Fault Tolerance

ARCOS Group

Distributed Systems
Bachelor In Informatics Engineering
Universidad Carlos III de Madrid

Introduction

The reliability of a system measures how well it conforms to its specification

A defect is a deviation from the expected external behavior

- Result of internal errors

The cause (mechanical, algorithmical, etc) of an error is a failure (fault)

- May be Hw or SW

Examples of failures

Ariane 5 explosion in 1996

- Sent by ESA in june of 1996 (first trip)
- Development cost: 10 yrs, $7000 mil.
- Exploded 40s after takeoff, at 3700m
- The failure was due to the lost of height information
- Cause: software design error
- Conversion of real floating point nr on 64 bits to int on 16 bits results in value > 32767 (max int on 16 bits) therefore exception.

Examples of failures

Patriot missiles

– Used in the 1991 Gulf war to intercept Scud missiles
– Failure due to error in timing
– Internal clk expresses tenths of sec as int, which gets converted to real on 24 bits (with corresponding loss in precision)

Examples of failures

Viking probe sent to Venus

instead of Fortran DO 20 I = 1,100

(i.e. 100 iteration loop over label 20) it said

DO 20 I = 1.100

Compiler does not consider spaces so interprets as

DO20I = 1.100

(i.e. var declaration (O20I) with value 1.100)

