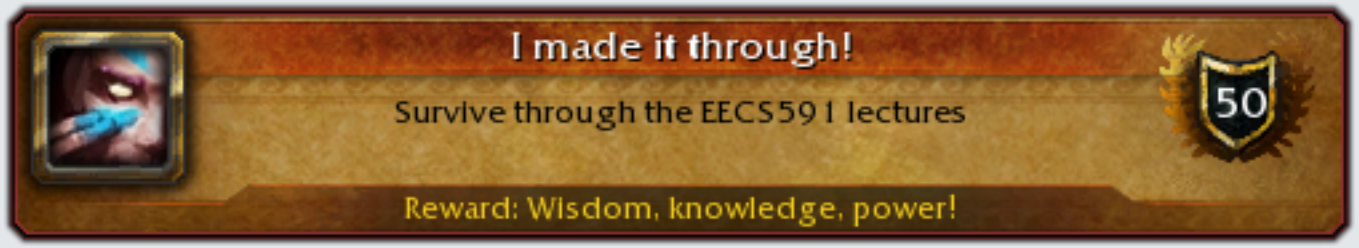
EECS591 CLASS REVIEW

What a long, strange trip it's been…



PART ONE: FUNDAMENTALS

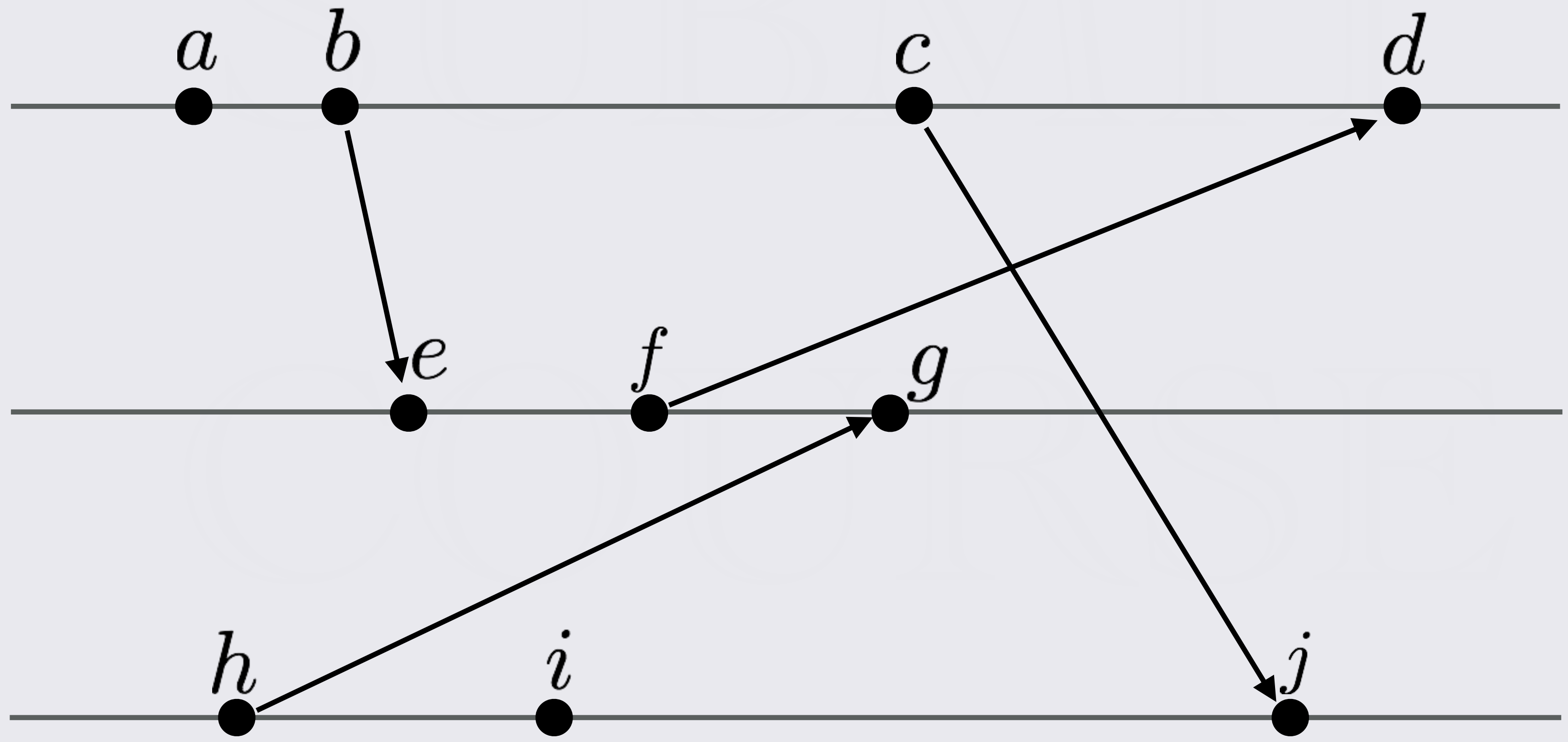
Two generals' problem

Both generals must attack together or face defeat

Communication is only by messengers sneaking through the valley

Messengers may not make it through…

ORDERING EVENTS WITHOUT PHYSICAL CLOCKS



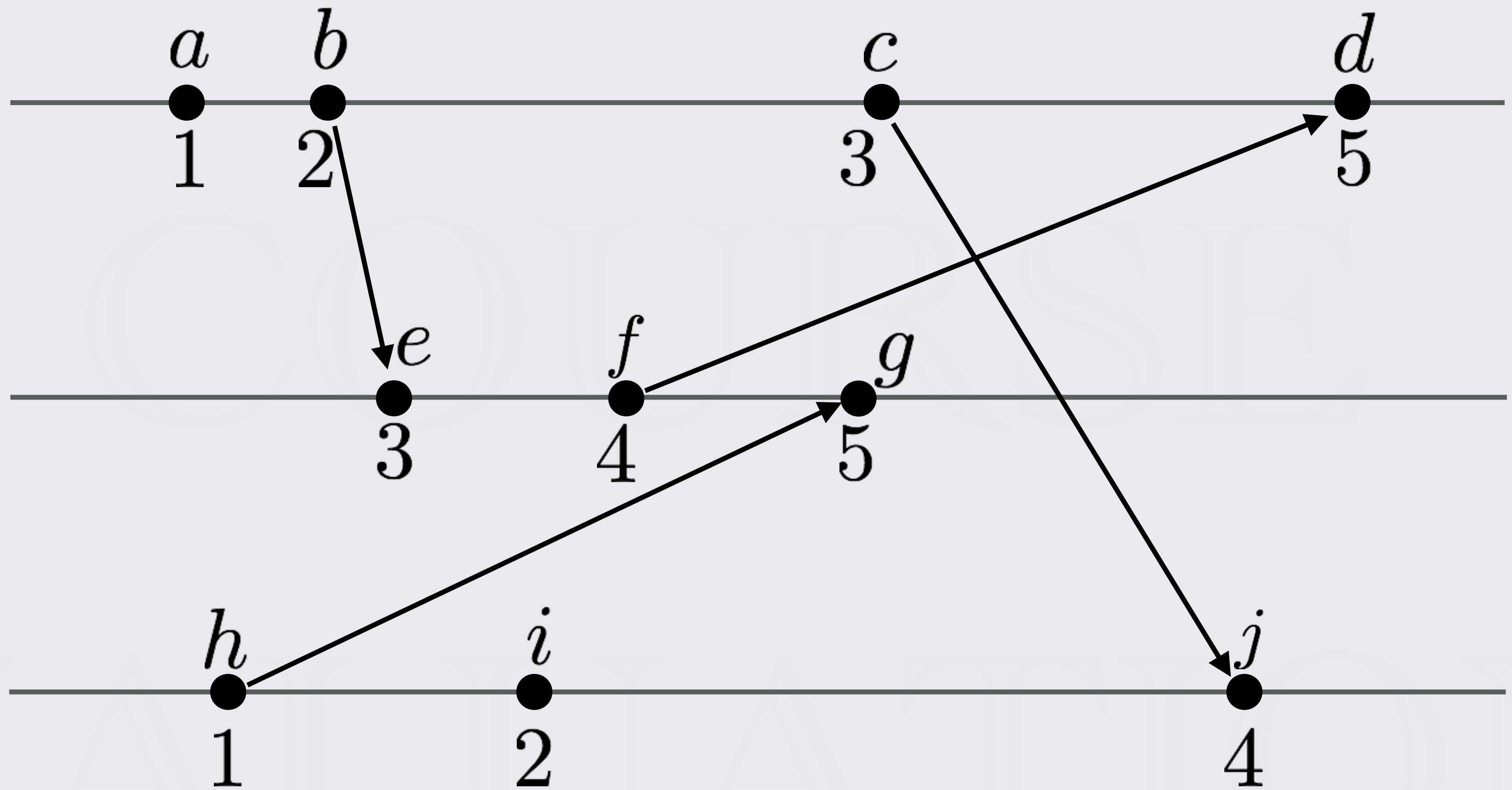
Question 1 (true or false)

