Fault Tolerance

Chapter 7
Basic Concepts

Dependability Includes

- Availability
- Reliability
- Safety
- Maintainability
# Failure Models

Different types of failures.

<table>
<thead>
<tr>
<th>Type of failure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crash failure</td>
<td>A server halts, but is working correctly until it halts</td>
</tr>
<tr>
<td>Omission failure</td>
<td>A server fails to respond to incoming requests</td>
</tr>
<tr>
<td>Receive omission</td>
<td>A server fails to receive incoming messages</td>
</tr>
<tr>
<td>Send omission</td>
<td>A server fails to send messages</td>
</tr>
<tr>
<td>Timing failure</td>
<td>A server's response lies outside the specified time interval holdal</td>
</tr>
<tr>
<td>Response failure</td>
<td>The server's response is incorrect</td>
</tr>
<tr>
<td>Value failure</td>
<td>The value of the response is wrong</td>
</tr>
<tr>
<td>State transition failure</td>
<td>The server deviates from the correct flow of control</td>
</tr>
<tr>
<td>Arbitrary failure</td>
<td>A server may produce arbitrary responses at arbitrary times</td>
</tr>
</tbody>
</table>
Failure Masking by Redundancy

(a)

(b)

Voter

Triple modular redundancy.
Flat Groups versus Hierarchical Groups

a) Communication in a flat group.
b) Communication in a simple hierarchical group
Agreement in Faulty Systems (1)

The Byzantine generals problem for 3 loyal generals and 1 traitor.

a) The generals announce their troop strengths (in units of 1 kilosoldiers).

b) The vectors that each general assembles based on (a)

c) The vectors that each general receives in step 3.
Agreement in Faulty Systems (2)

The same as in previous slide, except now with 2 loyal generals and one traitor.
Lost Request Messages
Server Crashes (1)

A server in client-server communication
a) Normal case
b) Crash after execution
c) Crash before execution
Server Crashes (2)

Different combinations of client and server strategies in the presence of server crashes.

<table>
<thead>
<tr>
<th>Reissue strategy</th>
<th>MPC</th>
<th>MC(P)</th>
<th>C(MP)</th>
<th>Strategy M -&gt; P</th>
<th>MPC</th>
<th>PC(M)</th>
<th>C(PM)</th>
<th>Strategy P -&gt; M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always</td>
<td>DUP</td>
<td>OK</td>
<td>OK</td>
<td></td>
<td>DUP</td>
<td>DUP</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>Never</td>
<td>OK</td>
<td>ZERO</td>
<td>ZERO</td>
<td></td>
<td>OK</td>
<td>OK</td>
<td>ZERO</td>
<td></td>
</tr>
<tr>
<td>Only when ACKed</td>
<td>DUP</td>
<td>OK</td>
<td>ZERO</td>
<td></td>
<td>DUP</td>
<td>OK</td>
<td>ZERO</td>
<td></td>
</tr>
<tr>
<td>Only when not ACKed</td>
<td>OK</td>
<td>ZERO</td>
<td>OK</td>
<td></td>
<td>OK</td>
<td>DUP</td>
<td>OK</td>
<td></td>
</tr>
</tbody>
</table>
Basic Reliable-Multicasting Schemes

A simple solution to reliable multicasting when all receivers are known and are assumed not to fail

a) Message transmission
b) Reporting feedback
Several receivers have scheduled a request for retransmission, but the first retransmission request leads to the suppression of others.
Hierarchical Feedback Control

The essence of hierarchical reliable multicasting.

a) Each local coordinator forwards the message to its children.
b) A local coordinator handles retransmission requests.
Virtual Synchrony (1)

The logical organization of a distributed system to distinguish between message receipt and message delivery.

Message comes in from the network → Local OS → Communication layer → Application.

Message is delivered to application.

Network
Virtual Synchrony (2)

The principle of virtual synchronous multicast.
## Message Ordering (1)

Three communicating processes in the same group. The ordering of events per process is shown along the vertical axis.

