CSci 5105

Introduction to Distributed Systems

FT cont’d
Last Time

- Fault tolerance
- Reliable multicast
Today

- Fault tolerance
- Chapter 8 TVS
- Midterm review
Two-Phase Commit (2PC)

- General protocol to implement reliable multicast and forms of consensus (e.g. group membership)

- Coordinator sends message and everyone either acts on message or not

- Typical action: commit a transaction
Two-Phase Commit (2PC)

• Multi-step (with coordinator)
  – Vote-request (coordinator)
  – Vote-commit or vote-abort (participant)
  – Global-commit or global-abort (coordinator)

• Impressions?
Two-Phase Commit (2PC)

- Distributed commit - all or none
- Starts when someone wants to commit a value and asks coordinator if it is ok
What about failure?

• Participant?
  – Before it votes
  – After it votes

• If it voted and crashed
  – When it recovers find out the outcome and take the action
What about failure?

- Coordinator failure
- Node P in READY state and times out
- Asks node Q

<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>

Safe to abort

Must block if everyone in READY state
2PC Failure/Recovery

- Nodes fail and may recover
- Use logging

**Actions by coordinator:**

```plaintext
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;
```
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;
}
2PC: Participant recovery

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;
}
2PC: Participant recovery (cont’d)

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 */
}

(b)

• Used to help other participants
3PC

- **2PC** is very expensive
- **Also** blocking after a failed node recovers to make a decision if coordinator crashes
- Add one more round: PREPARE-COMMIT
- Look at 3PC
Three-Phase Commit

If coordinator fails .... P contacts all other
if any in init or ready -> ?
if all in precommit => ?
What is a Byzantine Failure?

Three primary differences from Fail-Stop

1) Component can produce arbitrary output
   • Fail-stop: produces correct output or none

2) Cannot always detect output is faulty
   • Fail-stop: can always detect that component has stopped

3) Components may work together maliciously
   • No collusion across components
Simple Case

• Non-faulty decision maker or coordinator takes majority vote

• Suppose coordinator can fail?
  – 2PC very expensive

• Can we achieve agreement or consensus in the presence of arbitrary failures
  – ... without a non-faulty coordinator
Agreement in Faulty Systems

Need a multiple round solution
Every “good node to reach agreement on each node’s value
Agreement in Faulty Systems

1. Got(1, 2, x, 4)
2. Got(1, 2, y, 4)
3. Got(1, 2, 3, 4)
4. Got(1, 2, z, 4)

\[ \begin{array}{cccc}
1 \text{ Got} & 1 \text{ Got} & 2 \text{ Got} & 4 \text{ Got} \\
\frac{1, 2, y, 4}{(1, 2, y, 4)} & \frac{(a, b, c, d)}{(1, 2, z, 4)} & \frac{(e, f, g, h)}{(1, 2, z, 4)} & \frac{(1, 2, x, 4)}{(1, 2, y, 4)} \\
\end{array} \]

(b) (c)

Individual majority yields SAME answer in all non-faulty processors
Agreement in Faulty Systems

K failures, 2K+1 not sufficient!
General Impossibility Result

• Byzantine general’s problem

• No solution with fewer than $3m+1$ generals can cope with $m$ traitors
General Impossibility Result

• Byzantine general’s problem

• Commanding general: attack or retreat
• If Commander is loyal, must follow his decision
• All loyal generals must retreat or attack
• AVOID: some attacking, some retreating!
No agreement possible
Solution

• No solution with fewer than $3m+1$ total generals can cope with $m$ traitors

• $M=1 \Rightarrow 4,$

• $M=2 \Rightarrow 7$

• ...

• Since faulty can fool non-faulty need a stronger majority

• Multi-round protocol
Midterm

• Comprehensive through today
• Open book, open notes
• Single page one-sided cheat sheet
• Homework questions are useful to look at

• Will have a mix of shorter thinking questions and “work” problems
• 5 longer work questions
Examples “work”

• Given a message timeline, determine all of the causality relationships.

• Given a conit and several replicas, compute the various consistency metrics.

• For a set of processes and operations, determine the set of sequential or causal execution sequences.
Examples “work”

• Compute clock drift and re-synchronization using protocols A, B, C

• What happens in 2PC or 3PC when the states are {…} and a process fails?

• Given N processes under a quorum protocols; analyze behavior for certain settings
Examples

• You will be asked a question about mutual exclusion (taped lecture)