

Fundamentals of
Fault-Tolerant
Distributed Computing in
Asynchronous Environments
by Felix Gärtner

Tye Stallard
24 March 2004

Thanks to

- Graeme Coakley
 - <http://students.engr.scu.edu/~gcoakley/>
 - For COEN 317 at San Jose State

Overview

- Goals
 - Structure the field
 - Survey the fundamental building blocks
- Structure
 1. Terminology
 2. Redundancy
 3. Safety
 4. Liveness
 5. Conclusion

Terminology

States, Configurations and Guarded Commands

- Distributed system: finite set of processes.
- Local state: variables of each process.
- Event: State transition
 - send, receive, or internal event
- Actions: written as "guarded commands"
 - abstractly represent a local algorithm
 - $\langle \text{guard} \rangle \Rightarrow \langle \text{command} \rangle$
- Configuration: consists of local states of all processes plus state of communication subsystem.

process Redundancy

var $x \in \{0, 1, 2\}$ **init** 1 **{* local state *}**

begin

{* normal program actions: *}

$x = 1 \quad \Rightarrow \quad x := 2 \text{ \{*\ 1 *\}}$

$[] \quad x = 2 \quad \Rightarrow \quad x := 1 \text{ \{*\ 2 *\}}$

$[] \quad x = 0 \quad \Rightarrow \quad x := 1 \text{ \{*\ 3 *\}}$

{* fault action: *}

$[] \quad \text{true} \quad \Rightarrow \quad x := 0$

end

Terminology (continued)

Defining Faults and Fault Models

- Fault Class / Fault Model:
 - Crash, Fail-stop, Byzantine
 - Omission, Timing, Performance
- Fault: "an action on the possibly extended state of a process"
- Fault: *every* kind can be modeled as an unwanted state transition of a process (92', 93', 98')
- Failure: system violates its specification

Terminology (continued)

Properties of Distributed Systems:

Safety and Liveness (77')

- Safety property: some specific "bad thing" never happens within system.
- Liveness property: some specific "good thing" will eventually happen during system execution (e.g. *termination*)
- Problem Specification: consists of a safety and a liveness property

Formal View of Fault Tolerance

Definition 1:

- A distributed program A is said to *tolerate faults* from a fault class F for an invariant P iff there exists a predicate T for which the following requirements hold:
 - If P holds, T also holds (i.e. $P \Rightarrow T$)
 - T is closed in A and T is closed in F
 - Starting from any state where T holds, every computation that executes actions from A alone eventually reaches a state where P holds

Redundancy in space and redundancy in time

process Redundancy

var $x \in \{0, 1, 2\}$ **init** 1 *{* local state *}*

begin

{ normal program actions: *}*

$x = 1 \quad \Rightarrow \quad x := 2 \quad \{ * 1 * \}$

$x = 2 \quad \Rightarrow \quad x := 1 \quad \{ * 2 * \}$

$x = 0 \quad \Rightarrow \quad x := 1 \quad \{ * 3 * \}$

{ fault action: *}*

true \Rightarrow $x := 0$

end

Redundancy as the Key to Fault Tolerance (continued)

Claim:

- If A is a nontrivial distributed program that does not employ redundancy, then A may become incorrect regarding its correctness specification in the presence of faults.

Conclusion:

- While redundancy is not sufficient for fault tolerance, it is a necessary condition.
- Redundancy in space is widespread

Models of Computation

Models of Distributed Systems

- Synchronous systems: real-time bounds on message transmission and process response
- Partially synchronous: intermediate models that have bounds to a varying degree.
 - Timed asynchronous 98 and quasi-synchronous 98'
- Asynchronous systems: no bounds made.
 - 'Weakest' model and realistic model in many applications.
 - Every algorithm that works on this model, works on all other models.
 - Impossible to detect a crashed process (85')

Achieving Safety: Detection as the Basis for Safety

- To ensure safety, we need to employ detection and subsequently inhibit dangerous actions.
- Common Detection Mechanisms:
 - parity, checksums, duplicate calculations
- Detection includes checking whether a certain predicate Q holds over the entire system
 - Q is easier to specify if the type and effect of faults from F are known.

