DISTRIBUTED SYSTEMS
Principles and Paradigms
Second Edition
ANDREW S. TANENBAUM
MAARTEN VAN STEEN

Bearbeitung von Matthias Wallnöfer

Fault Tolerance

Fault Tolerance Basic Concepts

- Being fault tolerant is strongly related to what are called dependable systems
- Dependability implies the following:
  1. **Availability**
  2. **Reliability**
  3. **Safety**
  4. **Maintainability**

Failure Models

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure *Receive omission* *Send omission*	A server fails to respond to incoming requests A server fails to receive incoming messages A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure *Value failure* *State transition failure*	A server's response is incorrect The value of the response is wrong The server deviates from the correct flow of control
Arbitrary failure	A server may produce arbitrary responses at arbitrary times

Figure 8-1. Different types of failures.

RPC Semantics in the Presence of Failures

Five different classes of failures that can occur in RPC systems:

1. The client is unable to locate the server.
2. The request message from the client to the server is lost.
3. The server crashes after receiving a request.
4. The reply message from the server to the client is lost.
5. The client crashes after sending a request.

Failure Masking by Redundancy

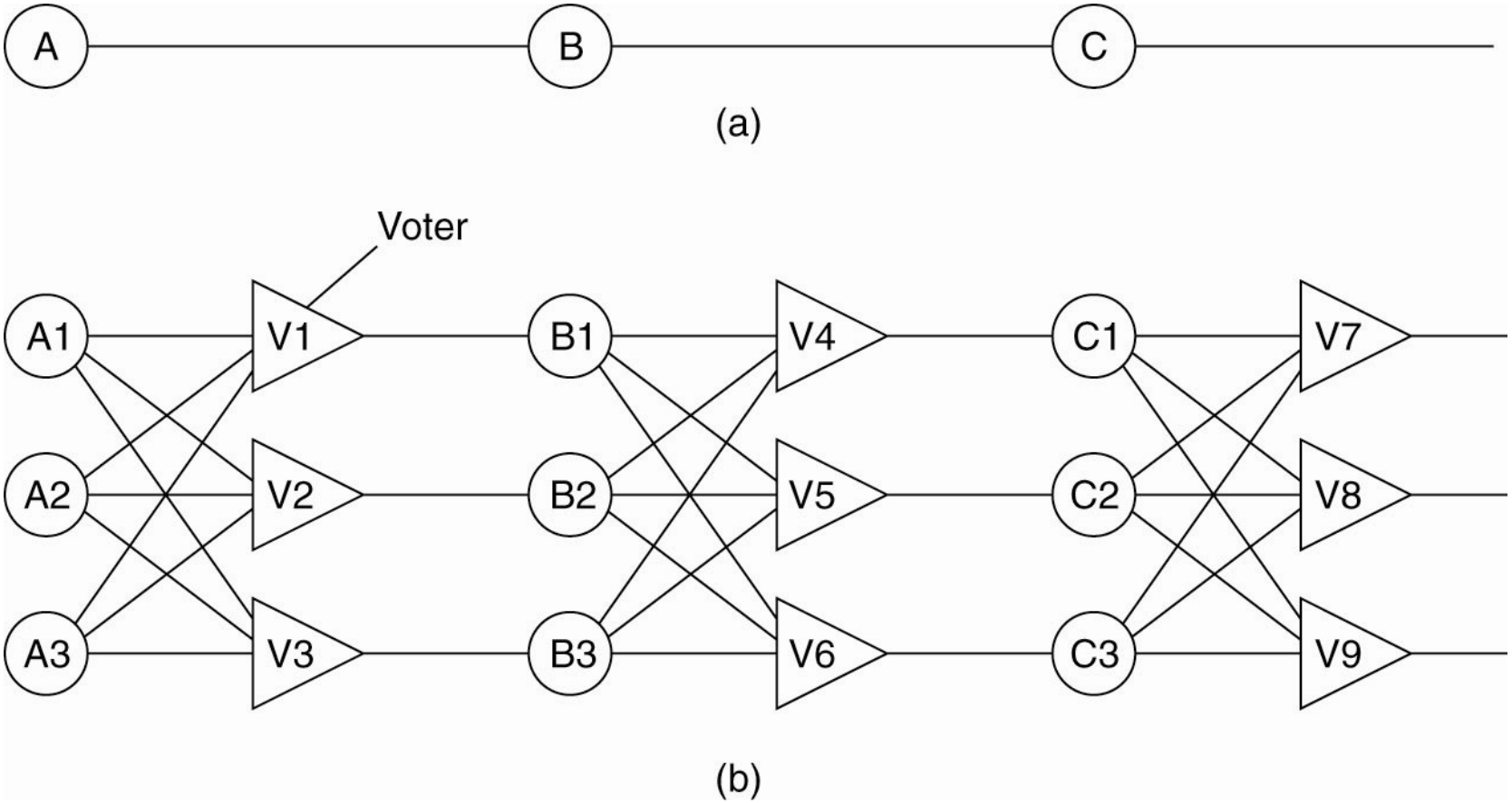
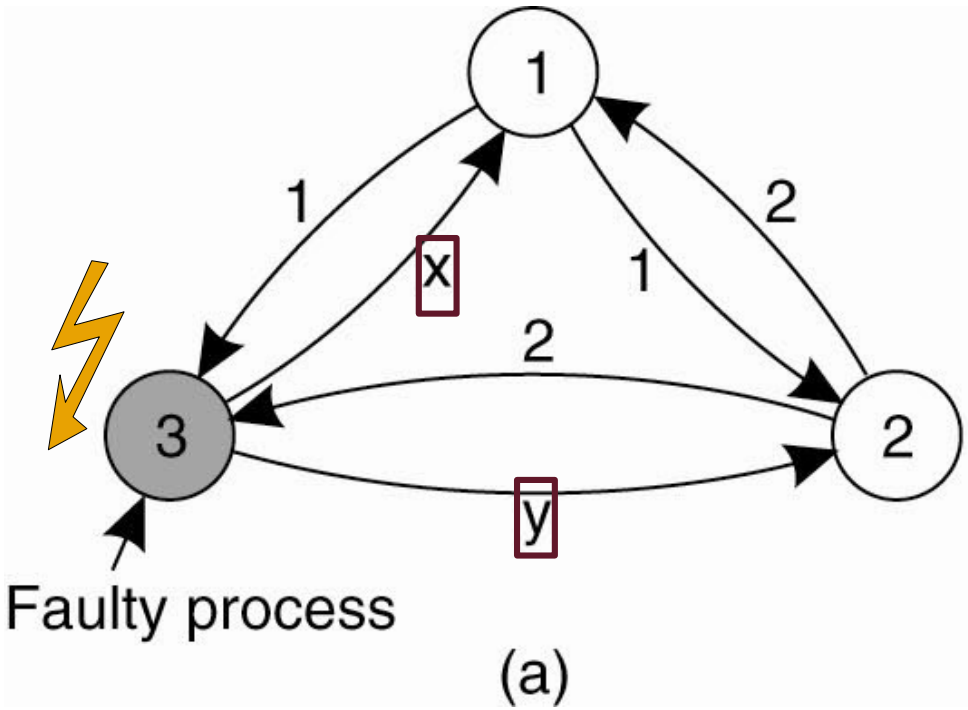


