Two-phase commit

Doug Woos

Logistics notes

Problem Set 2 out

- Due next Friday

Next week: Paxos

- Notoriously difficult topic

- Please do the reading!

Caching Discussion

"Complexity" as a downside

Do the scalability/performance issues mentioned in the paper exist today?

Why do we use NFS?

When is the recovery protocol executed?

Review: Sharding

Sharding

- Different data on different servers
- Partitioned via some function on keys
- Implement in Lab 4!

Another axis of scaling, cf. replication

- Usually, shard then replicate
- Each piece of data lives on one replicated shard

Sharding and Transactions

Adve rule we discussed lets us avoid synchronizing between shards for simple reads and writes

But what if we do need to synchronize across shards?

Transactions: atomic updates to multiple items

- Bank account transfer
- Calendar event
- Unix user creation
- etc.

Calendar event creation

I have three advisors (Tom, Zach, Mike)

Want to schedule a meeting with all of them

- Let’s try Tues at 11, people are usually free then

Calendars all live on different nodes!

Other students also trying to schedule meetings

Nodes can fail, messages can be dropped (of course)

Calendar event creation (wrong)

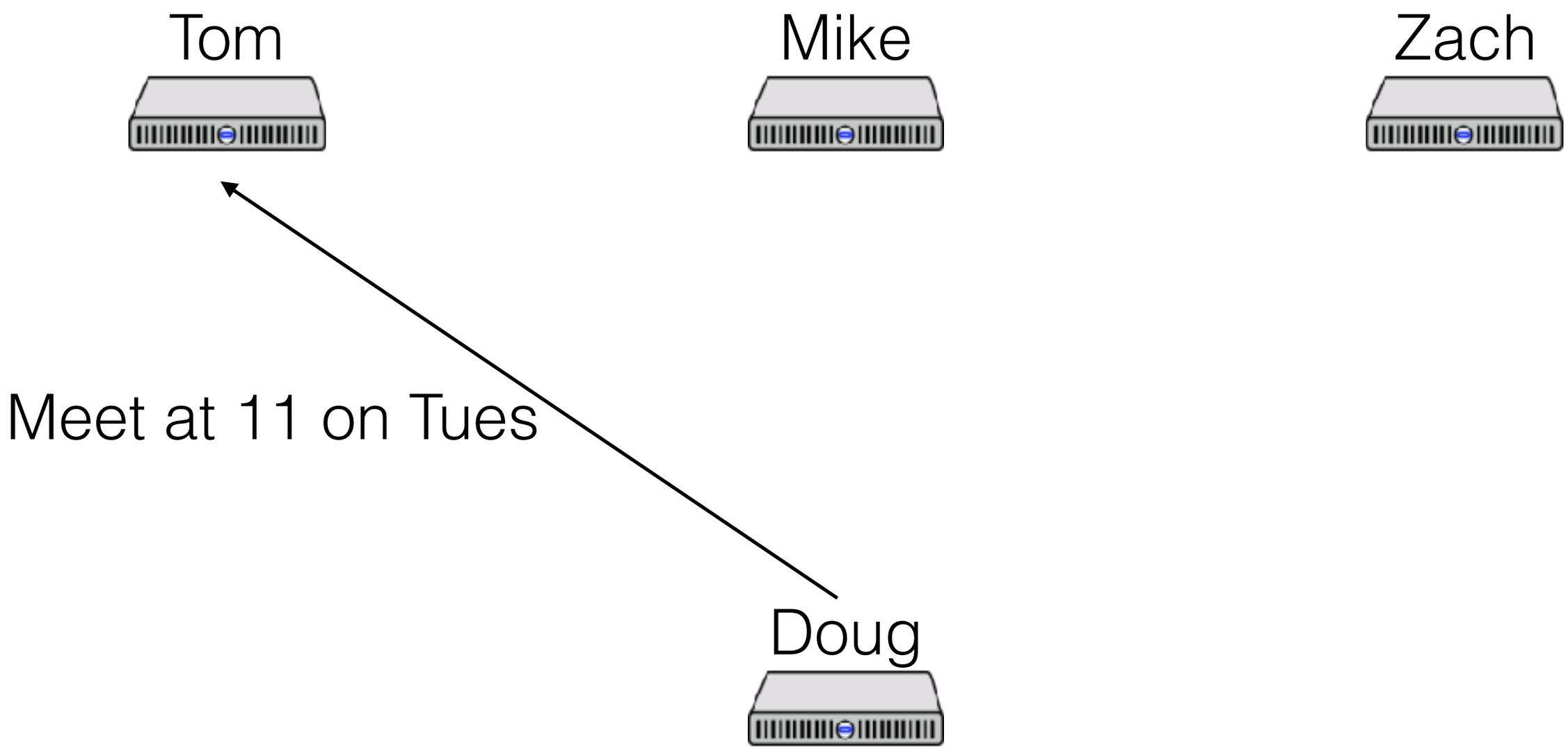
Tom

Mike

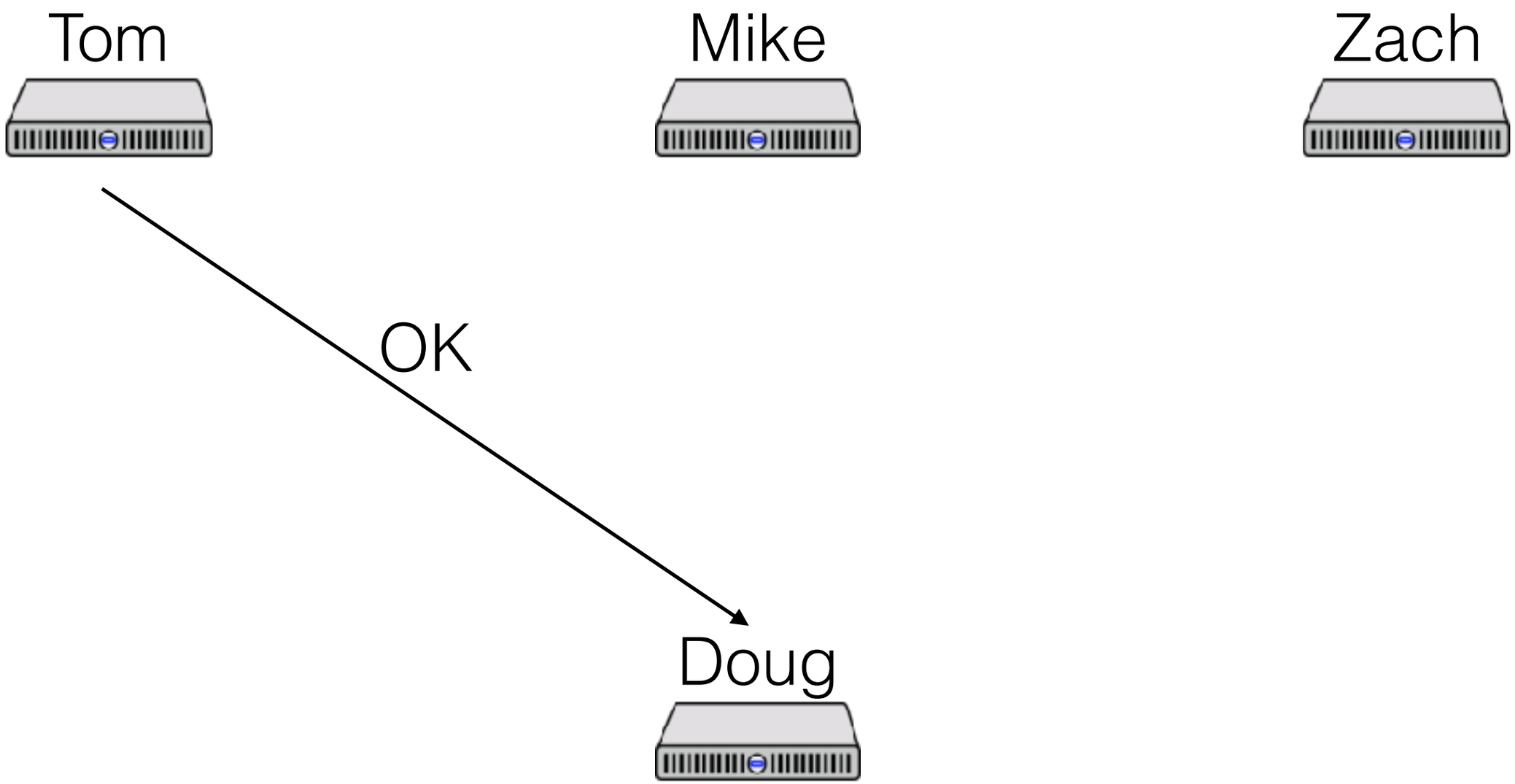
Zach

Doug

Calendar event creation (wrong)



