Dependability (3)

Impairment:

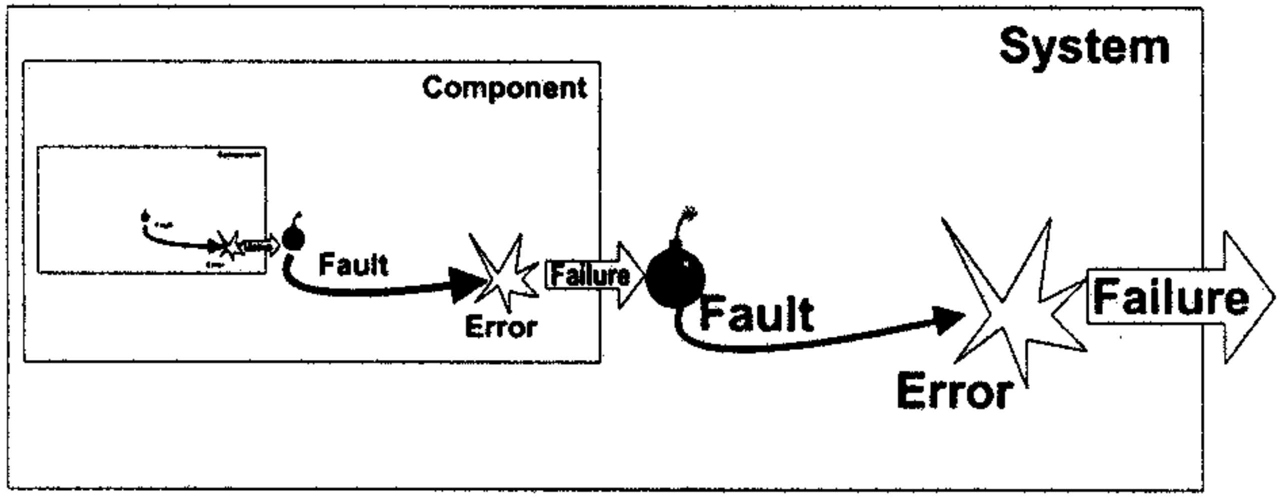
failure: fact that the system behavior (delivered service) violates (deviates from) its specification

What leads to failures?

fault: occurrence of an irregular event in (some component of) the system

error: manifestation of the fault in (part of) the system state which becomes erroneous

This chain can be applied recursively when the (distributed) system comprises several components:

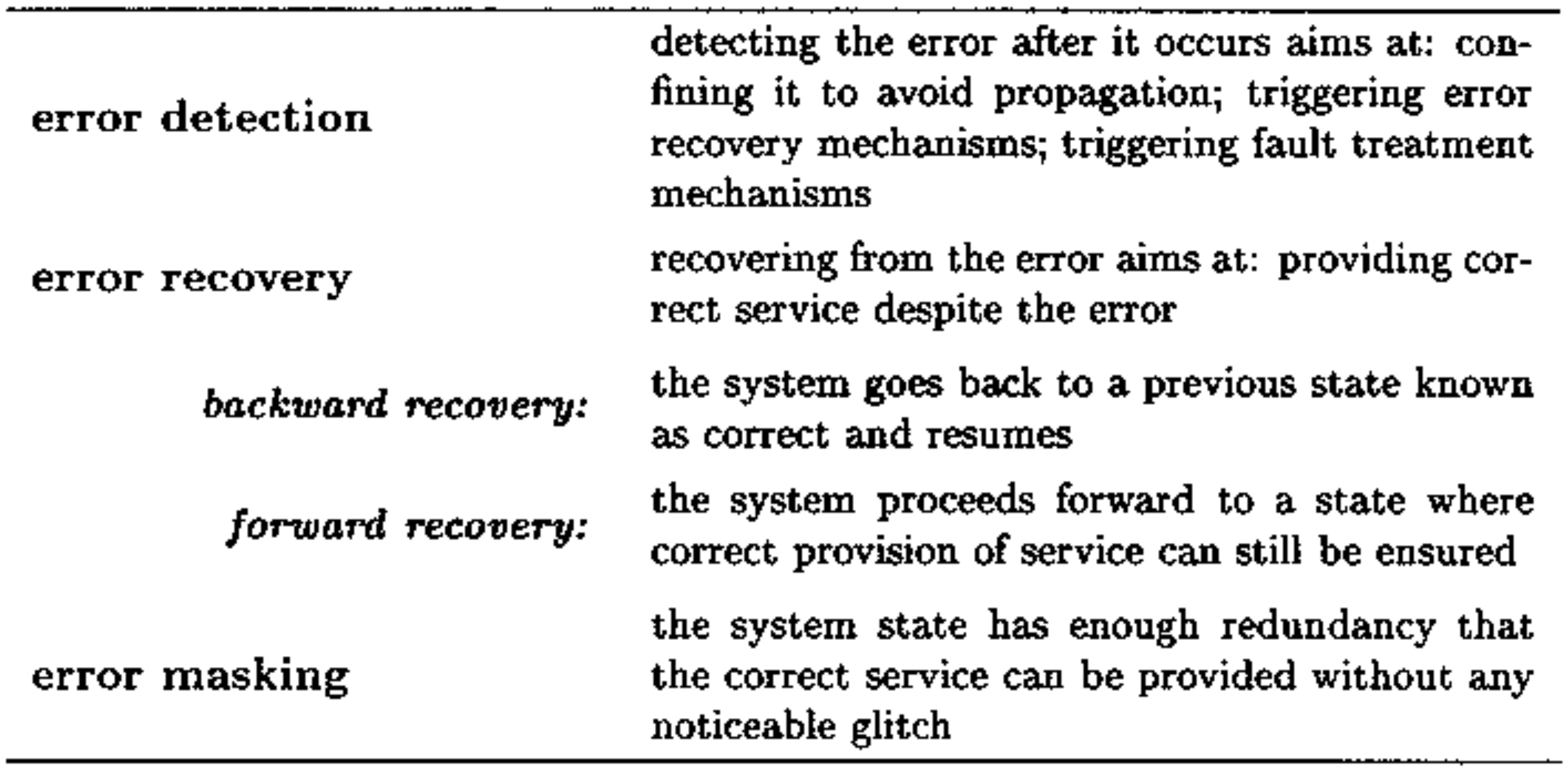


Faults are subdivided into classes depending on whether the are caused by hardware, software, or interaction.

Fault Tolerance (5)

Error Processing

Error Processing Techniques



error detection	detecting the error after it occurs aims at: confining it to avoid propagation; triggering error recovery mechanisms; triggering fault treatment mechanisms
error recovery	recovering from the error aims at: providing correct service despite the error
backward recovery:	the system goes back to a previous state known as correct and resumes
forward recovery:	the system proceeds forward to a state where correct provision of service can still be ensured
error masking	the system state has enough redundancy that the correct service can be provided without any noticeable glitch

Error Detection (1)

error detection = component failure detection

1. Local detection

“local” means that the two components are “close” enough to establish “perfect” communication

Model 1:

Assuming a special component called failure detector checking the target component

Examples:

- self-checking routines such as parity checks in memory, disks, or bus
- guardians checking the validity (plausibility) of the outputs produced by the observed component
- watchdogs checking the correct timing of the observed component

Model 2:

system diagnosis, i.e.all components are considered alike and mutually check each other

Assumption:

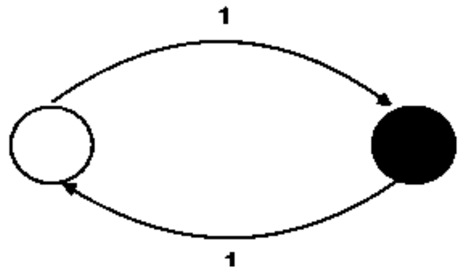
outcome of test reported by correct components can be trusted, faulty components not

---> no a priori knowledge of which components are faulty

---> diagnosis has to be performed by analyzing the reports of all components

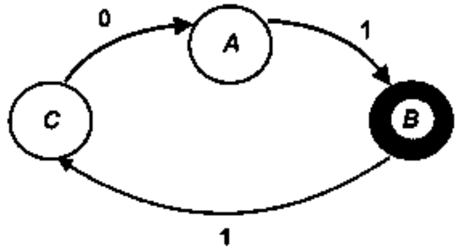
Error Detection (2)

Symmetric diagnosis (one faulty)



Result: It is impossible to assess which is the faulty component.

