EECS591 FINAL REVIEW
What a long strange trip it’s been...
Final exam: April 27, 10:30am-12:30pm
Location: DOW 1014
PART ONE: FUNDAMENTALS
TWO GENERALS’ PROBLEM

Both generals must attack together or face defeat.

Communication is only by messengers sneaking through the valley.

Messengers may not make it through…
Question 1 (true or false)

a. $e \rightarrow d$

b. $a \rightarrow j$

c. $g \rightarrow b$
Lamport clocks

$p \rightarrow q \Rightarrow LC(p) < LC(q)$

the Clock condition
VECTOR CLOCKS

\[ VC(e_i)[i] = \text{number of events executed by process } i \text{ (including } e_i) \]

\[ VC(e_i)[j] = \text{number of events executed by process } j \text{ that causally precede } e_i \]

\[ p \rightarrow q \iff LC(p) < LC(q) \]

Strong clock condition
**Vector clocks**

\[ VC(e_i)[i] = \text{number of events executed by process } i \text{ (including } e_i) \]

\[ VC(e_i)[j] = \text{number of events executed by process } j \text{ that causally precede } e_i \]

**Question 2:** what is the VC of:

a. event \( d \)

b. event \( g \)
Cristian's algorithm

\[ \text{slave} \quad P(t) \quad \text{master} \quad Q(t) \]

\[ \text{time=}? \quad 2D \quad \text{time=}T \]

\[ t \quad \text{min} + \alpha \quad 2d \quad \text{min} + \beta \quad t = x \]

\[ \alpha, \beta \geq 0 \]

\[ Q(x) = ? \]
2-Phase Commit

Coordinator $c$

1. sends VOTE-REQ to all participants

Participant $p_i$

2. sends $vote_i$ to Coordinator

   if $vote_i = \text{No}$ then
   
   $decision_i := \text{Abort}$
   
   halt

3. if (all votes are Yes) then

   $decision_c := \text{Commit}$

   send Commit to all

   else

   $decision_c := \text{Abort}$

   send Abort to all who voted Yes

   halt

4. if received Commit then

   $decision_i := \text{Commit}$

   else

   $decision_i := \text{Abort}$

   halt
3-Phase Commit

Coordinator $c$

1. sends VOTE-REQ to all participants

Participant $p_i$

2. sends $vote_i$ to Coordinator
   - if $vote_i = \text{No}$ then
     - $decision_i := \text{Abort}$
     - halt

3. if (all votes are Yes) then
   - send Precommit to all else
     - $decision_c := \text{Abort}$
     - send Abort to all who voted Yes
     - halt

4. if received Precommit then
   - send Ack

5. collect Ack from all participants
   - When all Ack’s have been received: $decision_c := \text{Commit}$
   - send Commit to all

6. When $p_i$ receives Commit, sets $decision_i := \text{Commit}$ and halts
**3PC: Which states are compatible?**

**Question 3:**

<table>
<thead>
<tr>
<th></th>
<th>Aborted</th>
<th>Uncertain</th>
<th>Committable</th>
<th>Committed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aborted</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uncertain</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Committable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Committed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A hierarchy of failure models
Ingredients: a server

1. Make server deterministic (state machine)
2. Replicate server
3. Ensure that all replicas go through the same sequence of state transitions
4. Vote on replica outputs
A PRIMARY-BACKUP PROTOCOL

\((f = 1)\)
Chain replication

Tail can respond immediately, without waiting for the new update

Head \( f + 1 \) replicas Tail
# Consensus

<table>
<thead>
<tr>
<th>Validity</th>
<th>If all processes that propose a value propose ( v ), then all correct processes eventually decide ( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement</td>
<td>If a correct process decides ( v ), then all correct processes eventually decide ( v )</td>
</tr>
<tr>
<td>Integrity</td>
<td>Every correct process decides at most one value, and if it decides ( v ), then some process must have proposed ( v )</td>
</tr>
<tr>
<td>Termination</td>
<td>Every correct process eventually decides some value</td>
</tr>
</tbody>
</table>
Our algorithm implementing consensus in a synchronous setting is correct! That is, it is both safe and live.
BAD NEWS

The FLP result:
There is no protocol that solves consensus in an asynchronous system where one process may crash

Fischer, Lynch, Paterson 1985
Abstract

The Paxos algorithm, when presented in plain English, is very simple.
PAXOS AT WORK
<table>
<thead>
<tr>
<th>Acceptors</th>
<th>Value</th>
<th>By leader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td></td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td></td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td></td>
<td>$y$</td>
<td>41</td>
</tr>
</tbody>
</table>

**Question 4:** What is the set of possible values that leader #50 can propose?
THE THREAT TO LIVENESS: DUELING PROPOSERS

Greetings, peasants! I am your fearless leader #1! Grant me your blessing!

