A faulty process is a process that crashes.
A correct process is a process that never crashes.
A set $P$ of processes may crash in step $t$. (Initially stopped, crashes in parallel, or crashes in an arbitrary order.)

**Failure model**: Process crash: a process behaves as if it has crashed a process is defined as having crashed at $t$. A process is said to crash in step $t$ if it has crashed by step $t$.

**Timing model**: Asynchronous: No upper bound on the time required to execute a computation step.

**System model**: An eventually leader oracle and $n$ algorithms and their properties.

**Assumptions**:

- The eventual leader oracle $\mathcal{O}$
- System model

---

**BASE ASYNCHRONOUS MODEL**

---

**Part 1**

---

**From an Intermittent Rotating Star**

---

**From an Intermittent Rotating Star**

---

**Antonio Fernández**

---

**Michel Raynal**

---
Eventually, leader service: definition

Eventually, leader service:

- A black box (object) that provides the process with a "leader" that satisfies the following properties:
  - Primitive denote leader(): it is called each time
  - Validly: Leader() returns a process (node) id each time
  - Eventually: Invocations return the same id, that is the id of an alive process

Eventually, leadership:

- There is a time after which, all leader() invocations return the same id, until a process possibly fails, all is leader-

Part 1

Reliable message-passing communication system

Asynchronous communication model

Asynchronous system model

Fault-free model: Reliable link:

- Every message sent by p i to p j is eventually received by p j (if + is correct)
- No creation, no duplication of messages
- Every pair of processes is connected by a link

Timing model: Asynchronous

- No upper bound on trans-
A sensor network problem

The problem is a well-defined object

By pretty weak properties of them being faulty (i.e.)

It is possible that, before that time, there is an anomaly

forever satisfied is finite but unknown

The time from which the leadership property becomes

Eventual leader service: discussion

Consensus problem

This problem is a particular instance of the famous

order

The replies have to process the commands in the same

operation

Clients issue commands (operations)

A (deterministic) server is replicated

State-machine replication paradigm

This allows a modular decomposition when solving a

problem

First design and prove a protocol based on the abstract properties of

from an implementation (avoiding hardware clocks, network topology, message delays, etc.)

Z is not defined in terms of a particular implementation

Z is defined in terms of abstract properties: (abstract data-

Z is a modular decomposition when solving a

From an implementation (avoiding hardware clocks, network topology, etc.)

Z in action (2)

Z in action (1)

From an implementation (avoiding etc.)
Process alive[\(u\)](\(t\)) message received by its destination message alive[\(u\)](\(t\)) is winning if it belongs to the first destination process at most \(g\) time units after it has been sent. A message alive[\(u\)](\(t\)) is \(g\)-timely if it is received by its destination.

**Messages definition**

---

**Round-based computation model**

---

**Disruption-free to Non-blocking** is ... 2.

- The weakest contention manager for going from Non-blocking STM
- Obstruction-free STM

**Software transactional memory**

---

**The Assumptions**

- A and A
Property in (\(T_{ct}\)) of the \(g\)-fairy property in \(G(T_{ct}(1))\) and the \(g\)-fairy property in \(G(T_{ct}(2))\) can satisfy the \(g\)-fairy property in \(G(T_{ct}(2))\) when the other one satisfying the \(g\)-fairy property, while the other one \(G(T_{ct}(d))\) is allowed to satisfy different properties. If \(b \leftarrow d\) \((\lambda_{ct}(g) \cap \{d\})\) at any \(t\), two points of the star fail with \(\lambda_{ct}(g) \cap \{d\}\), \(\lambda_{ct}(g) \cap \{d\}\), and \(\lambda_{ct}(g) \cap \{d\}\).

Dynamic notions in AS" +

Illustration: a rotating star

System model AS" +

\[ u < (1 + t) + (t - u) \]

\[ t = 2 \quad 6 = u \]
By the bounded 2

The limitation on this dynamicity dimension is expressed by the following.

A defines an intermittent rotating l-star centered at p from time to time. This is why referred to it.

A wears ++

When D = 1, A boils down to +

A adds dynamicity to ++

Verifying A++ to obtain A

Particular cases, cont'd

- The message (time) is unbounded.
- The message (time) is bounded.
- The message (time) is constrained.
- Properties are satisfied.

Finding the properties A1 and A2

For any \( \forall k \leq \forall n \in \infty \), there is a set of processes \( \forall n \) such that for \( \forall n+1 \), all numbers \( \forall n+1 \) are not necessary.

There is an infinite sequence of increasing round numbers.

Verifying the following properties:

- for any \( \forall m < \forall m \), there is a set of processes \( \forall m \) such that for \( \forall m+1 \), and bound \( \forall m+1 \), and bound \( \forall m \).

There are \( \forall m \) processes. Two bounds, bound \( \forall m \).

Particular cases, cont'd

Verifying A++ to obtain A

Particular cases
Recevier side of $p_i$}

\[ \text{\textbf{Receiver side of } } p_i \]

\[
\begin{align*}
\text{\textbf{Receiver side of } } p_i \\
\{ f \} \cap [\text{alive}_t] & \to \text{receive}_t \\
\text{if } \text{max} \{ \text{send}_t \} & > [\text{send}_t] \text{ end-do} \text{ do} \text{ end-do} \\
\text{end-for each } k \text{ do} \text{ end-for each } k. \text{ do} \\
\end{align*}
\]

\[ \text{Upon reception alive}(m, s) \text{ from } p_k \]

\[ \text{end}\]

sender side of $p_i$}

\[ \text{\textbf{Sender side of } } p_i \]

\[
\begin{align*}
\text{\textbf{Sender side of } } p_i \\
\text{An } A^*\text{-based Algorithm} \\
\end{align*}
\]

\[ \text{An } A^*\text{-based Algorithm} \]
Background task of $p_i$ (part 2)
... • They stop suspecting if behavior of a process agrees with its announced promise, • As soon as the i other processes recognize the current • Its actual behavior to the i other processes one of them has to be correct and give a promise on • Enough processes (i+1) have to be involved (star struct-

What do we really need to implement? ?

end-do

if [y]!sprechert → [y]!sprechert

then susp(sprechert) \[ n \| \min = [y]!sprechert(\min = [y]!sprechert)

(1 - u \leq [y,x]!sprechert(sprechert))

\( u > x > [y]!sprechert \) \( \forall [x,u]!sprechert(sprechert) \) \( \forall [y,u]!sprechert(sprechert) \) \( \forall \)

foreach \( k \in \text{sprechert} \) from \( p \) do

upon reception susp(sprechert) from \( p \) it bounded

Even the timers are now bounded • Answer: speechs only if it is the current leader

Am: bound each susp(sprechert) [y] variable

Bounding all the variables (but the round numbers)