<table>
<thead>
<tr>
<th>Process P1</th>
<th>Process P2</th>
<th>Process P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>sends m1</td>
<td>receives m1</td>
<td>receives m2</td>
</tr>
<tr>
<td>sends m2</td>
<td>receives m2</td>
<td>receives m1</td>
</tr>
</tbody>
</table>
Message Ordering (2)

<table>
<thead>
<tr>
<th>Process P1</th>
<th>Process P2</th>
<th>Process P3</th>
<th>Process P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>sends m1</td>
<td>receives m1</td>
<td>receives m3</td>
<td>sends m3</td>
</tr>
<tr>
<td>sends m2</td>
<td>receives m3</td>
<td>receives m1</td>
<td>sends m4</td>
</tr>
<tr>
<td></td>
<td>receives m2</td>
<td>receives m2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>receives m4</td>
<td>receives m4</td>
<td></td>
</tr>
</tbody>
</table>

Four processes in the same group with two different senders, and a possible delivery order of messages under FIFO-ordered multicasting.
Implementing Virtual Synchrony (1)

<table>
<thead>
<tr>
<th>Multicast</th>
<th>Basic Message Ordering</th>
<th>Total-ordered Delivery?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable multicast</td>
<td>None</td>
<td>No</td>
</tr>
<tr>
<td>FIFO multicast</td>
<td>FIFO-ordered delivery</td>
<td>No</td>
</tr>
<tr>
<td>Causal multicast</td>
<td>Causal-ordered delivery</td>
<td>No</td>
</tr>
<tr>
<td>Atomic multicast</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td>FIFO atomic multicast</td>
<td>FIFO-ordered delivery</td>
<td>Yes</td>
</tr>
<tr>
<td>Causal atomic multicast</td>
<td>Causal-ordered delivery</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Six different versions of virtually synchronous reliable multicasting.
Implementing Virtual Synchrony (2)

a) Process 4 notices that process 7 has crashed, sends a view change

b) Process 6 sends out all its unstable messages, followed by a flush message

c) Process 6 installs the new view when it has received a flush message from everyone else
Two-Phase Commit (1)

The finite state machine for the coordinator in 2PC.

The finite state machine for a participant.
Two-Phase Commit (2)

<table>
<thead>
<tr>
<th>State of Q</th>
<th>Action by P</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMIT</td>
<td>Make transition to COMMIT</td>
</tr>
<tr>
<td>ABORT</td>
<td>Make transition to ABORT</td>
</tr>
<tr>
<td>INIT</td>
<td>Make transition to ABORT</td>
</tr>
<tr>
<td>READY</td>
<td>Contact another participant</td>
</tr>
</tbody>
</table>

Actions taken by a participant $P$ when residing in state \textit{READY} and having contacted another participant $Q$. 
actions by coordinator:

while 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 {
        while GLOBAL_ABORT to local log;
        multicast GLOBAL_ABORT to all participants;
        exit;
    }
    record vote;
}
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;
}

Outline of the steps taken by the coordinator in a two phase commit protocol
Two-Phase Commit (4)

actions by participant:
write INIT to local log;
wait for VOTE_REQUEST from coordinator;
if timeout {
   write VOTE_ABORT to local log;
   exit;
}
if participant votes COMMIT {
   write VOTE_COMMIT to local log;
   send VOTE_COMMIT to coordinator;
   wait for DECISION from coordinator;
   if timeout {
      multicast DECISION_REQUEST to other participants;
      wait until DECISION is received; /* remain blocked */
      write DECISION to local log;
   }
   if DECISION == GLOBAL_COMMIT
      write GLOBAL_COMMIT to local log;
   else if DECISION == GLOBAL_ABORT
      write GLOBAL_ABORT to local log;
} else {
   write VOTE_ABORT to local log;
   send VOTE_ABORT to coordinator;
}
actions for handling decision requests: /* executed by separate thread */

while true {
    wait until any incoming DECISION_REQUEST is received; /* remain blocked */
    read most recently recorded STATE from the local log;
    if STATE == GLOBAL_COMMIT
        send GLOBAL_COMMIT to requesting participant;
    else if STATE == INIT or STATE == GLOBAL_ABORT
        send GLOBAL_ABORT to requesting participant;
    else
        skip; /* participant remains blocked */
Three-Phase Commit

(a) Finite state machine for the coordinator in 3PC
(b) Finite state machine for a participant
Recovery Stable Storage

a) Stable Storage
b) Crash after drive 1 is updated
c) Bad spot

Sector has different value

Bad checksum
Checkpointing

A recovery line.

Initial state

Recovery line

Checkpoint

Failure

Time

Message sent from P2 to P1

Inconsistent cut

A recovery line.
Independent Checkpointing

The domino effect.
Incorrect replay of messages after recovery, leading to an orphan process.