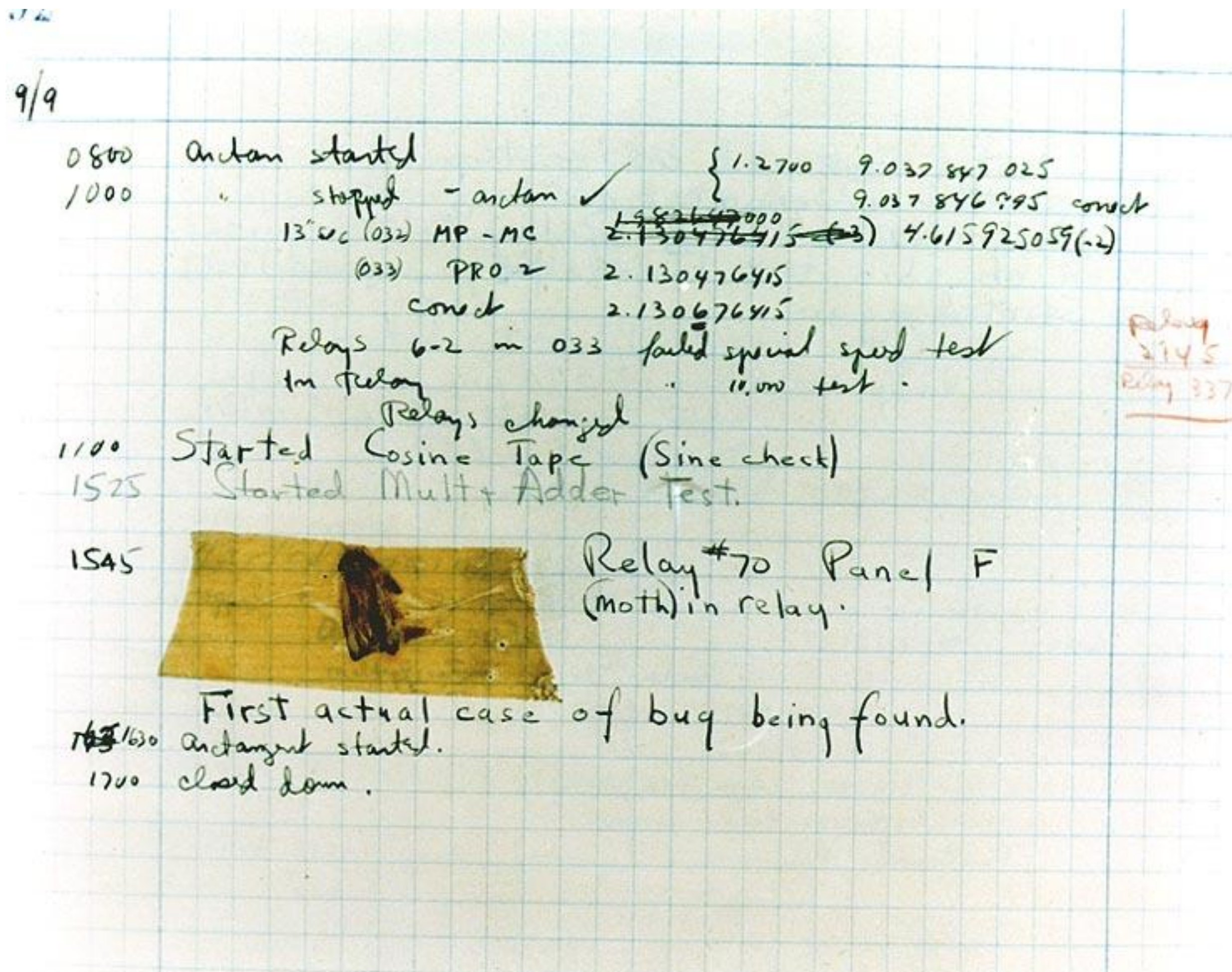
D identifies a real type

Examples of failures

The Spirit robot

- Had a RAD6000 similar to an IBM RS6000 (with radiation protection) with an early-generation PowerPC processor w. 128 MB RAM and 256 MB flash memory
- Uses RTOS VxWorks within 32 MB RAM
- The very file needs some OS-controlled information
- The 32 MB ended up full of data associated to thousands of files and OS rebooted
- Every reboot ended up in memory full and subsequent reboot therefore systems seemed not working

The moth inside the computer… bug

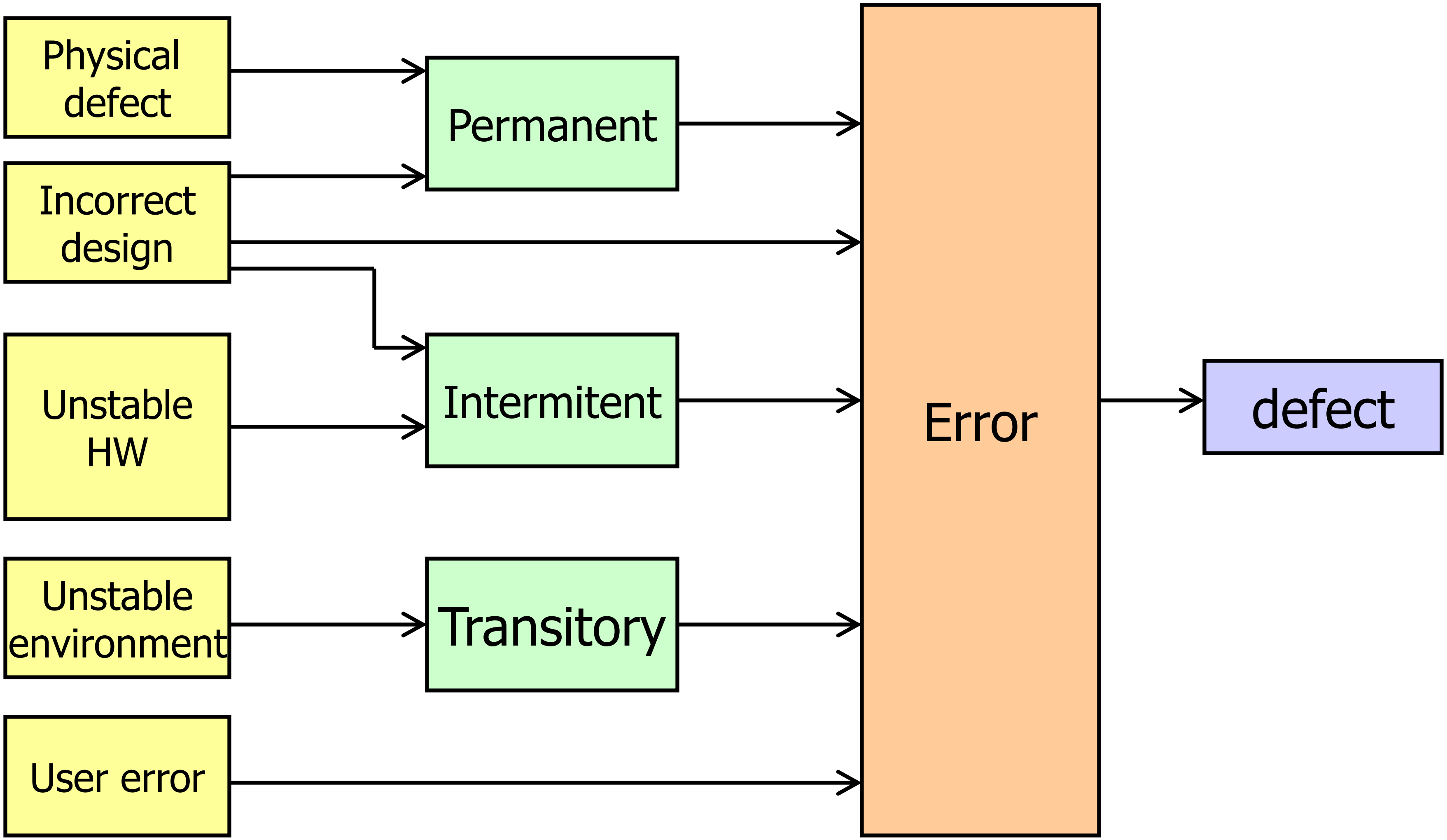


Examples of failures

- More on SW failures at:

http://www.cs.tau.ac.il/~nachumd/verify/horror.html

Failure types



Definition

Fault tolerant system:

– A system capable to ensure the correct and uninterrupted execution despite HW / SW failures.