Achieving Safety: Detection in Distributed Settings

- Deciding whether a predicate over the global state does or does not hold is not easy.
- Cooper and Marzullo introduced two transformers (modal logic):
 - Possibility(Q) is true iff there exists a continuous observation of the computation for which Q holds at some point.
 - Definitely(Q) is true iff for all possible continuous observations of the computation Q holds at some point.
- Place detector at unsafe point in the fault span

Achieving Liveness: Correction

- Liveness tied to notion of correction.
- Correction refers to turning a bad state into a good one
 - "The basic idea is to impose a state predicate on the system in cases where possibly illegal or unsafe states are detected"
- Common methods include:
 - retransmission, error-correction codes, rollback recovery, rollforward recovery, etc.
- On detecting a bad state via a detection predicate Q , the system must try to impose a new target predicate R onto the system.

Achieving Liveness: Correction via Consensus

- Correction corresponds to the decision phase of consensus algorithms.
- State machine approach (Schneider)
 - Servers are made fault tolerant by replicating them and coordinating their behavior via consensus algorithms.
- Other methods based on several forms of fault-tolerant broadcasts.

Related and Current (99')

- Detection and Correction from 1976
- Multitolerance (Arora and Kulkarni98)
- Four forms of Fault Tolerance
 - Traditionally, mostly locally shared variables with tight synchrony assumptions

Related and Current (99')

- Consensus
 - Popular until impossibility theorem (85')
 - Unreliable failure detectors "has made this problem a little more feasible" (96')
- Failure detectors / suspectors (93')
 - Four papers have been written beyond the crash fault class (97'-98')