a. $e \rightarrow d$

b. $a \rightarrow j$

c. $g \rightarrow b$

LAMPORT CLOCKS

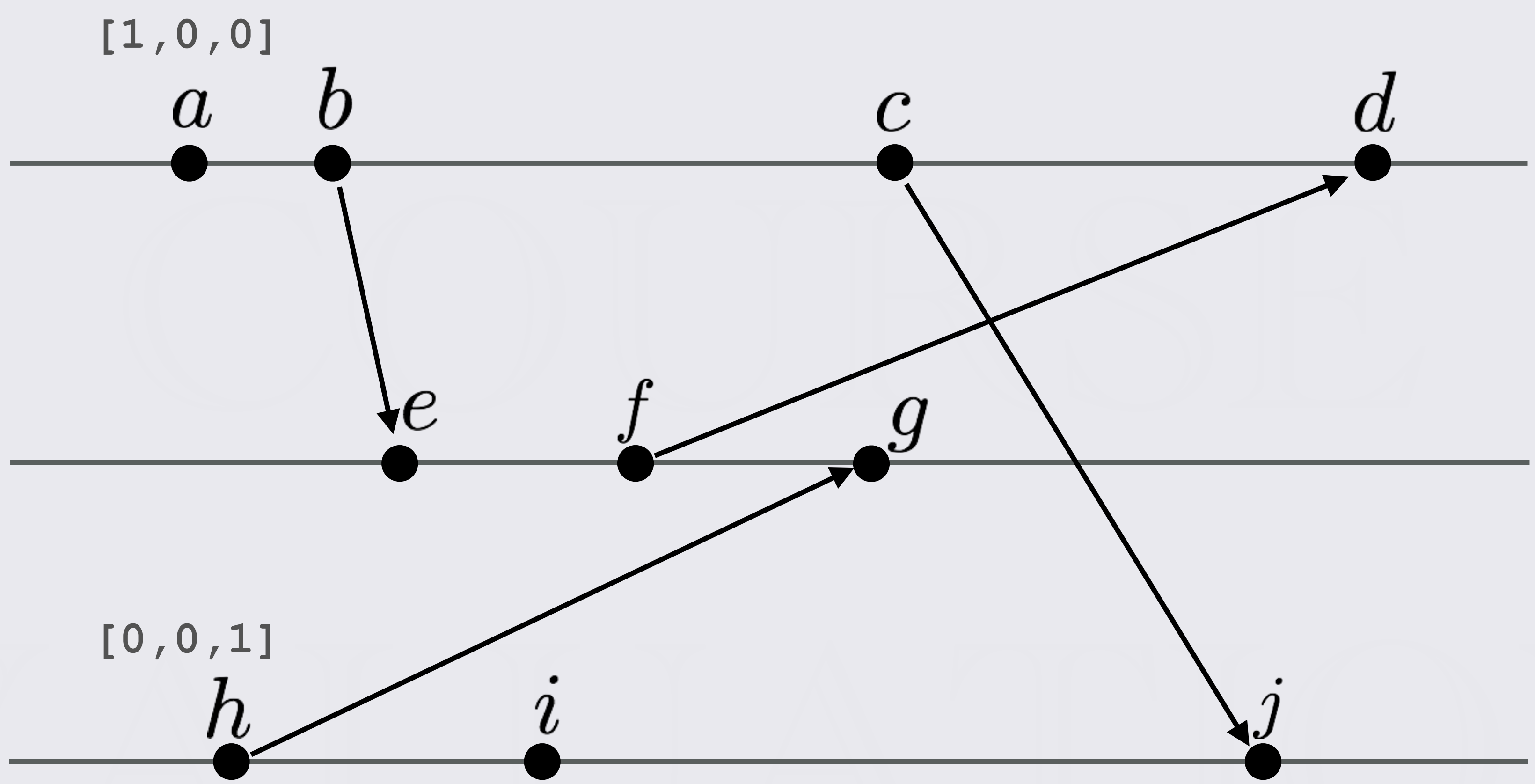


$$p \rightarrow q \Rightarrow LC(p) < LC(q)$$

the Clock condition

VECTOR CLOCKS

$VC(e_i)[j]$ = number of events executed by process j that causally precede e_i

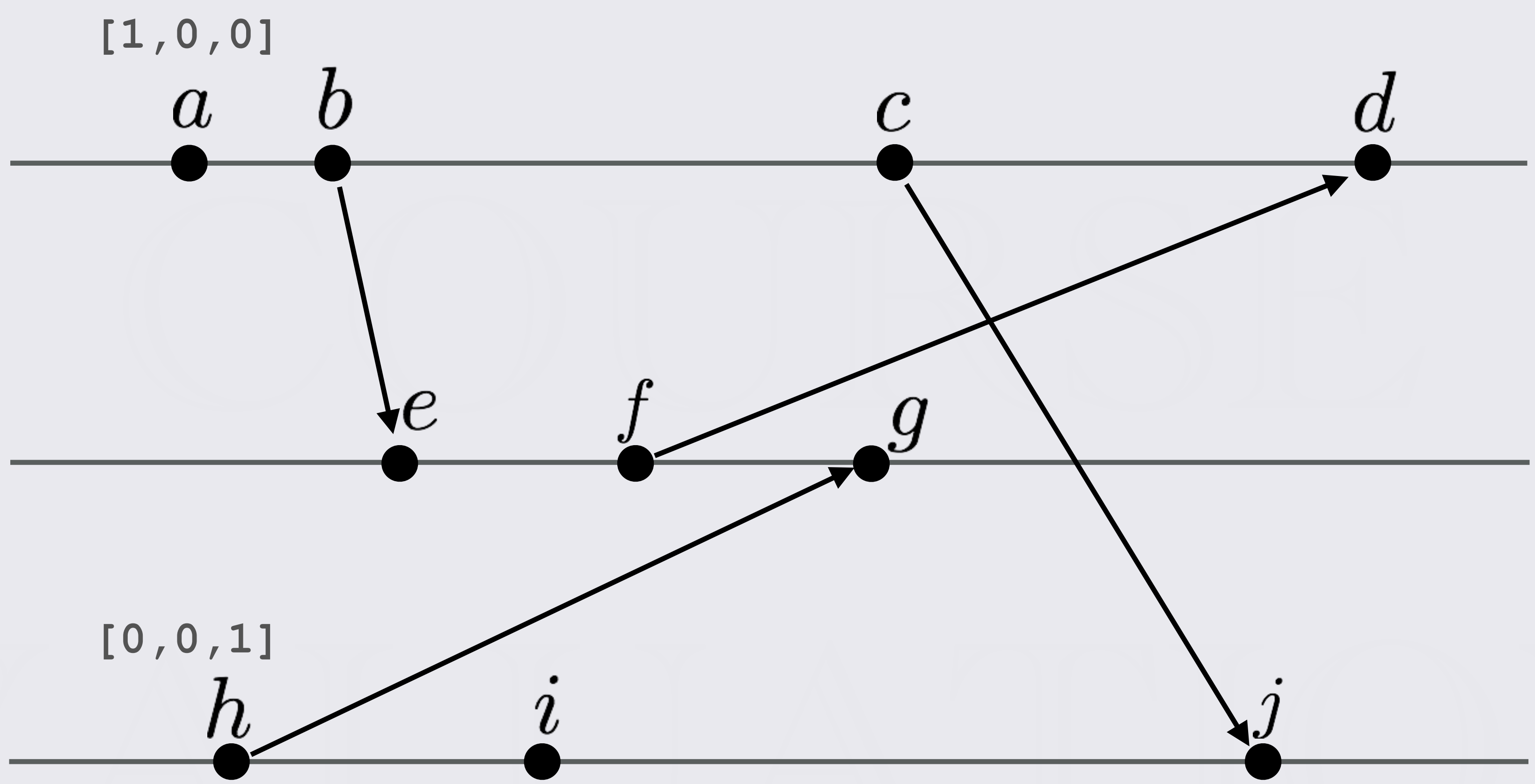


$$p \rightarrow q \Leftrightarrow LC(p) < LC(q)$$

Strong clock condition

VECTOR CLOCKS

$VC(e_i)[j]$ = number of events executed by process j that causally precede e_i

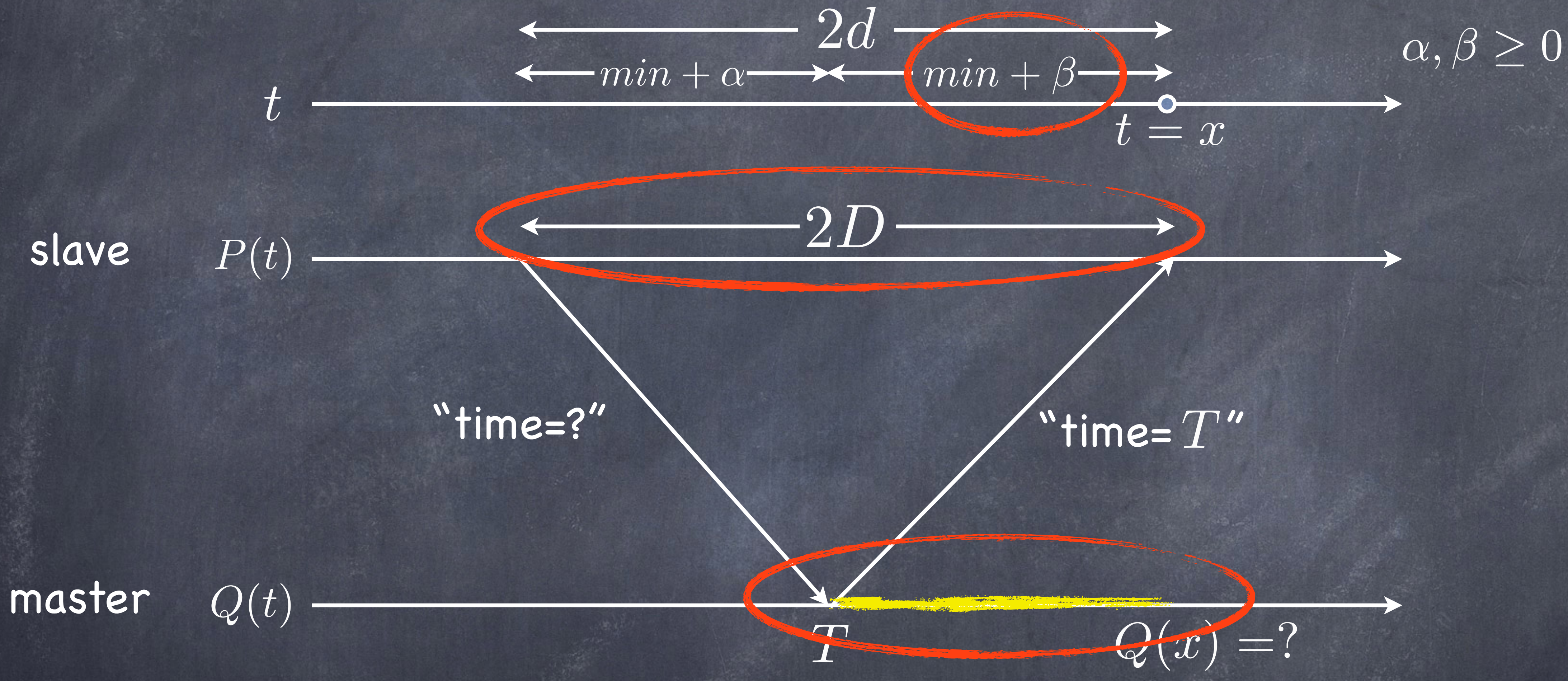


Question 2: what is the VC of:

a. event d

b. event g

Cristian's algorithm



2-Phase Commit

Coordinator c

Participant p_i

1. sends VOTE-REQ to all participants

2. sends $vote_i$ to Coordinator