Goal:

– High reliability

Fault tolerance in distributed systems:

– Basic approach: Replication

Replication

Goals

- Better performance (caché)
- Better disponibility
  - If p is the failure prob. in a server
  - With n servers the failure prob. is p^n

Replication types

- Data
- Processes

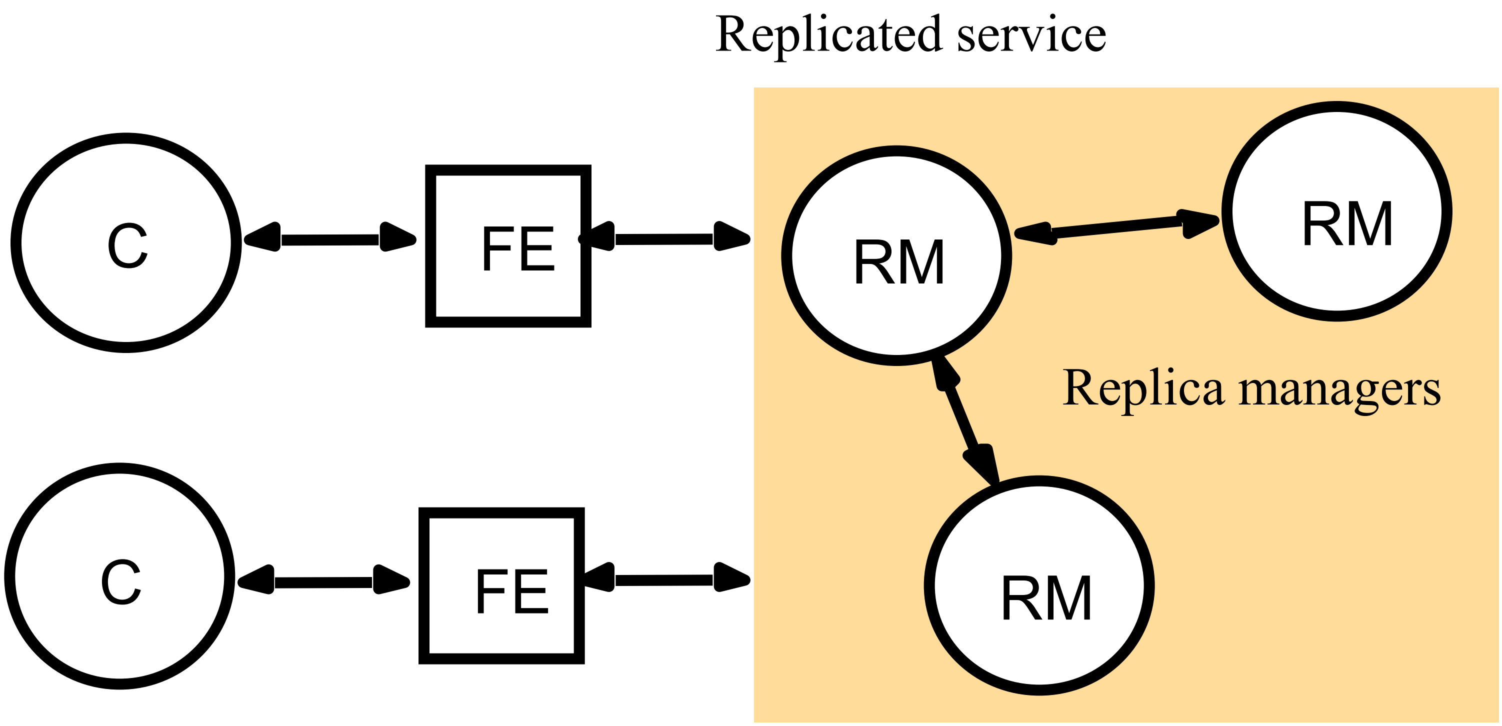
Problems

- Consistency

Requirements

- Transparency
- Consistency
- Performance

Basic replication scheme



Front-end: makes replication transparent

Replication methods

Pesimistic: ensure consistency of replicas

- Primary copy
- Active replicas
- Voting schemes (quorum)
  - Static
  - Dynamic

