>
<td>Neither</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( f = 1, \Delta \geq 3 )</th>
<th>Imp.</th>
<th>Imp.</th>
<th>Imp.</th>
<th>Imp.</th>
<th>Imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f = 1, \Delta \leq 2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pos. (with ( r = 0 ))</td>
</tr>
<tr>
<td>( f \geq 2 )</td>
<td></td>
<td></td>
<td>Imp.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Unfair**
- Weakly fair: Imp.
- Strongly fair: Imp., Imp.

**Weakly fair**
- Minimal: Imp.
- Priority: Imp.
- Neither: Imp.

**Strongly fair**
- Minimal: Imp.
- Priority: Imp.
- Neither: Pos. (with \( r = 0 \))
Principle
Principle

\[ H_p := \frac{\alpha + \bar{\alpha}}{2} \text{ if } h \neq \frac{\alpha + \bar{\alpha}}{2} \]
**Principle**

\[ \overline{p} : \frac{\alpha + \overline{\alpha}}{2} \text{ if } h \neq \frac{\alpha + \overline{\alpha}}{2} \]
Principle

\[ H_p := \begin{cases} 
  h + 1 & \text{if } h + 1 \in l \\
  \min\{l\} & \text{otherwise}
\end{cases} \]
Example
Example
Example
Example
Example
Example
1. Fault-Tolerance
2. Unison
3. Model
4. Impossibilities results
5. On the remaining cases...
6. Conclusion
   - Summary
   - Perspectives
## Summary

Results about \((f, r)\)–ftss AU:

<table>
<thead>
<tr>
<th></th>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
<td>Neither</td>
</tr>
<tr>
<td>(f = 1, \Delta \geq 3)</td>
<td>Imp. ((\forall r))</td>
<td>Imp. ((\forall r))</td>
<td>Imp. ((\forall r))</td>
</tr>
<tr>
<td>(f = 1, \Delta \leq 2)</td>
<td>Imp. ((\forall r))</td>
<td>Imp. ((\forall r))</td>
<td></td>
</tr>
<tr>
<td>(f \geq 2)</td>
<td></td>
<td></td>
<td>Imp. ((\forall r))</td>
</tr>
</tbody>
</table>
## Summary

Results about \((f, r)\)-ftss \textbf{AU}:

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
</tbody>
</table>

- **Unfair**:\(f = 1, \Delta \geq 3\)
  - Imp. \((\forall r)\)
  - Imp. \((\forall r)\)
  - Imp. \((\forall r)\)

- **Weakly fair**: 
  - Minimal
  - Priority
  - Neither

- **Strongly fair**: 
  - Imp. \((\forall r)\)
  - Imp. \((\forall r)\)
  - Pos. \((\forall r)\) (with \(r = 0\))

- **Unfair**: \(f \geq 2\)
  - Imp. \((\forall r)\)
## Summary

Results about \((f, r) - \text{ftss} \ AU:\)

<table>
<thead>
<tr>
<th>Unfair</th>
<th>Weakly fair</th>
<th>Strongly fair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimal</td>
<td>Priority</td>
</tr>
<tr>
<td>(f = 1, \ \Delta \geq 3)</td>
<td><strong>Imp.</strong> ((\forall r))</td>
<td><strong>Imp.</strong> ((\forall r))</td>
</tr>
<tr>
<td>(f = 1, \ \Delta \leq 2)</td>
<td><strong>Imp.</strong> ((\forall r))</td>
<td><strong>Imp.</strong> ((\forall r))</td>
</tr>
<tr>
<td>(f \geq 2)</td>
<td><strong>Imp.</strong> ((\forall r))</td>
<td></td>
</tr>
</tbody>
</table>
Perspectives

- Solve open cases.
- Byzantine failures.
- Bounded clocks.
Perspectives

- Solve open cases.
- Byzantine failures.
- Bounded clocks.
Perspectives

- Solve open cases.
- Byzantine failures.
- Bounded clocks.
Perspectives

- Solve open cases.
- Byzantine failures.
- Bounded clocks.
THANK YOU!