   if $vote_i$ = **No** then
      $decision_i$:= **Abort**
   halt

3. if (all votes are **Yes**) then
      $decision_c$:= **Commit**
      send **Commit** to all
   else
      $decision_c$:= **Abort**
      send **Abort** to all who voted **Yes**
   halt

4. if received **Commit** then
      $decision_i$:= **Commit**
   else
      $decision_i$:= **Abort**
   halt

3-PHASE COMMIT

Coordinator c | **Participant p_i**

1. sends VOTE-REQ to all participants

2. sends $vote_i$ to Coordinator

   if $vote_i$= **No** then
   $decision_i$:= **Abort**
   halt

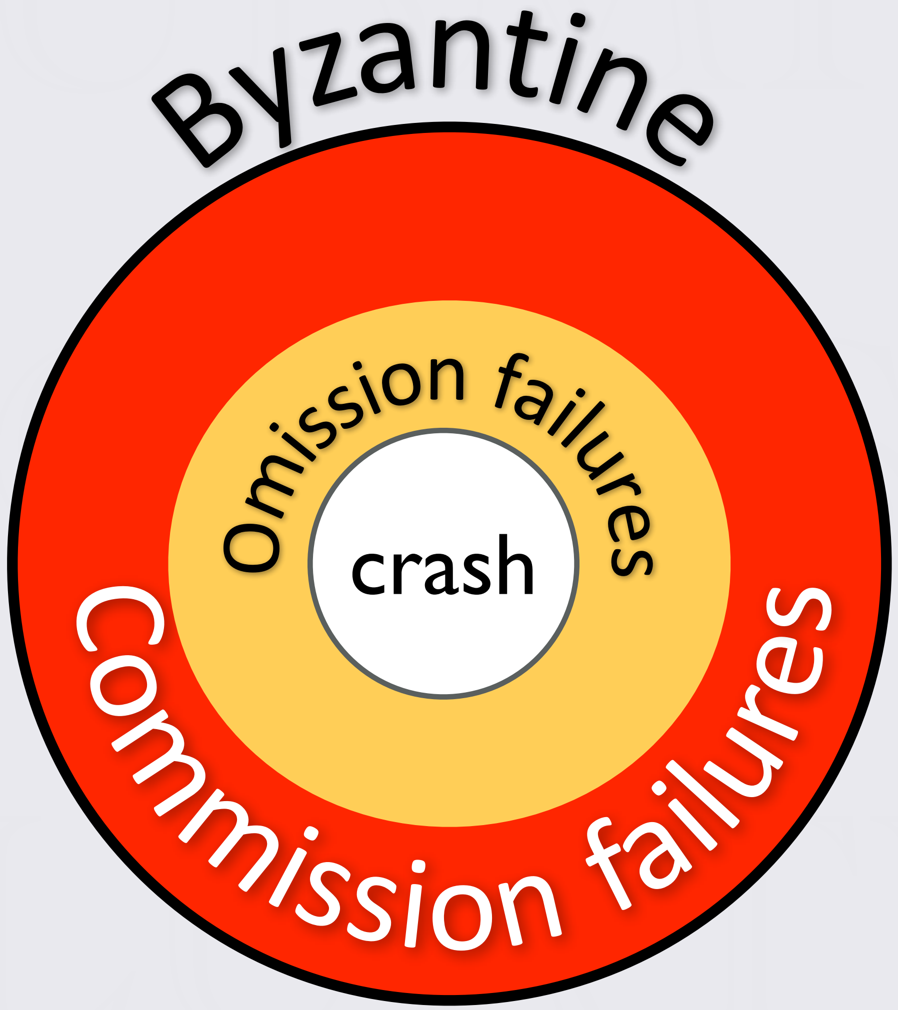
3. if (all votes are **Yes**) then
   send **Precommit** to all
   else
   $decision_c$:= **Abort**
   send **Abort** to all who voted **Yes**
   halt