Optimistic: no consistency

- No limitations on data use when failure
- E.g.: version vectors, CODA file system

Consistency models

Strong:

- Pesimistic replication
- Total consistency

Weak:

- Optimistic replication
- Local W w/o restrictions
- Modifications which result in inconsistencies must be rolled back or corrected
- Good for cases with few concurrent W processes

Primary copy

If k failures one needs k+1 copies

- Primary copy
- K nodes w replicas

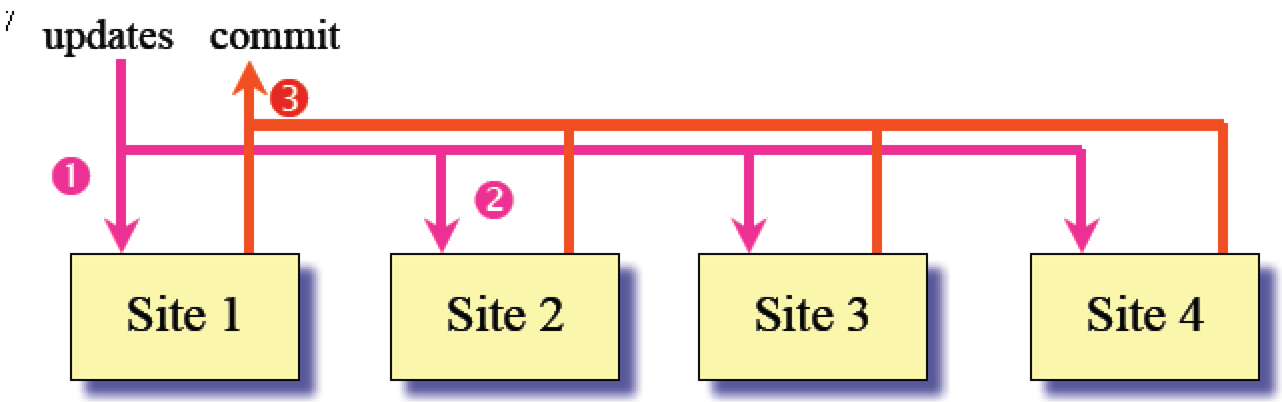
R: send to primary

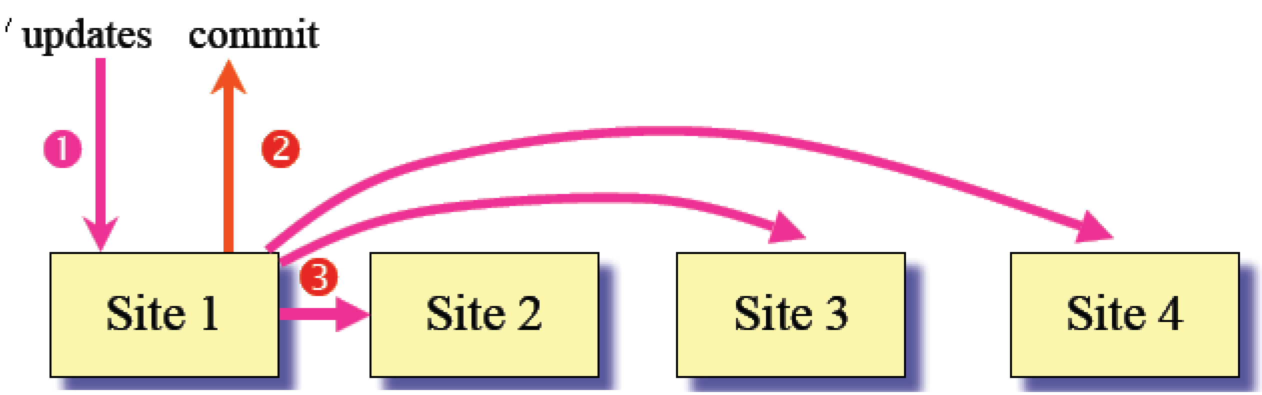
W: send to primary

- Primary does the update
- And updates the rest of the replicas
- Then it replies to the client