Diagnosis Ring (one faulty)



Result: The faulty component can be clearly identified using this criterion:

The incoming arc is marked "faulty" and the corresponding source node (detector) is marked "correct "

Theorem (Preparata):

If up to *f* components may be faulty, $n >= 2f+1$ components are needed for diagnosis and each component must be tested by at least *f* other components

Error Detection (3)

2. Distributed Detection

Assumption:

- communication channel between the observer and the target may be faulty (no longer perfect)
- nodes (processes) can only fail by crashing
- the system is synchronous, i.e. delays are bounded
- the network provides full connectivity (abstraction from the details of the network level)
- any process plays both roles (observer and target)

Properties defining the notion of consistency of distributed failure detection:

Strong Accuracy
no correct process is ever considered failed
Strong Completeness
a failure must be detected eventually by every correct process

A failure detector method meeting both properties is called *perfect.*
If perfect channels are available*, heartbeats* are a perfect failure detector.

Error Detection (4)

Impossibility of perfect failure detection if there is a lack of bounds

- on the number and type of faults of the communication channel
  - unbounded number of omission faults
  - network partitioning due to link failures
- for the timely behavior of system components (processes or links), i.e. asynchrony of the system

Consequence: weaker notions of consistency

Weak Accuracy

at least one correct process is never considered failed by all correct processes

Weak Completeness

a failure must be detected eventually by at least one correct process

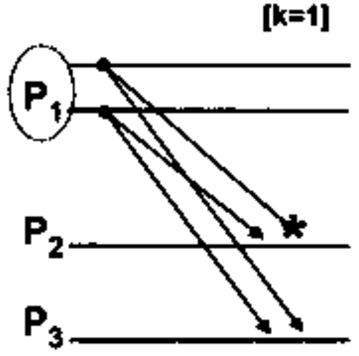
Depending on the given distributed system and its applications, different classes of failure detectors can be defined by combining weak and strong accuracy and completeness properties.

Fault-Tolerant Communication (1)

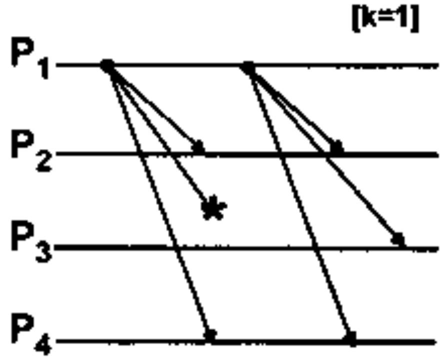
1. Network Omissions

Alternatives for error processing:

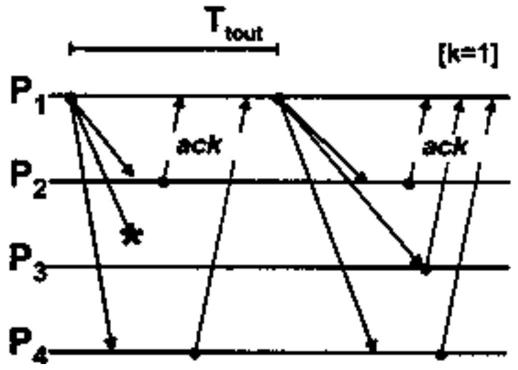
Spatial error masking



Temporal error masking



Detection/Recovery



Fault-Tolerant Communication (3)

2. Message corruption (Value faults)

a) during transmission

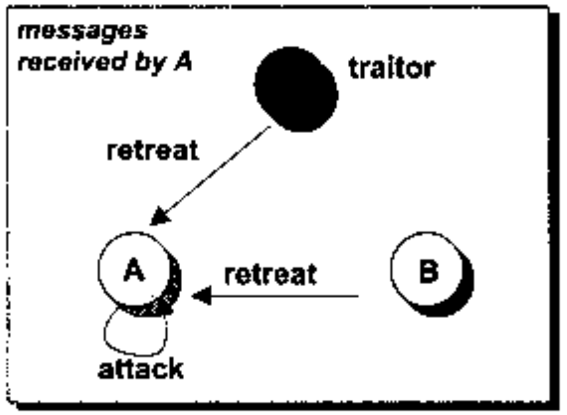
using checksums ---> corruptions are detected by the receiver ---> message is discarded and treated like an omission fault