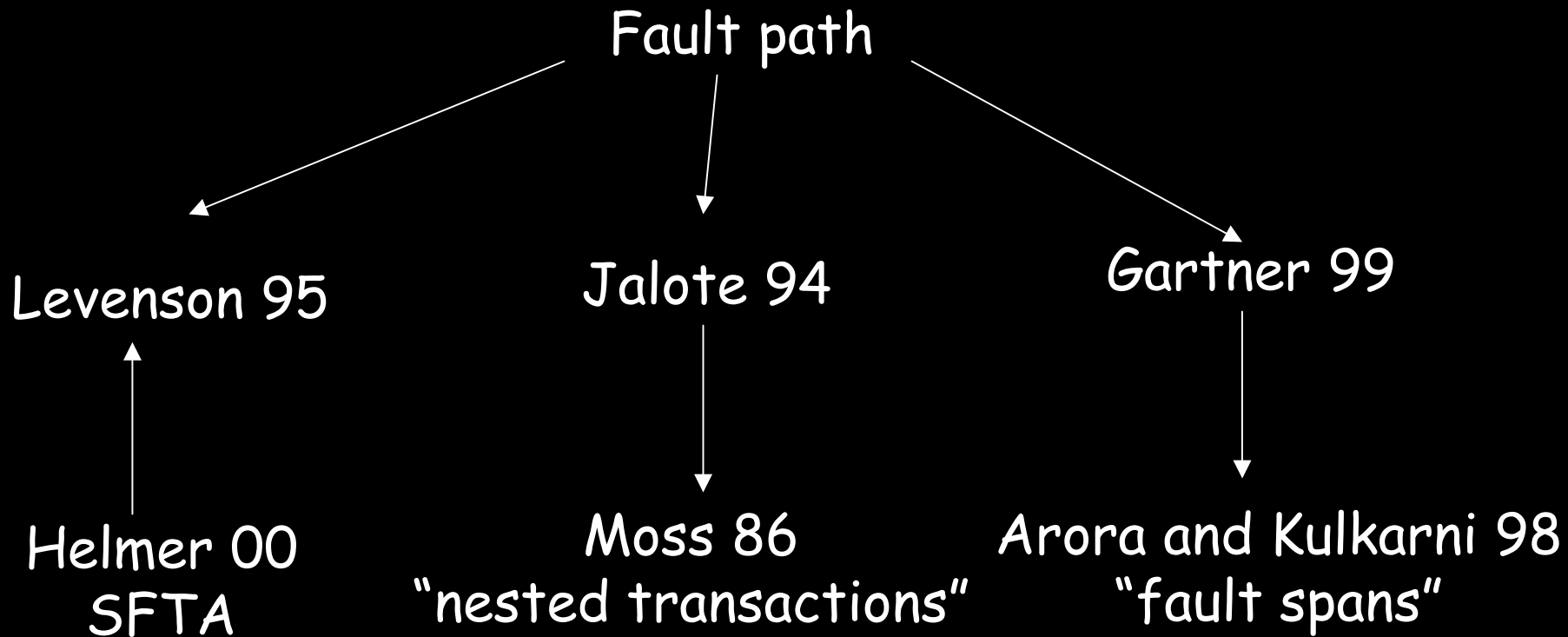
Future work

- Fault diagnosis
- Reliable communication
- Network topology and partitions
- Software design faults
- "Evaluation and simulation of practical systems is key"

The paper's contributions

- Good terminology
- Formalization of redundancy & consequences
- Generalized the four types of fault tolerance
- ~110 references

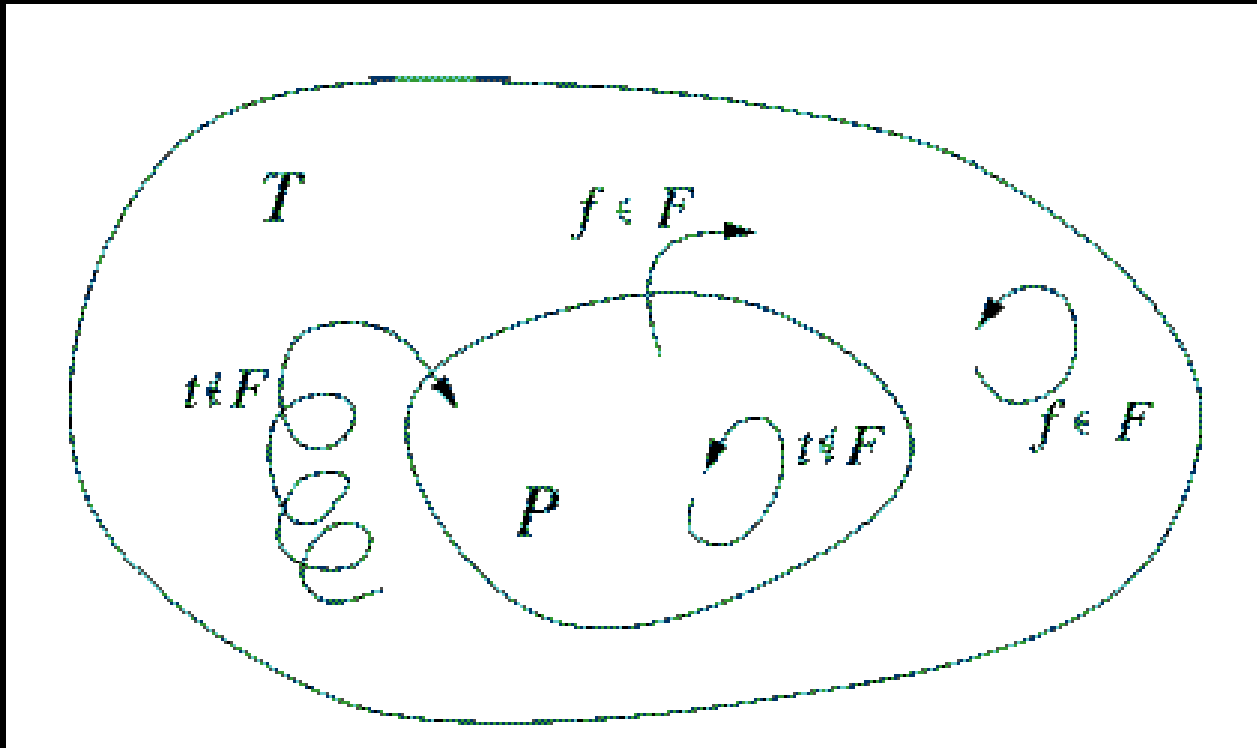
Fault trees & attack graphs



Four Types of Fault Tolerance

		Liveness Property Satisfied	
		Yes	No
Safety Property Satisfied	Yes	Masking	Fail Safe
	No	Nonmasking	None

Schematic overview of the definition 1



State sets