4. if received **Precommit** then
   send **Ack**

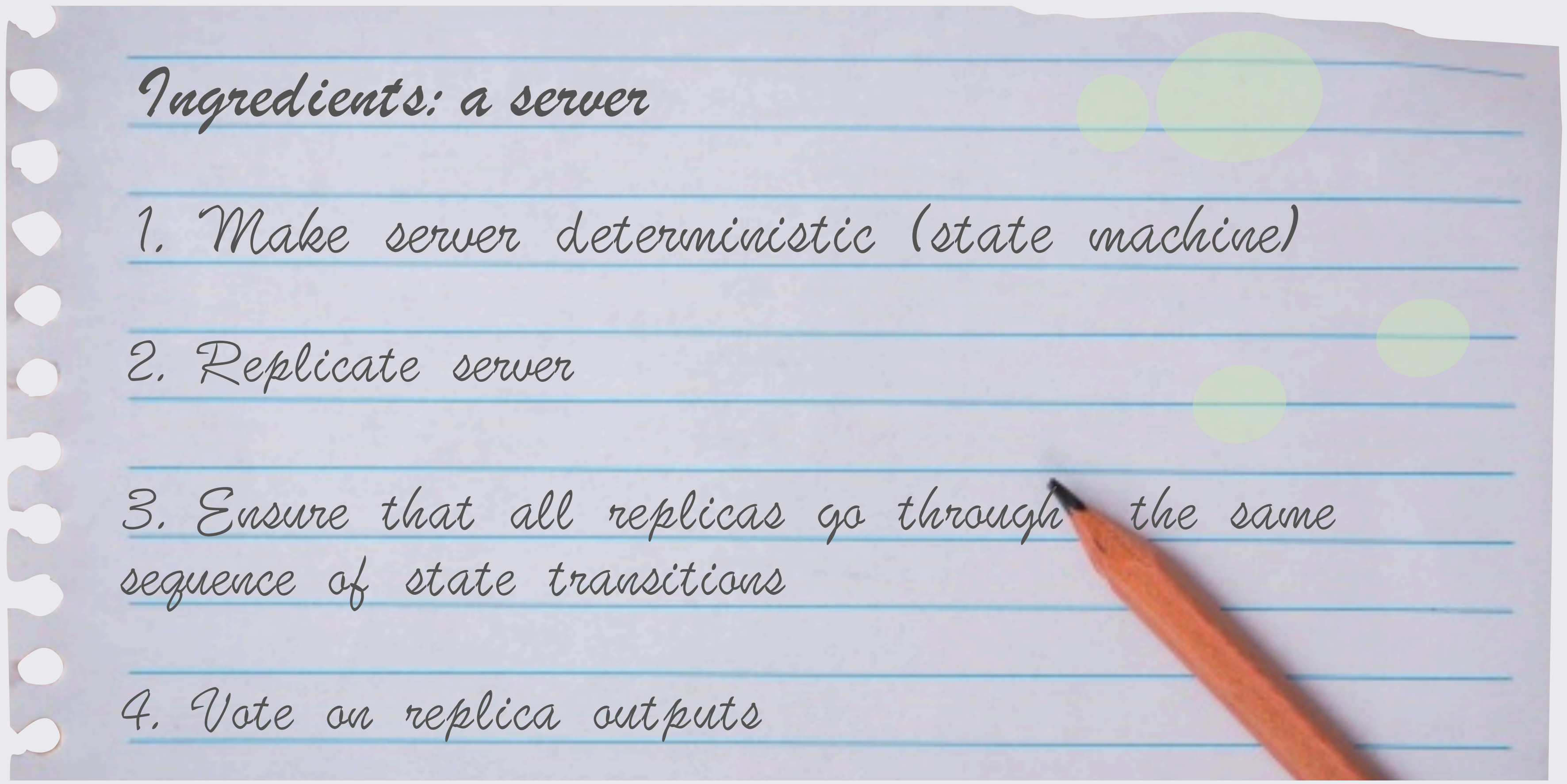
5. collect **Ack** from all participants
   When all **Ack**'s have been received:
   $decision_c$:= **Commit**
   send **Commit** to all

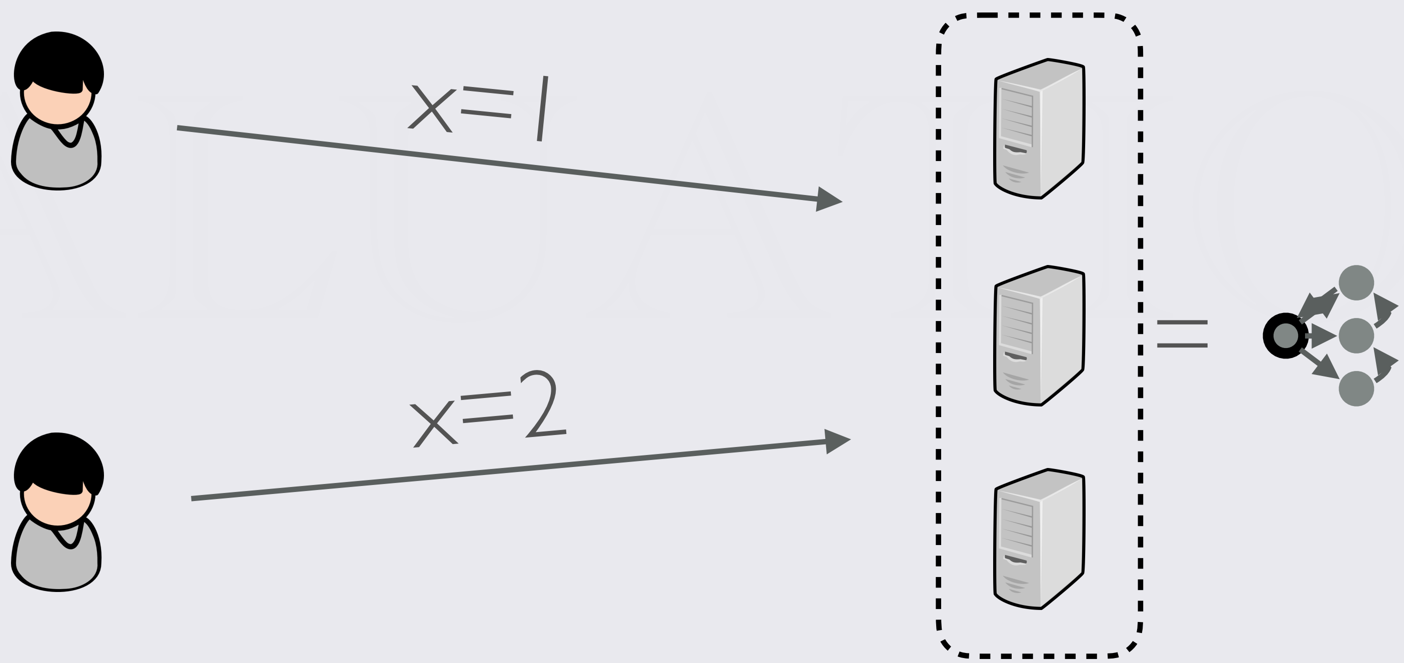
6. When p_i receives **Commit**, sets $decision_i$:= **Commit** and halts