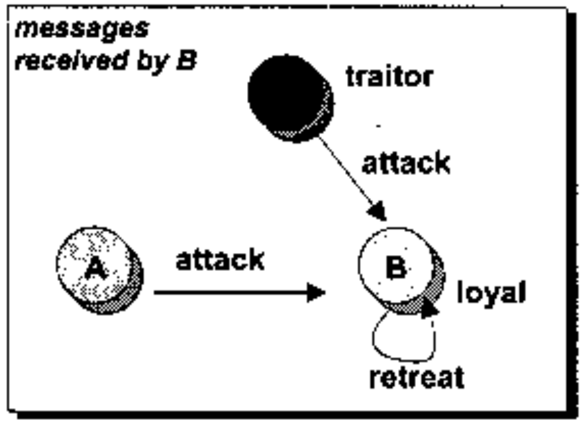
b) caused by a faulty sender that already produces an error in the information to be sent (semantic fault)

using space redundancy, e.g. TMR, N-Version-Programming

3. Arbitrary Faults (*Byzantine agreement)*

3 members (generals)





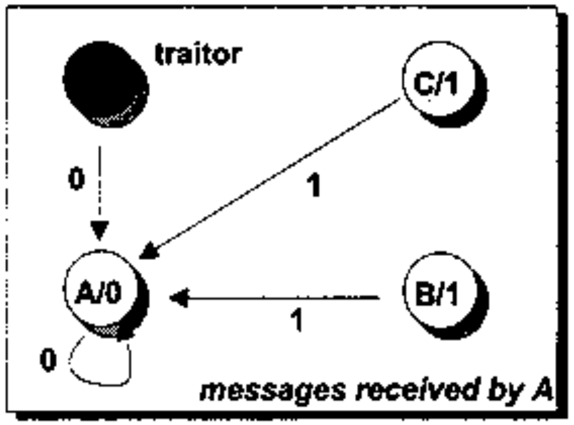
Applying majority voting: A (retreat) and B (attack) ---> conflicting decisions!!

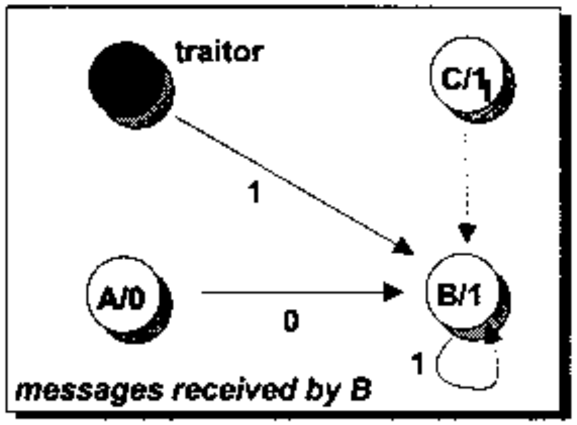
Fault-Tolerant Communication (4)

Additional rounds of message exchange would not help. In fact, it can be proven that it needs at least *3f+1* members to tolerate *f* Byzantine faults.

b) 4 members (generals)

First approach: one round of message exchange as before





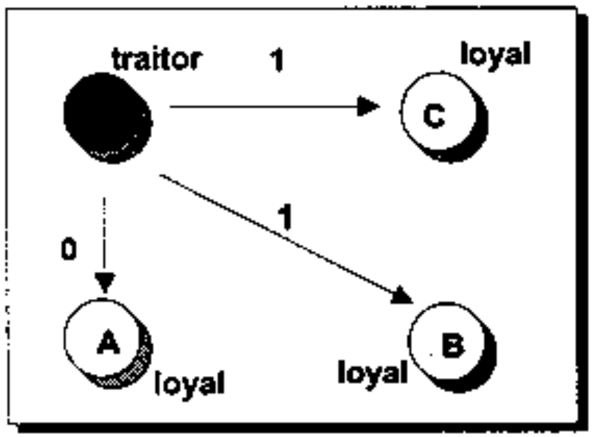
Again, by sending contradicting information, the “traitor” can force A and B to disagree.

