EECS591 CLASS REVIEW
What a long, strange trip it’s been...
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

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

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

Strong clock condition
**Vector clocks**

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

---

**Diagram:**

- Nodes: a, b, c, d, e, f, g, h, i, j
- Edges:
  - a to b
  - b to e
  - f to g
  - g to j
  - h to i
  - i to j

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

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

\[ \text{time} = \min + \alpha \]
\[ \text{time} = \min + \beta \]
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
     - $decision_i := \text{Abort}$
     - $halt$
   - else

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 $v_i$ to Coordinator

if $v_i = \text{No}$ then

$\text{decision}_i := \text{Abort}$

halt

3. if (all votes are Yes) then

send Precommit to all

else

$\text{decision}_c := \text{Abort}$

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:

$\text{decision}_c := \text{Commit}$

send Commit to all

6. When $p_i$ receives Commit, sets $\text{decision}_i := \text{Commit}$ and halts
A hierarchy of failure models

Byzantine

Omission failures

Commission failures

crash
**State Machine Replication**

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

EVALUATIONS
A primary-backup protocol

\[(f = 1)\]
Chain replication

Tail can respond immediately, without waiting for the new update.

Head \( f + 1 \) replicas

Tail
<table>
<thead>
<tr>
<th><strong>Validity</strong></th>
<th>If all processes that propose a value propose ( v ), then all correct processes eventually decide ( v )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agreement</strong></td>
<td>If a correct process decides ( v ), then all correct processes eventually decide ( v )</td>
</tr>
<tr>
<td><strong>Integrity</strong></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><strong>Termination</strong></td>
<td>Every correct process eventually decides some value</td>
</tr>
</tbody>
</table>
Good news

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

Proposer

IAmLeader YouAreLeader Decree

Acceptors

Learner

Accept
**ACCEPTOR STATES**  
*(as leader #50 comes to power)*

<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?
### EXAMPLES OF ACCEPTOR STATES
(as leader #50 comes to power)

<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>y</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>41</td>
</tr>
</tbody>
</table>

Question 5:
What is the set of possible values that leader #50 can propose?
Greetings, peasants! I am your fearless leader #1! Grant me your blessing!

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

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

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

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

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

Greetings, peasants! I am your fearless leader #6! 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)
THINGS I HOPE YOU WILL REMEMBER

1. Need causality? Don’t reinvent vector clocks!

2. No perfect clock sync; but we can get very close.

3. Fewer than 2f+1 replicas $\rightarrow$ you don’t tolerate asynchrony

4. Fewer than 3f+1 replicas $\rightarrow$ you don’t tolerate non-benign faults
4b. Be able to tell Lorenzo apart from his evil twin

1. Evil Lorenzo Speaks French
2. And was born in Corsica
3. Went to Dartmouth instead of Cornell
4. Rides a Ducati instead of a Moto Guzzi
5. Still listens opera, but doesn't care for Puccini
5. Evil Lorenzo thinks that 2f+1 is good enough

Bonjour!!!!!
2f+1 works for me!
Things I hope you will remember (cont.)

5. Always write to disk before sending a message

6. 2PC is blocking; and so is 3PC (just less frequently)

7. Be careful when messing with Paxos, or you’ll get it wrong :-(
Course evaluations due today

Research project report due December 14th
(note new date)
AdminISTRIVIA

SUBMIT COURSE EVALUATIONS

Research project report due December 14th
(note new date)

Course evaluations due today
How to structure a research paper

- Introduction
  - Most important part of the paper
- Related work
- Design
- Implementation
- Evaluation
- Conclusion
PART TWO: RESEARCH
SYSTEMS ON REPLICATION AND FAULT TOLERANCE

Paxos optimizations
FastPaxos
Flexible Paxos

Replication in the real world
ZooKeeper
CORFU

Others
Zyzzyva
Falcon
Mencius
IronFleet
LARGE SCALE STORAGE SYSTEMS

Eventual and causal consistency
Bayou
Dynamo
COPS

Google:

Megastore

entity group replicas (across datacenters)

Bigtable

tablet servers

GFS

chunkservers

Spanner

participant leader

other group's participant leader

other group's participant leader

Colossus

????
LARGE SCALE COMPUTATION SYSTEMS

MapReduce

input

DOG CAT RAT
CAR, CAR, RAT
DOG, CAR, CAT

split

DOG, CAT, RAT
CAR, CAR, RAT
DOG, CAR, CAT

map

DOG, I
CAT, I
RAT, 1

DOG, I
CAR, I
RAT, 1

DOG, I
CAR, I
CAT, 1

DOG, I
CAR, I
CAT, 1

DOG, I
CAR, I
RAT, 1

DOG, I
CAR, I
RAT, 1

DOG, I
CAR, I
RAT, 1

DOG, I
CAR, I
RAT, 1

shuffle

DOG, I
DOG, I

CAR, I
CAR, I
CAR, I

CAR, I
CAR, I
CAR, I

CAR, I
CAR, I
CAR, I

CAR, I
CAR, I
CAR, I

RAT, I
RAT, I

RAT, I
RAT, I

RAT, I
RAT, I

reduce

DOG, 2

CAR, 3

CAT, 2

RAT, 2

final result

DOG, 2
CAR, 3
CAT, 2
RAT, 2

Spark

• No fixed graph - more expressive
• Coarse-grained transformations
• Much faster
CRYPTOCURRENCIES

Bitcoin, Ethereum, Hyperledger Fabric, Algorand
Things I hope you will remember

1. Consistency-performance tradeoff

2. Read papers critically

3. Read papers often
PRESENTATIONS

- Motivation, motivation, motivation!
- Keep it simple
  - Give the high-level intuition
  - Don’t go too deep
- Avoid the “wall of text”
- Speak normally, with changes to your inflection
- Practice, practice, practice!
PRESENTATIONS (FINE-GRAINED)

- Your talk is a story, not a sequence of slides
- Look at the audience, not your laptop
- Use an outline, refer back to it frequently
  - reconnect back to your story
- Use examples to clarify your points
- Make sure everyone can see your text
Operational need for measurement

- When services become unavailable or slow, want to alert and reroute quickly.
- Need diagnostic capabilities to find the root cause of issues.
  - Comcast in Seattle is having trouble reaching my CDN. Are they able to reach other networks?
- Want to measure impact of changes on end-users.
  - Want to take a front-end offline for maintenance. What is the performance impact on that front-end's users?
Serviceability is Key

- All parts of the system can be updated, and all of which requires no-to-little, or no downtime, paths. Or VMM can be live migrated.
- MMU image is live, VMM layer, VMM is hypervisor
- VMM requires high uptime, and NVS storage
- VMM also allows the VMs to be managed under the MMU

Instead of having the protocol, VMM uses support transparent protocol between the VMM and VM.
Questions?

- On EECS591
- On distributed systems
- On computer science
- On research
- On Life, the Universe and Everything…
THANK YOU FOR ATTENDING EECS591!