Figure 8-2. Triple modular redundancy.

Agreement in Faulty Systems (1)



1 Got(1, 2, x)
2 Got(1, 2, y)
3 Got(1, 2, 3)

???

1 Got	2 Got
(1, 2, y)	(1, 2, x)
(a, b, c)	(d, e, f)

No majority → no consensus

(c)

Figure 8-6. The Byzantine agreement problem for two nonfaulty and one faulty process. (b) The vectors that each process assembles based on (a). (c) The vectors that each process receives in step 3.

Agreement in Faulty Systems (2)

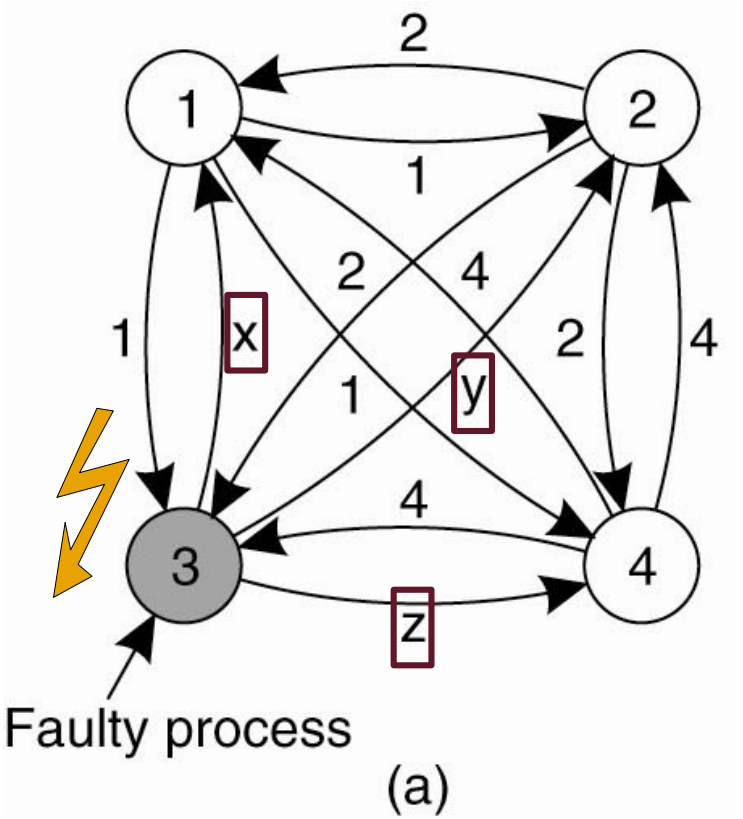
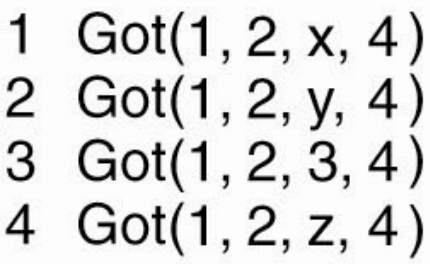
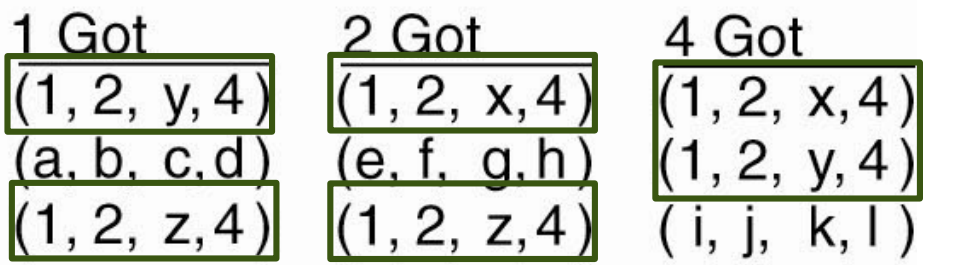


Figure 8-5. The Byzantine agreement problem for three nonfaulty and one faulty process. (a) Each process sends their value to the others.

Agreement in Faulty Systems (3)



(b)



Majorities (1, 2, _, 4) → **consensus achieved**

(c)

Figure 8-5. (b) The vectors that each process assembles based on (a). (c) The vectors that each process receives in step 3.

Two-Phase Commit (1)

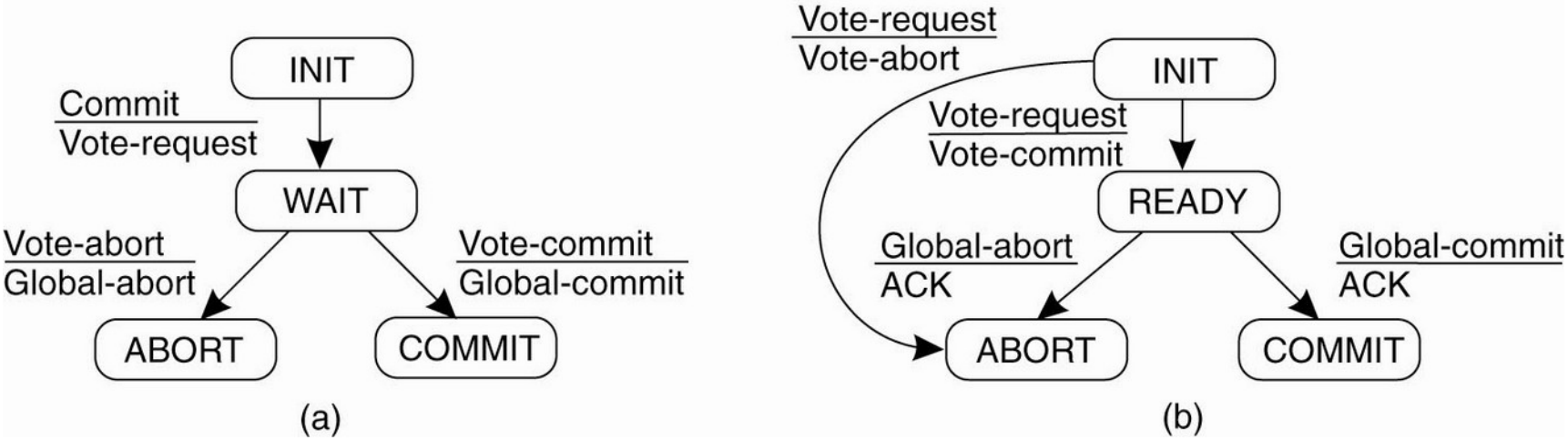


Figure 8-18. (a) The finite state machine for the coordinator in 2PC. (b) The finite state machine for a participant.