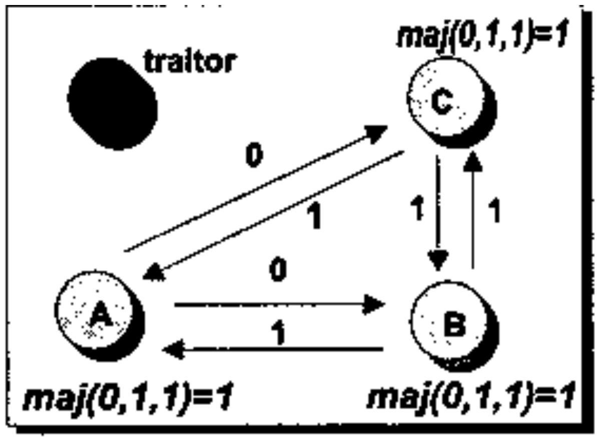
Fault-Tolerant Communication (5)

We now have enough redundancy in the system to mask the influence of the traitor with an additional round of messages.

Second approach:

1. First and second (b) round of message exchange showing the “traitors” messages (a) and the analysis on that done by the others



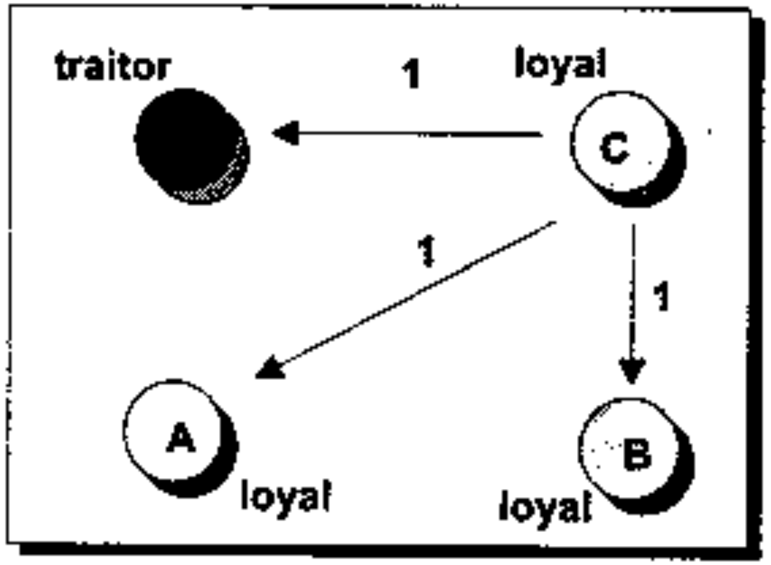


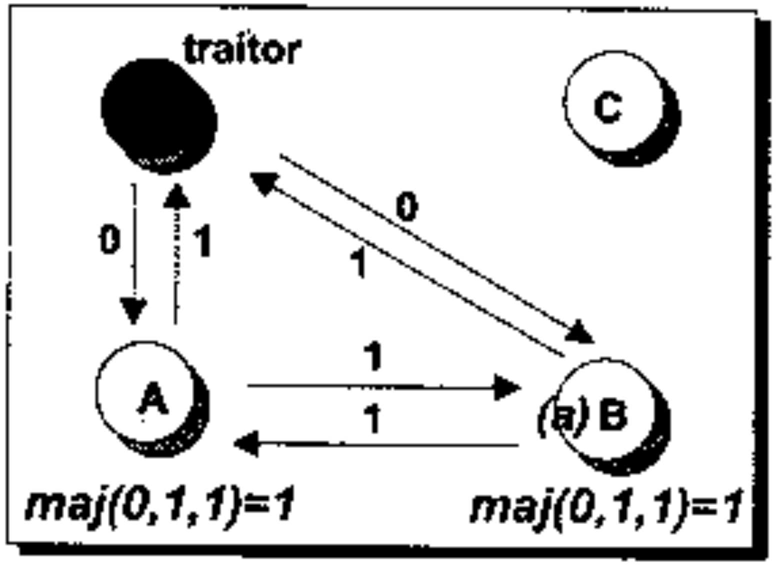
Now, the “loyal generals” do agree on the same value.

Fault-Tolerant Communication (6)

Second approach:

2. First and second (b) round of message exchange showing the “loyal” messages (a) and the analysis on that done by the others





Finally, all “loyal generals” do agree on what to do.