When primary fails another node replaces it

Replica synchronization





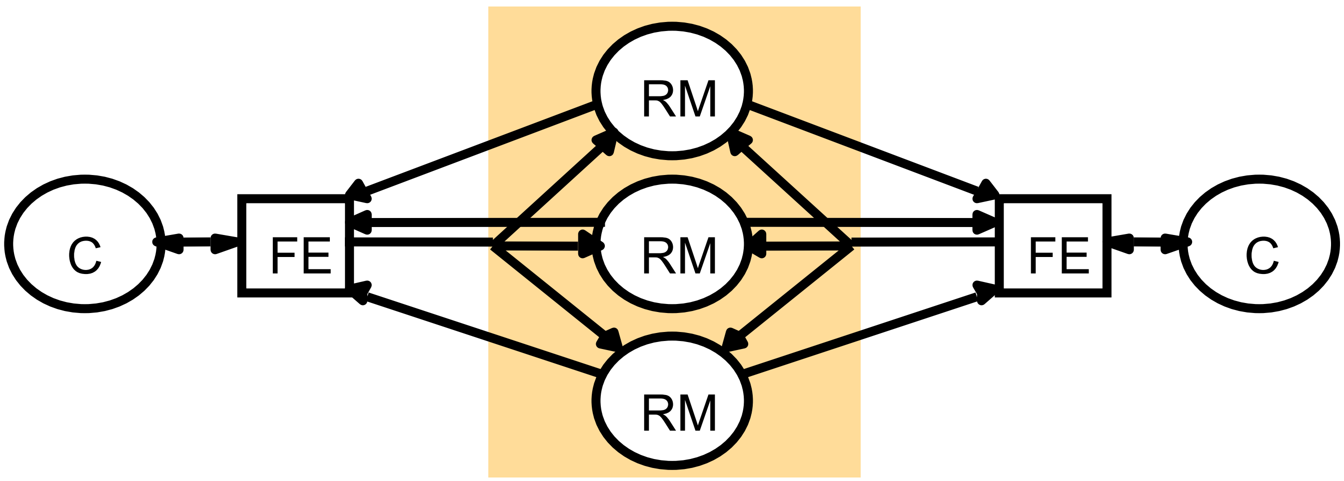
Active replicas

All nodes act as primary

– Better performance on R

Atomic multicast necessary on W!

– W order is important



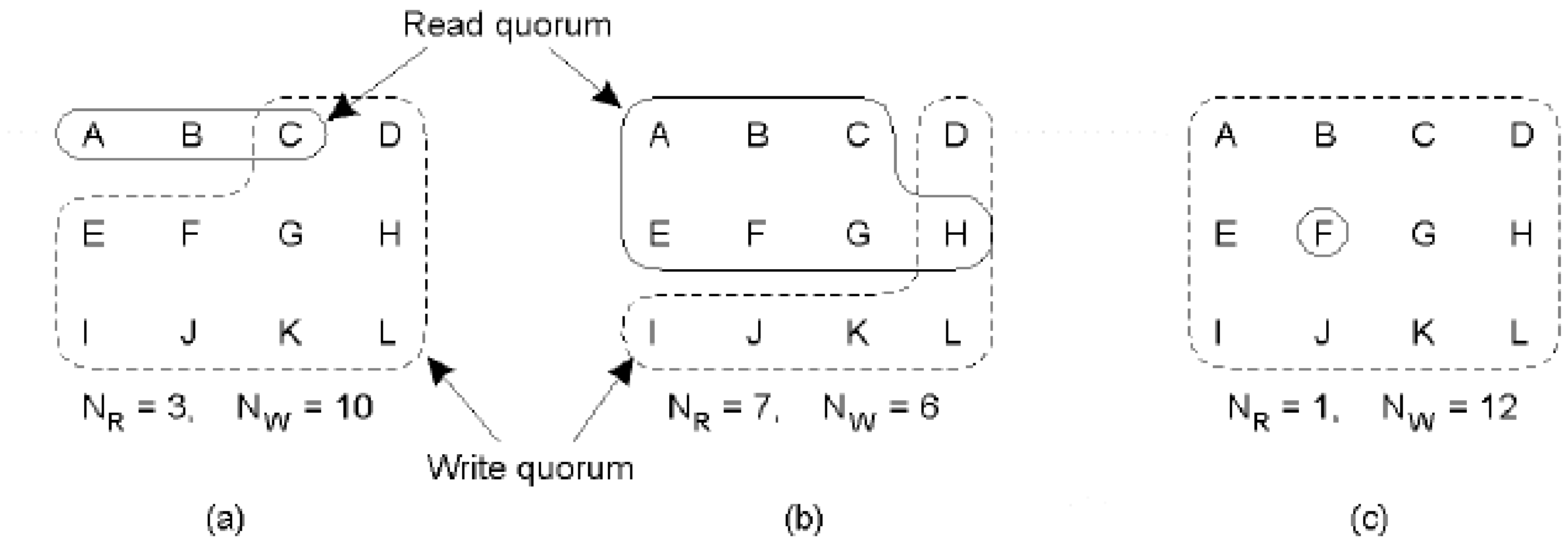
Voting schemas (quorum)

Define two operations: READ, WRITE

N nodes that serve requests

- The READ operation is done on the R copies
- The WRITE operation is done on the W copies
- Every replica has a version number V
- It must be the case that:
  - **$R + W > N$**
  - **$W + W > N$**
  - **$R, W < N$**

Examples of quorums



Invalid!