Two-Phase Commit (3)

Actions by coordinator:

```
write START_2PC to local log;
multicast VOTE_REQUEST to all participants;
while not all votes have been collected {
        wait for any incoming vote;
        if timeout {
                write GLOBAL_ABORT to local log;
                multicast GLOBAL_ABORT to all participants;
                exit;
        }
        record vote;
}
        . . .
```

Figure 8-20. Outline of the steps taken by the coordinator in a two-phase commit protocol.

Two-Phase Commit (4)

```
. . .

if all participants sent VOTE_COMMIT and coordinator votes COMMIT {
        write GLOBAL_COMMIT to local log;
        multicast GLOBAL_COMMIT to all participants;
} else {
        write GLOBAL_ABORT to local log;
        multicast GLOBAL_ABORT to all participants;
}
```

Figure 8-20. Outline of the steps taken by the coordinator in a two-phase commit protocol.

Recovery – Stable Storage

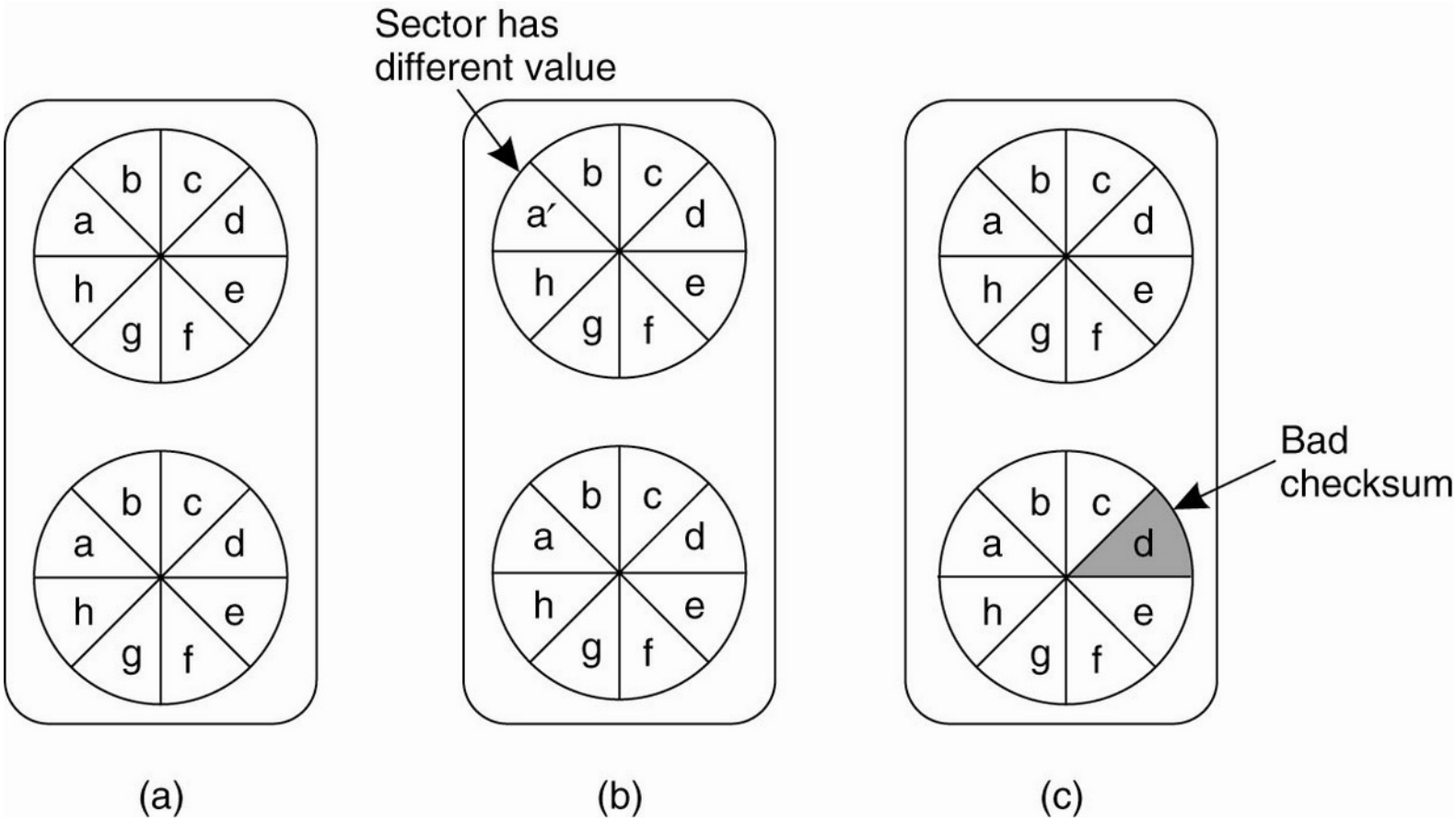


Figure 8-23. (a) Stable storage.
(b) Crash after drive 1 is updated. (c) Bad spot.