A HIERARCHY OF FAILURE MODELS



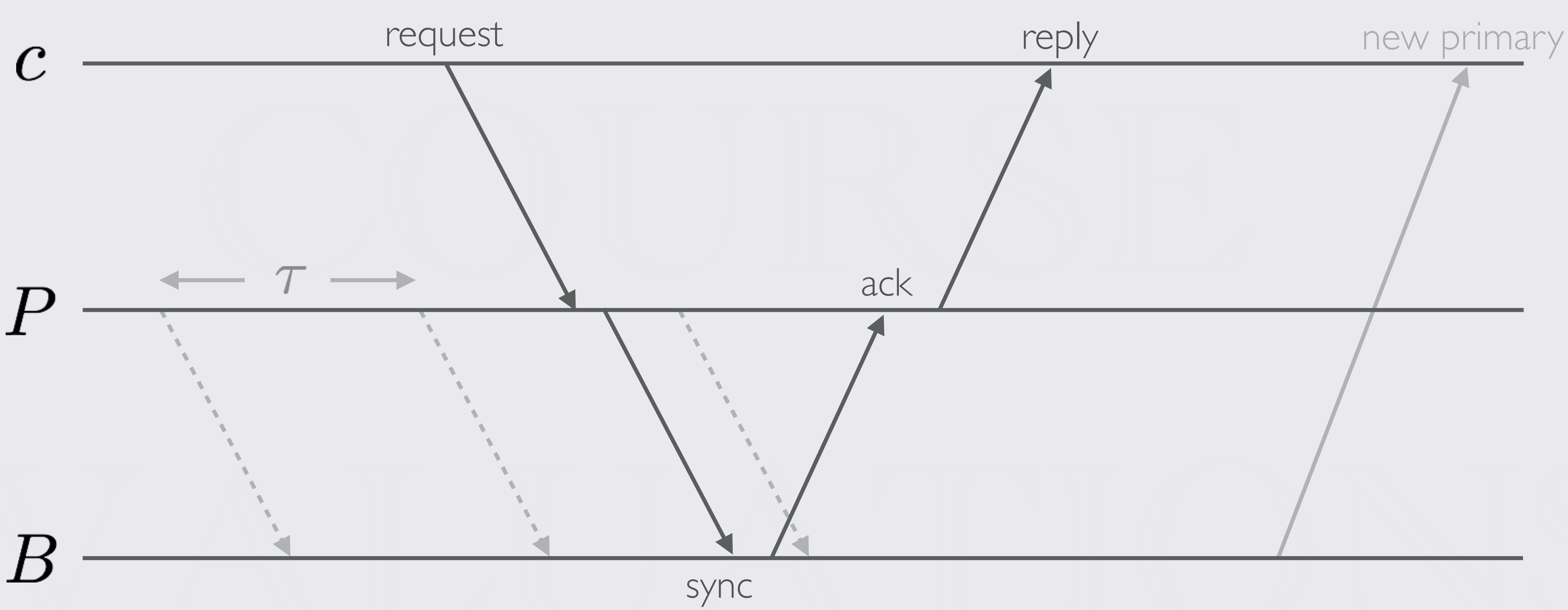
State Machine Replication





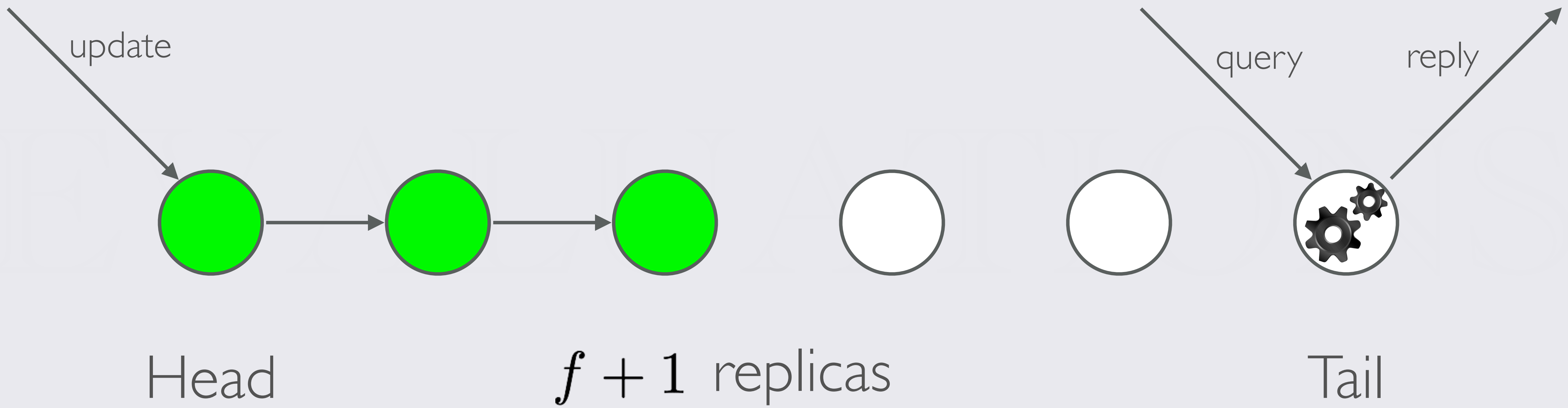
A PRIMARY-BACKUP PROTOCOL

$(f = 1)$



Chain replication

Tail can respond immediately, without waiting for the new update



Consensus

Validity If all processes that propose a value propose v, then all correct processes eventually decide v

Agreement If a correct process decides v, then all correct processes eventually decide v

Integrity Every correct process decides at most one value, and if it decides v, then some process must have proposed v

Termination Every correct process eventually decides some value

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

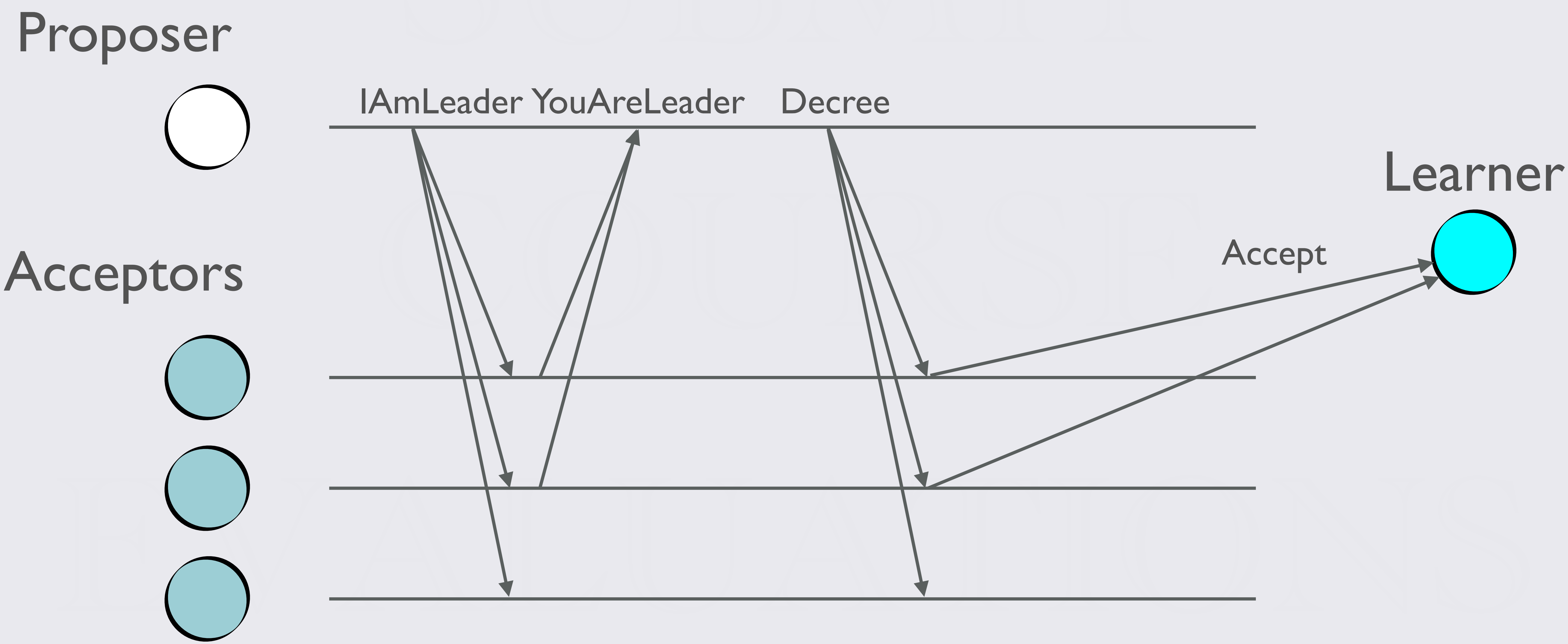
Fischer, Lynch, Paterson 1985

Paxos

Abstract

The Paxos algorithm, when presented in plain English, is very simple.

PAXOS AT WORK



ACCEPTOR STATES

(as leader #50 comes to power)

Acceptors	**Value**	**By leader**
	x	37
	-	-
	-	-
	-	-
	y	41

Question 4:
What is the set of possible values that leader #50 can propose?