How to choose W and R?

Two factors are important:

- Performance: depends on the % of R / W and their cost
  - Total cost = cost R * p * nr_R + cost W * (1- p) * nr_W
- Fault tolerance: depends on the probability of failure
  - Prob. failure = Prob. failure R + Prob. failure W

E.g.:

- N=7
- Cost of W = 2 times cost of R (K)
- Percentage of R (p) = 70%
- Prob. failure = 0.05

Reliability for different configurations

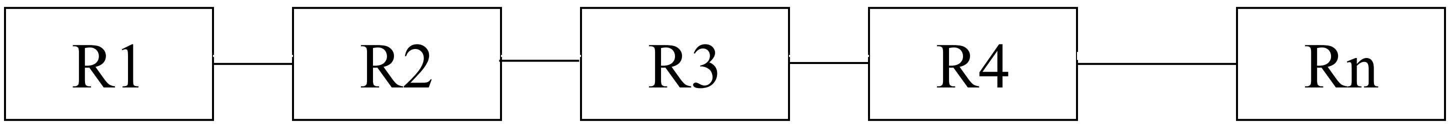
Series configurations:

Ri (t) = reliability of component I

configuration fails whenever SOME component fails

assume independent failures

overall reliability: $R(t) = \prod_{i=1}^{N} R_i(t)$

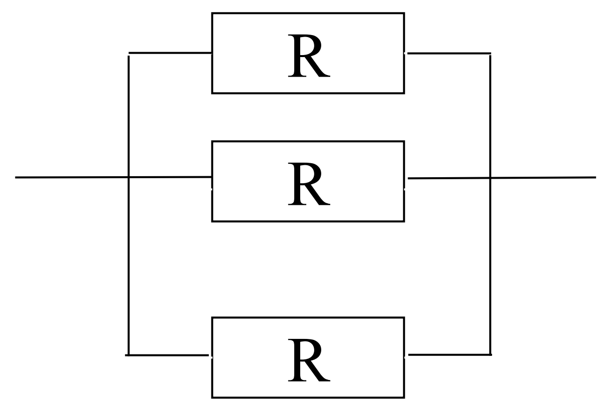


Parallel configurations:

ALL components must fail

overall reliability: 1 – prob. that all fail

$$R(t) = 1 - \prod_{i=1}^{N} Q_i(t) \quad \text{donde} \quad Q_i(t) = 1 - R_i(t)$$



Solution

R	W	Coste	Probabilidad de fallo en R	Probabilidad de fallo en W	Probabilidad de fallo
1	7	4,9	7,8125E-10	0,301662704	9,05E-02
2	6	5	1,04688E-07	0,044380542	1,33E-02
3	5	5,1	6,02734E-06	0,003757043	1,13E-03
4	4	5,2	0,000193578	0,000193578	1,94E-04

Voting methods

READ

– Read all R copies, mantain last version

WRITE

– First READ to figure out version number
– Initiate a 2PC to update data and version number in W copies

two-phase commit

Two-phase-commit (2PC)

There exists a coordinator

Coordinator:

multicast: *ok to commit*?

