

Distributed Systems

- ◆ Distributed versus Parallel
 - Loosely coupled vs. tightly coupled
 - » shared memory
 - » synchronized clocks, global clocks
 - » network topology
 - fully connected,
 - partially connected, e.g. ring, tree, star, k-connected
 - » network communication
 - point-to-point
 - CSMA/CD (Carrier Sense Multiple Access, Collision Detect)

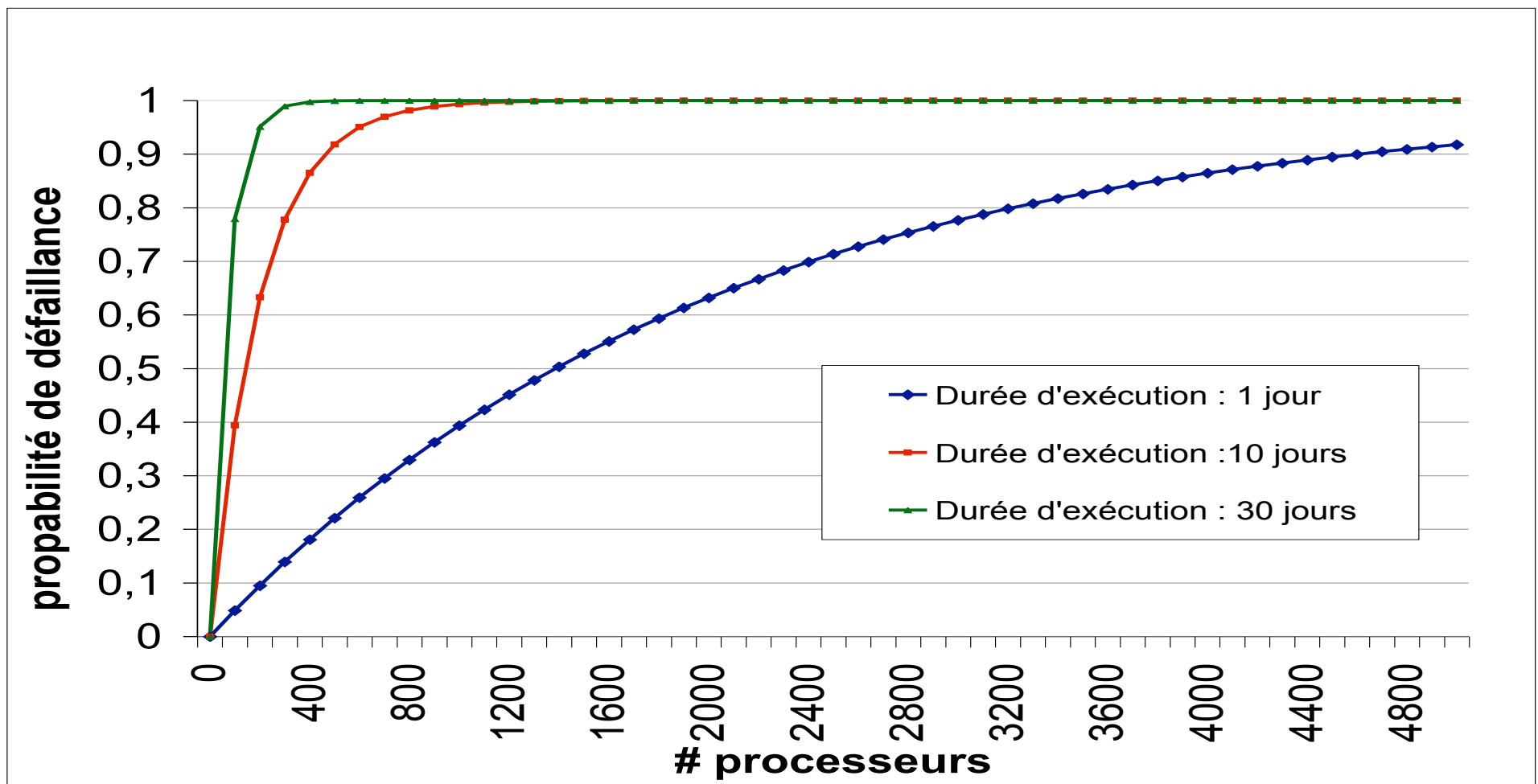
Unreliability in the absence of FT

Computation on Cluster

MTBF = 2000 days (48,000h, approx. 5 1/2 years)