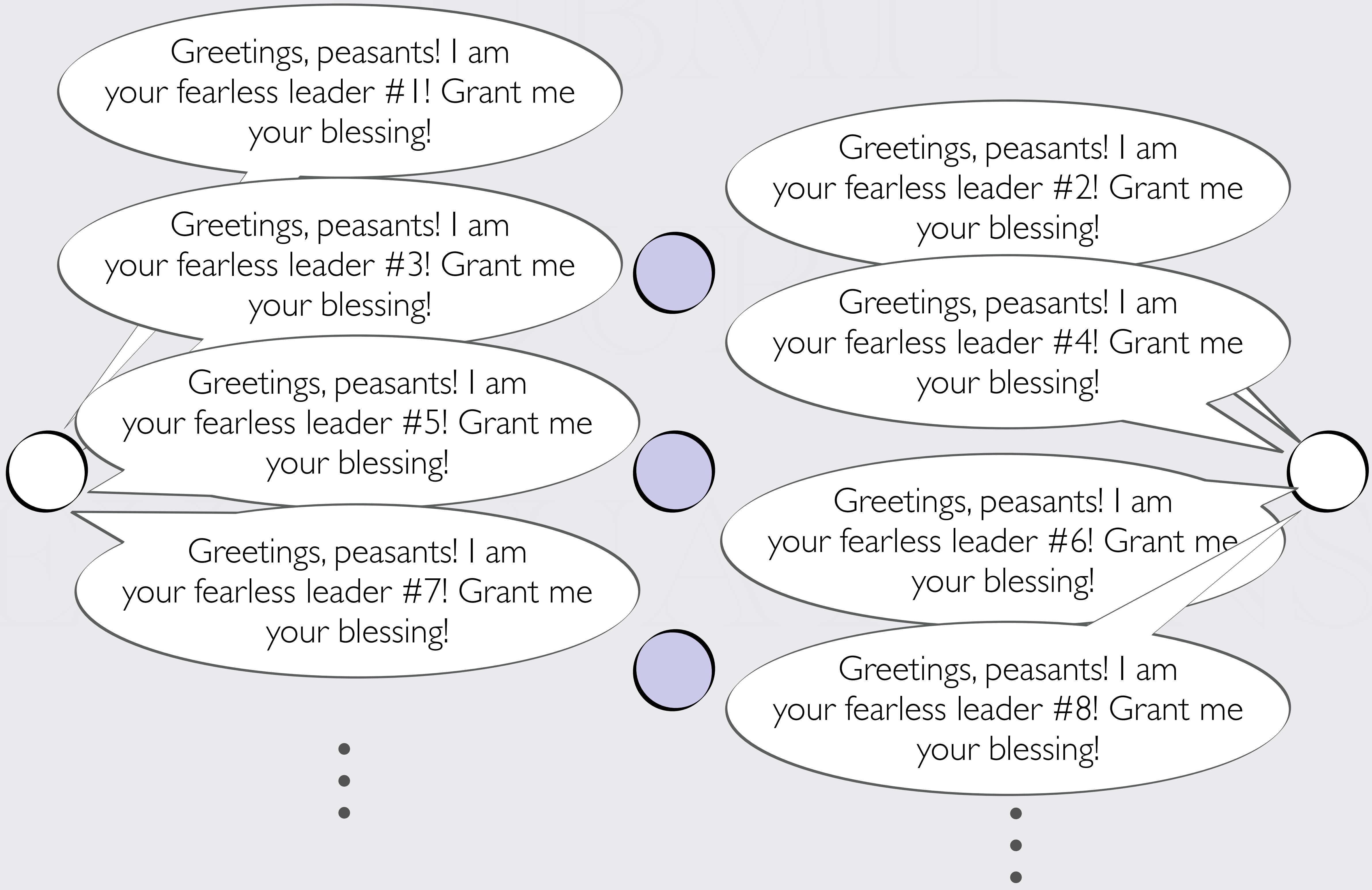
Examples of acceptor states

(as leader #50 comes to power)

Acceptors	Value	By leader
	x	37
	y	42
	-	-
	x	37
	x	41

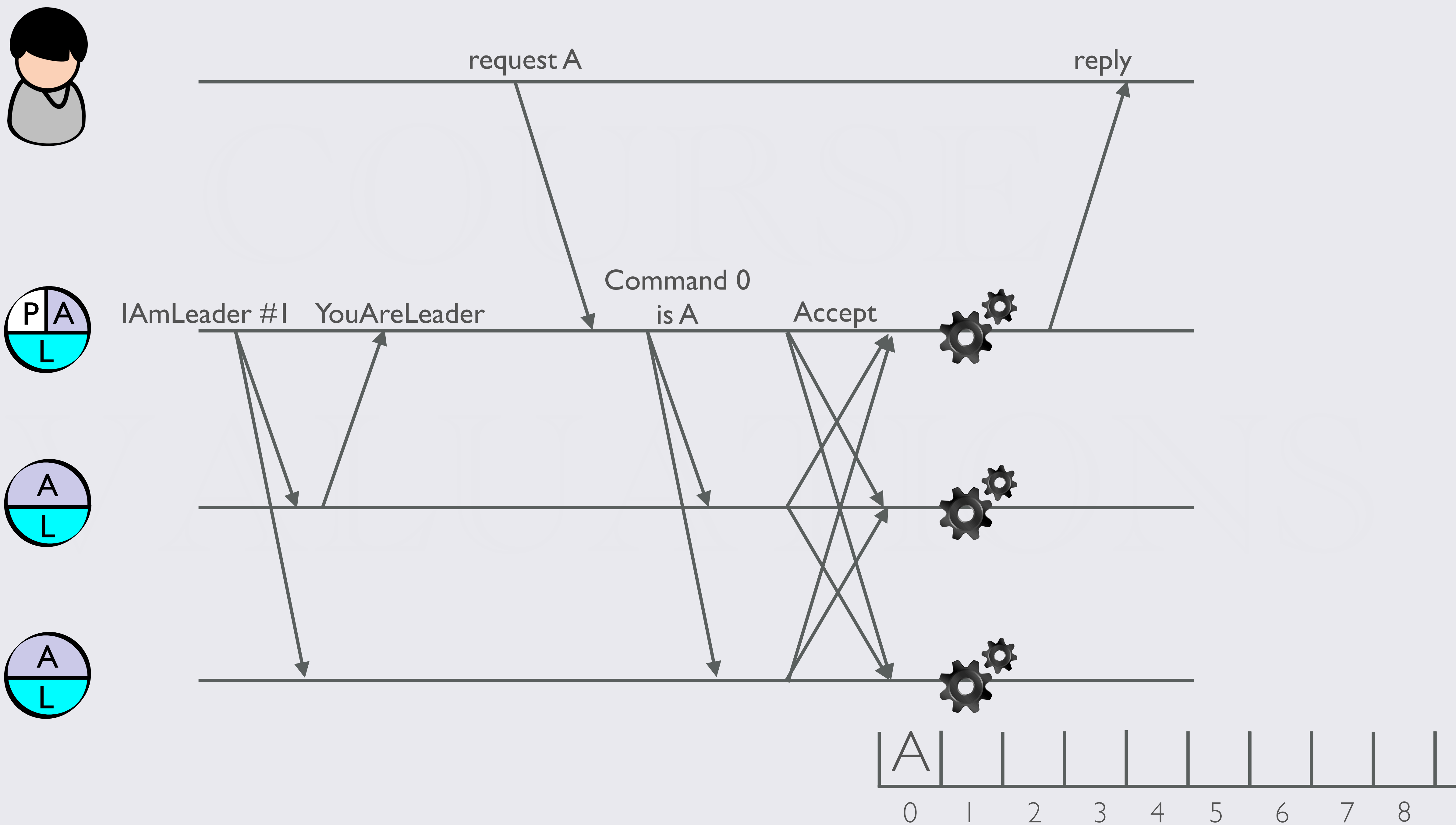
Question 5:
What is the set of possible values that leader #50 can propose?