Greetings, peasants! I am your fearless leader #2! Grant me your blessing!

Greetings, peasants! I am your fearless leader #3! Grant me your blessing!

Greetings, peasants! I am your fearless leader #4! Grant me your blessing!

Greetings, peasants! I am your fearless leader #5! Grant me your blessing!

Greetings, peasants! I am your fearless leader #6! Grant me your blessing!

Greetings, peasants! I am your fearless leader #7! Grant me your blessing!

Greetings, peasants! I am your fearless leader #8! Grant me your blessing!

. . .
Paxos/SMR in real life

Proposers, acceptors and learners are all collocated on $2f + 1$ replicas.
PBFT

Primary

Replica 1

Replica 2

Replica 3

Pre-prepare phase | Prepare phase | Commit phase | Reply phase
First execute... (multithreaded and without agreeing on the order)

...then verify (that replicas agree on the outcome)
PART TWO: SYSTEMS
Systems on Replication and Fault Tolerance

Co-designing Paxos with the network
SpecPaxos
NOPaxos

Optimizing BFT
Zyzzyva
XFT

Real-world coordination services
Chubby
ZooKeeper

Replication for the datacenter
CORFU
Tango

Leveraging storage replication
Locks with intent

Replication for WANs
Mencius

Fast and accurate fault detection
Falcon

Co-designing 2PC and Paxos
TAPIR
LARGE SCALE STORAGE SYSTEMS

Various consistency levels

Eventual consistency
- Bayou
- Dynamo

Causal+
- COPS
- Eiger

Fork-Join-Causal
- Depot

Red-Blue
- Gemini

Fast crash recovery
- RAMCloud

Industrial systems

Yahoo!: PNUTS (relational)

Facebook: Cassandra (key-value)

Google:
- Megastore
- Bigtable
- GFS

entity group replicas (across datacenters)

master

Chubby

tablet servers

master

chunkservers

Spanner

participant leader

participant leader

replica

tablet

replica

tablet

Colossus

participant leader

participant leader

replica

tablet

replica

tablet
LARGE SCALE COMPUTATION SYSTEMS

MapReduce
- input: DOG CAT RAT, CAR CAR RAT, DOG CAR CAT
- split: DOG, CAT, RAT
- map: CAR, RAT
- shuffle: CAR, RAT
- reduce: CAR, RAT
- final result: DOG, CAR, CAT

Dryad
- No fixed graph - more expressive
- Coarse-grained transformations
- Much faster

Spark
FORMAL VERIFICATION OF DISTRIBUTED SYSTEMS
IronFleet

DECENTRALIZED DIGITAL CURRENCY
Bitcoin

SCALABLE LOOKUP
Chord
Presentations

First, you should always make a script for your presentation, before you start making slides. This helps you organize your thoughts and present them clearly to your audience. The script should be at the high level, a kind of summary of the presentation with about one or two sentences per slide. Also, you should avoid having lots of text on one slide, as this is guaranteed to put your audience to deep, dreamless slumber. Where most presentations fail is that their authors, convinced they are producing some kind of stand-alone document, put everything they want to say onto their slides, in great big chunky blocks of text. While speaking, your voice should not be a flat monotonic drone, but you should try to change inflection often, so as to avoid putting your audience to sleep. And, of course, you should never try to read aloud the text written in your slides. If you find yourself doing that during your practice talks, it means there’s something wrong with the presentation. Unless of course you are trying to make a point, as I am doing right now :)

Presentations (for real this time)

- Motivation, motivation, motivation!
- Keep it simple
  - Give the high-level intuition
- Avoid the “wall of text”
- Speak normally, with changes to your inflection
- Practice, practice, practice!
Questions?

• On EECS591
• On distributed systems
• On computer science
• On research
• On Life, the Universe and Everything…
Thank you for attending EECS591!

Remember to fill in the evaluation form