wait for replies

all ok => *send(commit)*

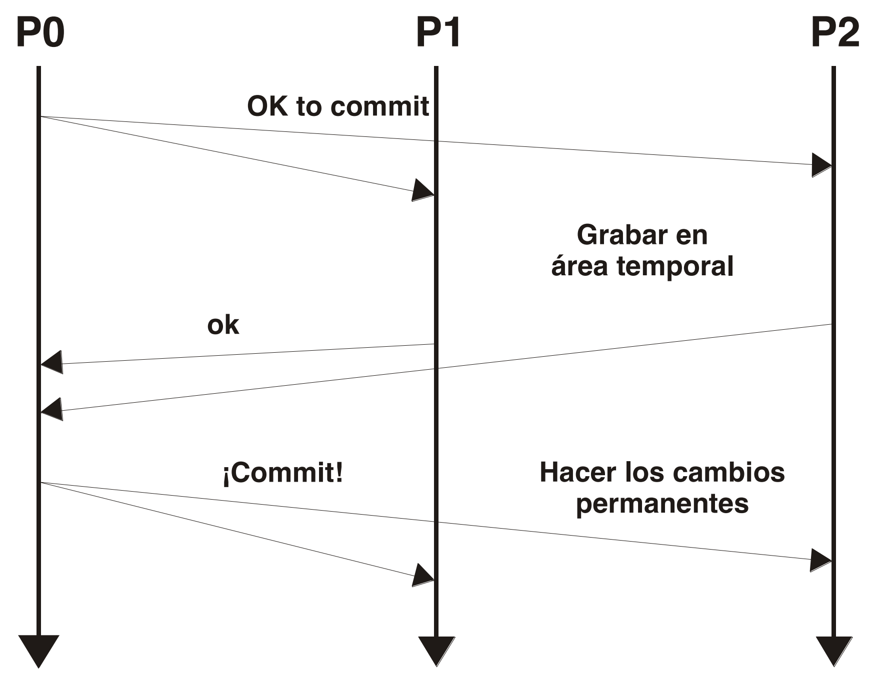
else => *send(abort)*

Processes:

ok to commit=> save changes, reply *ok*

commit => commit changes

abort => erase changes

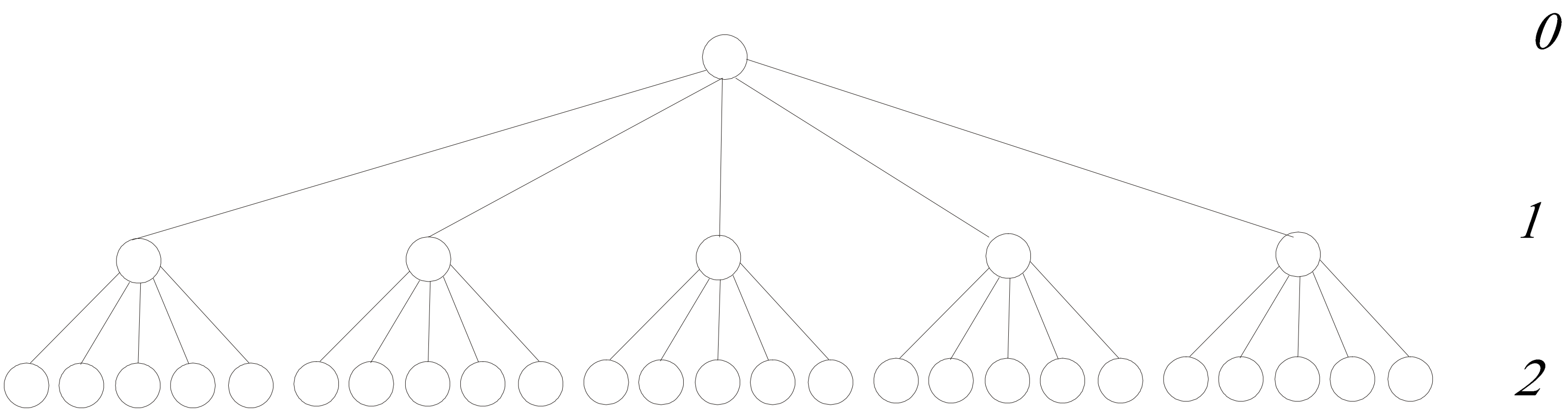


Hierarchical voting

W grows with the number of replicas

Solution: hierarchical voting

- E.g.: replica nr = 5 x 5 = 25 (leaves)



Hierarchical voting

quorum on levels

R1	W1	R2	W2	RT	WT
1	5	1	5	1	25
1	5	2	4	2	20
1	5	3	3	3	15
2	4	2	4	4	16
2	4	3	3	6	12
3	3	3	3	9	9

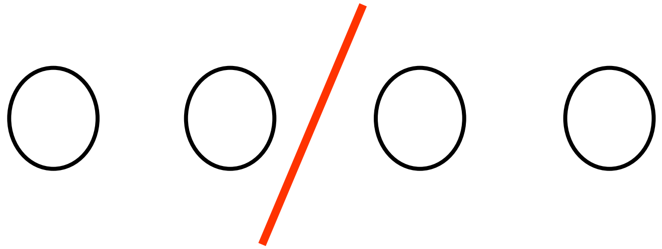
Which one we choose?