THE THREAT TO LIVENESS: DUELING PROPOSERS

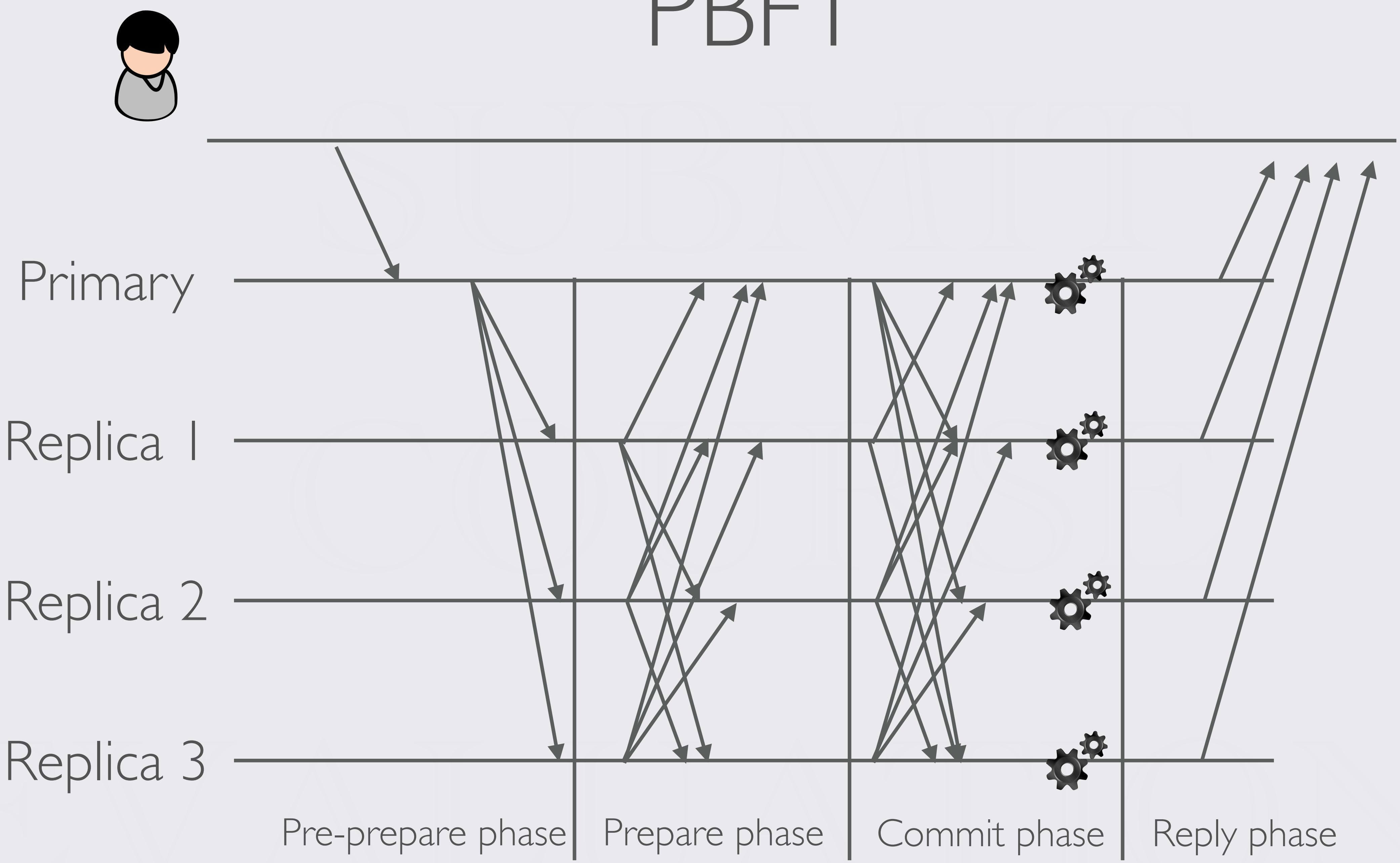


PAXOS/SMR IN REAL LIFE

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

EXECUTE-VERIFY

Execute

Verify

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 → you **don't** tolerate asynchrony

4. Fewer than 3f+1 replicas → 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

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 :-)

SUBMIT COURSE EVALUATIONS

ADMINISTRIVIA

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

Course evaluations due today

ADMINISTRIVIA

SUBMIT COURSE EVALUATIONS

Research project report due December **14**th
(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