Unreliability of one node: $F(t) = 1 - R(t) = 1 - e^{-t}$

Figure source: Jafar, Krings, Gautier and Roch, EIT 2005



Distributed Systems

◆ Design Issues

- circuit and technology level
 - » radiation robust components, gallium arsenide chips
 - » circuit designed for testability and reliability
- node-level architecture
 - » design of VLSI
- internode architecture
 - » node connections and configurations, reconfiguration in the presence of faults
- operating system
 - » fault recovery, e.g. rollback procedure
 - » load balancing
- application level
 - » checks on results, signatures, bounds

Distributed Systems

- ◆ Asynchronous Message Passing
 - send(data, destination)
 - receive(data, source)
 - receive(message)
 - » source is extracted from message itself
 - queuing messages => finite buffers
 - » in reality no pure asynchronous message passing exists
 - » with finite buffer receiver may block

Distributed Systems

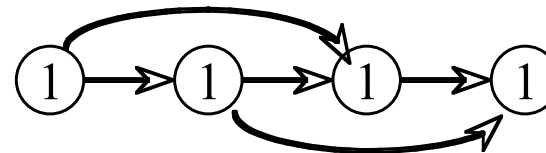
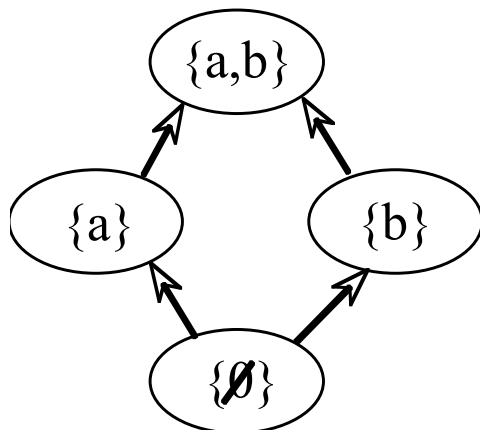
- ◆ Synchronous Message Passing
 - no buffering
 - guards
 - » execute statements if the guards are “true”
 - alternative construct
 - » list of guarded statements
 - » non-determinism
 - CSP (Communicating Sequential Processes)
 - » to send a message:
P ! message
 - » to receive a message:
P ? message

Distributed Systems

- ◆ Remote Procedure Call (RPC)
 - the service to be provided by the server is treated as a procedure that resides on the machine on which the server is located.
 - failure during communication
 - » unwanted executions called orphans
 - a client that crashes during an RPC and restarts on recovery may reissue the call to the server, even though the earlier call is still being executed by the server

Ordering -- Synchronizing

- ◆ Which event happened first?
 - When events happen on distributed systems it is often not clear which events happened first.
- ◆ Partial Order
 - A partial order is a Relation R , that is
 - » irreflexive, i.e. $(a, a) \notin R$ for any a and
 - » transitive, i.e. $(a, b)(b, c) \Rightarrow (a, c)$
- ◆ Partial Order and Total Order



Ordering -- Synchronizing

- ◆ Define “happened-before” as partial order

$a \rightarrow b \equiv a$ happened before b

$a \rightarrow b \quad b \rightarrow c \implies a \rightarrow c$

$a \not\rightarrow b \quad b \not\rightarrow c \implies$ events are concurrent

- Note, PO does not define a total order

Ordering -- Synchronizing

◆ Logical Clock

Let C_i denote the logical clock for process P_i

$C_i(a)$ is the value associated with event a of P_i

◆ Clock Conditions

- C1: if a & b are events in process P_i ,
 - » and a comes before b ,
 - » then $C_i(a) < C_i(b)$

- C2: if a is the sending of a message by process P_i and b is the receipt of that message by process P_j ,
 - » then $C_i(a) < C_j(b)$

Ordering -- Synchronizing

- ◆ Satisfy C1:
 - each P_i increments C_i between any two successive events
- ◆ Satisfy C2:
 - if event a is the sending of a message m by P_i , then m contains a time-stamp T_m , with $T_m = C_i(a)$
 - upon receiving of m , process P_j sets C_j greater than or equal to its present value and greater than T_m .
- ◆ Logical clocks have no relationship to actual, physical clocks