Dynamic methods

Previous methods (static) are not adaptative when failures occur

E.g.:

- 4 replicas with R=2 y W=3
- Given following partition



- Cannot write

Dynamic voting

Each data d exists on N replicas $\{d1..dn\}$

Each data di on node i has a version number VNi (initially 0)

Actual version: $NVA(d) = \max\{VNi\} \quad \forall\ i$

A replica of di is actual if $VNi = VNA$

A group is a majority partition if it contains a majority of actual copies of d

Each di has a cardinality update number associated SCi = nr of nodes that participated in the update

Dynamic voting

Initially $SCi = N$

When updating di

- SCi = nr of copies of d updated during the current W

A node may W if it is a member of a majority partition

W algorithm

$$\forall\ i \text{ requests } NVi \text{ and } SCi$$

$$M = \max\{NVi\} \text{ including itself}$$

$$I = \{i \text{ s.t. } VNi = M\}$$

$$N = \max\{SCi,\ \ i \in I\}$$

$$\mathbf{if}\ \left|I\right| \leq N/2$$

then

node not member of majority partition, don't W

else {

$\forall$ nodes $\in\ I$

update

$VNi = M+1$

$SCi = /I/$

}

E.g.

N= 5

Initially:

	A	B	C	D	E
VN	9	9	9	9	9
SC	5	5	5	5	5

partition:

	A	B	C	D	E
VN	9	9	9	9	9
SC	5	5	5	5	5

E.g.

W on partition 2?

- $M = \max\{9, 9\} = 9$
- $I = \{D, E\}$
- $N = 5$, $|I| = 2 \leq 5/2 \Rightarrow$ nope

W on partition1?

- $M = \max\{9, 9, 9\} = 9$
- $I = \{A, B, C\}$
- $N = 5$
- $|I| = 3 > 5/2 \Rightarrow$ yep

	A	B	C	D	E
VN	10	10	10	9	9
SC	3	3	3	5	5

E.g.

New partition

	A	B	C	D	E
VN	10	10	10	9	9
SC	3	3	3	5	5

Partition 1 Partition 2 Partition 3

W on partition1?

- N=max{10,10} = 10
- I = {A, B}
- N= 3
- $|I| = 2 > 3/2 \Rightarrow$ yep

E.g.

	A	B	C	D	E
VN	11	11	10	9	9
SC	2	2	3	5	5

Partition 1 Partition 2 Partition 3

When a node joins a group

…it must update its state:

$$M = \max\{VNi\}$$

$$I = \{\text{Aj, s.t. } M = VNj\}$$

$$N = \max\{SCk,\ k \in I\}$$

$$\textbf{if } |I| \leq N/2$$

then

cannot join

else {

update state

$VNi = M$

$SCi = N + 1$

}

E.g.

join 2 and 3

	A	B	C	D	E
VN	11	11	10	9	9
SC	2	2	3	5	5

Partition 1 | Partition 2

W to partition 2?

- $M = \max\{10, 9, 9\} = 10$
- $I = \{C\}$
- $N=3$
- $|I| = 1 \leq 3/2 \Rightarrow$ nope

E.g.

Join 1 and 2

	A	B	C	D	E
VN	11	11	10	9	9
SC	2	2	3	5	5

Partition 1 Partition 2

W to partition 1?

- $M = \max\{11, 11, 10\} = 11$
- $I = \{A, B\}$
- $N = 2$
- $|I| = 2 > 2/2 \Rightarrow$ yep

E.g.

	A	B	C	D	E
VN	12	12	12	9	9
SC	3	3	3	5	5

Partition 1 Partition 2

Replication of the CODA file system

Optimistic replication

Every copy has a version vector V w. n components = replication degree

For node i $Vi[j]$ is the number of updates on j's copy

W/o network failures all vectors are the same, otherwise they differ

Given $V1$ & $V2$, $V1$ dominates $V2$ iff $V1(i) \geq V2(i)\ \forall\ i$

If $V1$ dominates $V2$ there are more updates in the copy of $V1$

$V1$ & $V2$ in conflict if neither dominates

When groups join they compare vectors

- The copy of group w the dominant vector updates the second
- If conflicts then error

E.g.

3 servers {A, B, C}

Initially $V = (0,0,0)$ for all

On update: $V=(1,1,1)$ for all

Assume network failure: Group 1: {A,B}, Group 2: {C}

Update group 1: $V=(2,2,1)$ for group 1

Assume network failure: Group 1: {A}, $V=(2,2,1)$

Group 2: {B, C}

- $(2,2,1) \geq (1,1,1) \Rightarrow$ update C's copy, $V = (2,2,2)$ for B, C

Update group 2: $V=(2,3,3)$ for {B,C}

E.g. (cont'd)

Case 1: join {A} and {B,C}

– (2,2,1) ≤ (2,3,3) ⇒ update {A}'s copy, V =(3,3,3)

Case 2:

– Modify {A}'s version ⇒ en A, V= (3,2,1)
– Join A w. V=(3,2,1) with {B,C} w. V=(2,3,3)
– Compare (3,2,1) and (2,3,3) neither dominetes! **conflict**