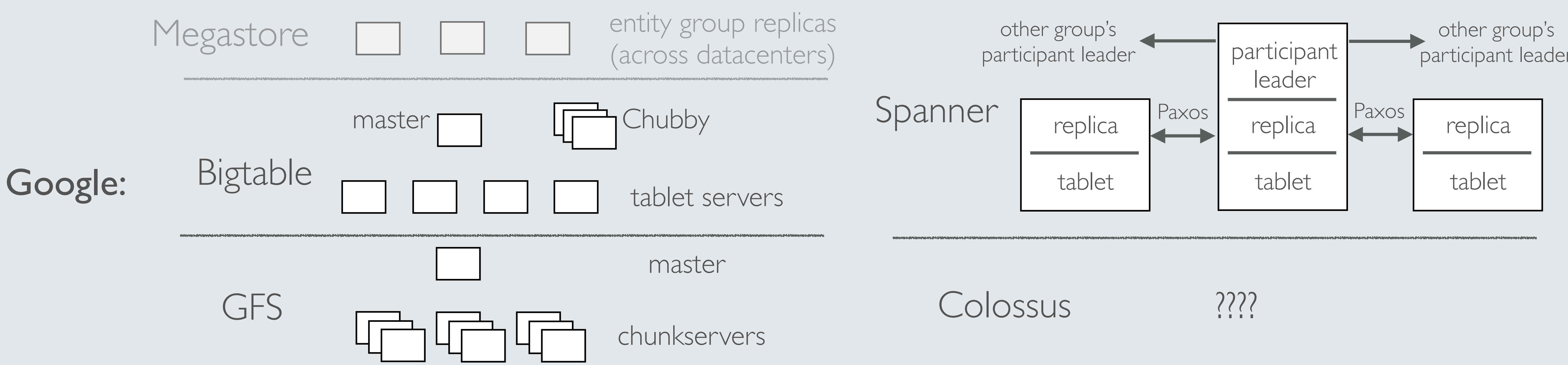
Falcon

Mencius

IronFleet

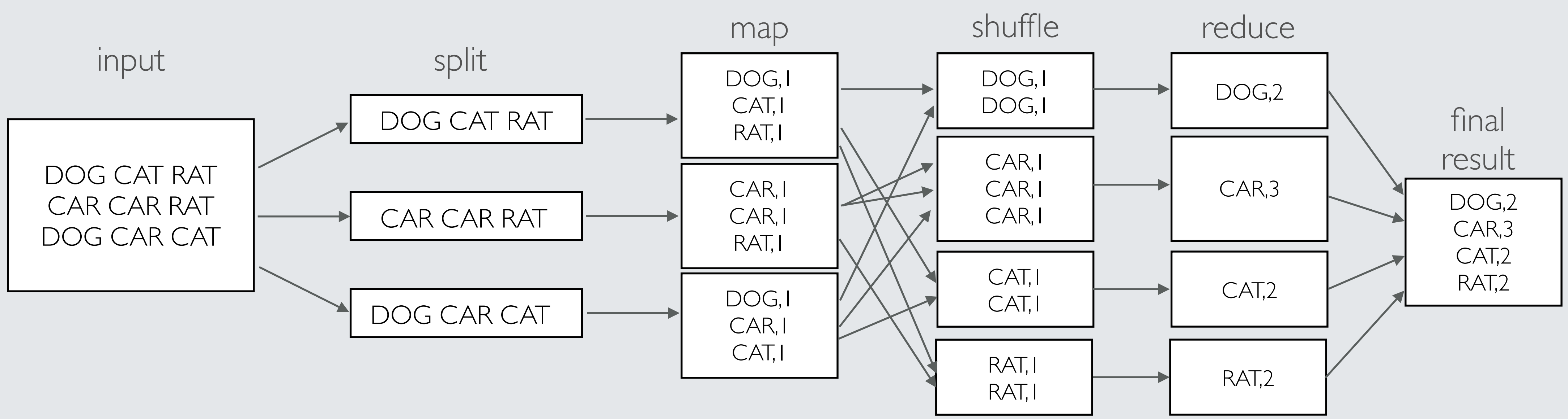
LARGE SCALE STORAGE SYSTEMS

Eventual and causal consistency
Bayou
Dynamo
COPS



Large scale computation systems

MapReduce



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

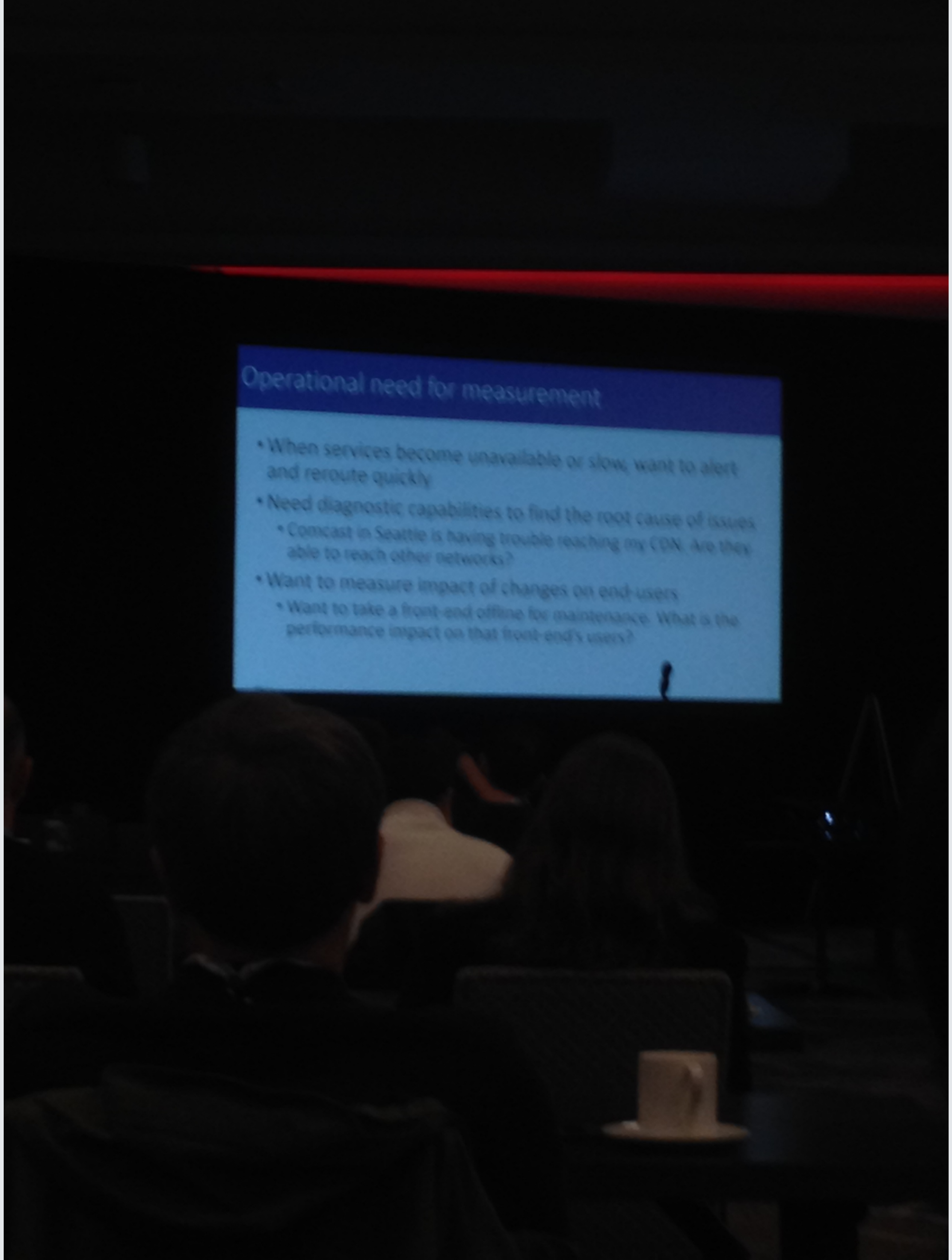
3. Read papers often

Presentations

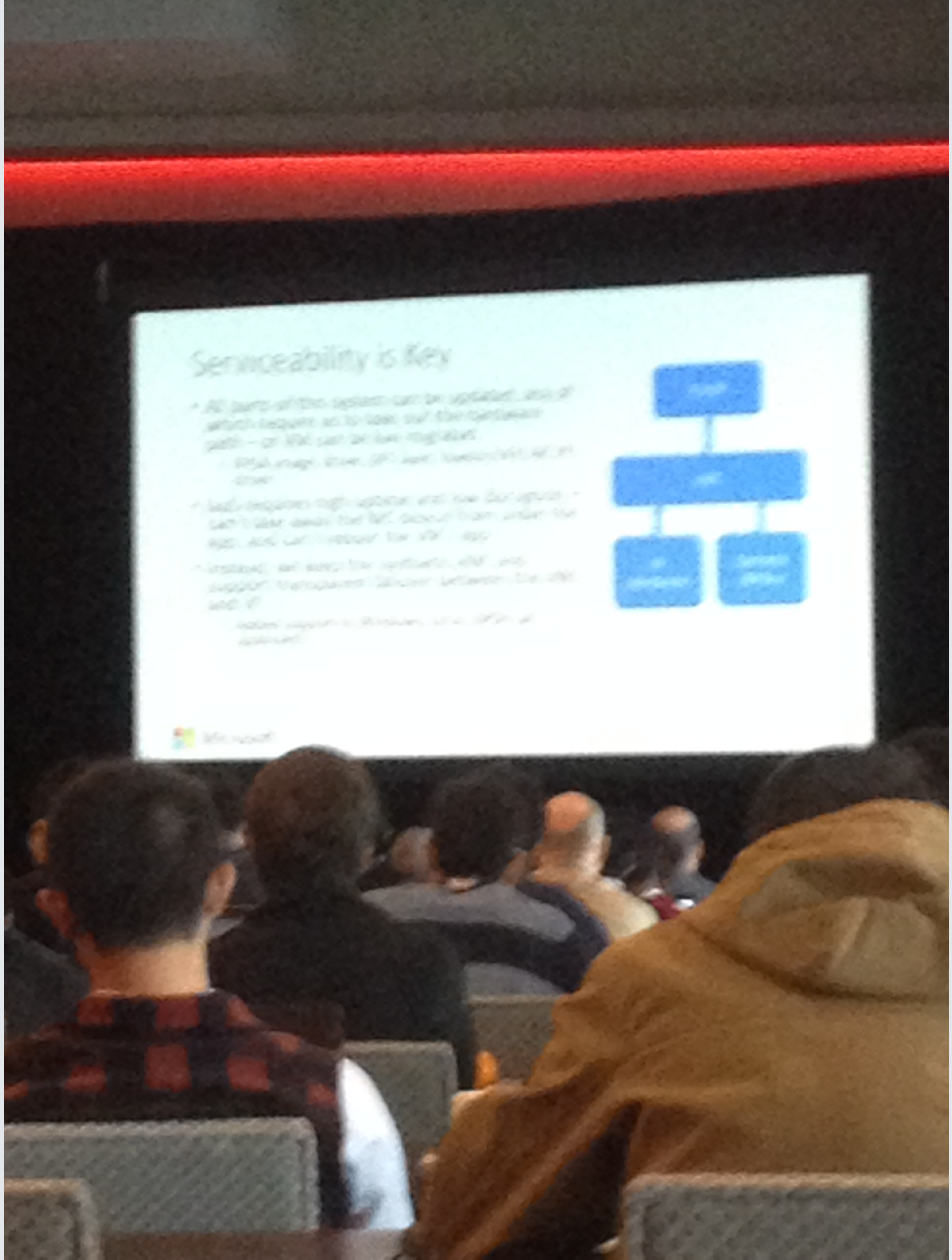
- Motivation, motivation, motivation!
- Keep it simple
  - Give the high-level intuition
  - Don't go too deep
- Avoid the "wall of text"
- Speak normally, with changes to your inflection
- Practice, practice, practice!

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?


Unified Flow Tables – A Fastpath Through VFP
VFP


Serviceability is Key


Serviceability is Key

Questions?

- On EECS591
- On distributed systems
- On computer science
- On research
- On Life, the Universe and Everything…

Thank you for attending EECS591!