Calendar event creation (wrong)



Calendar event creation (wrong)

Tom

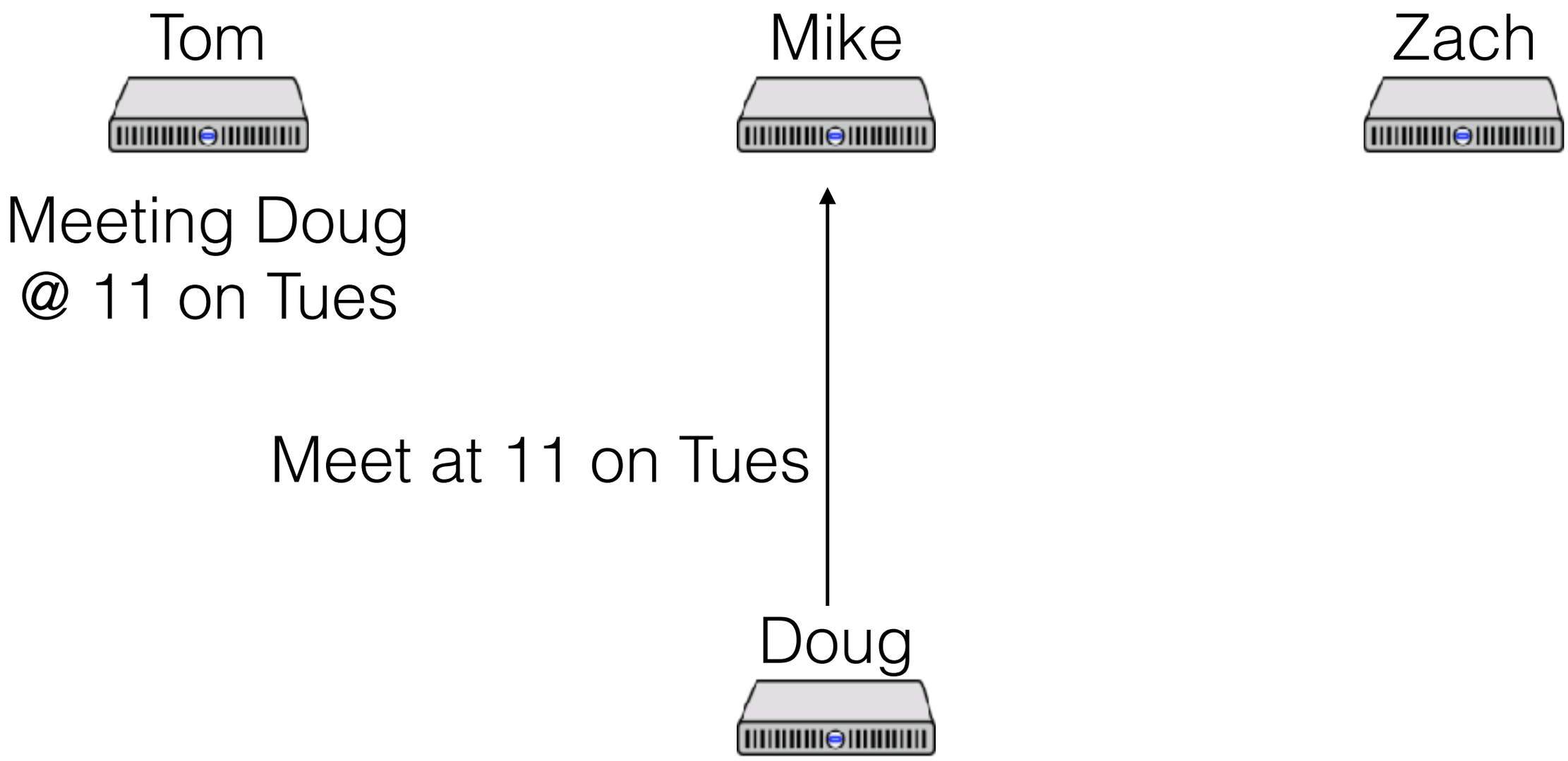
Meeting Doug
@ 11 on Tues

Mike

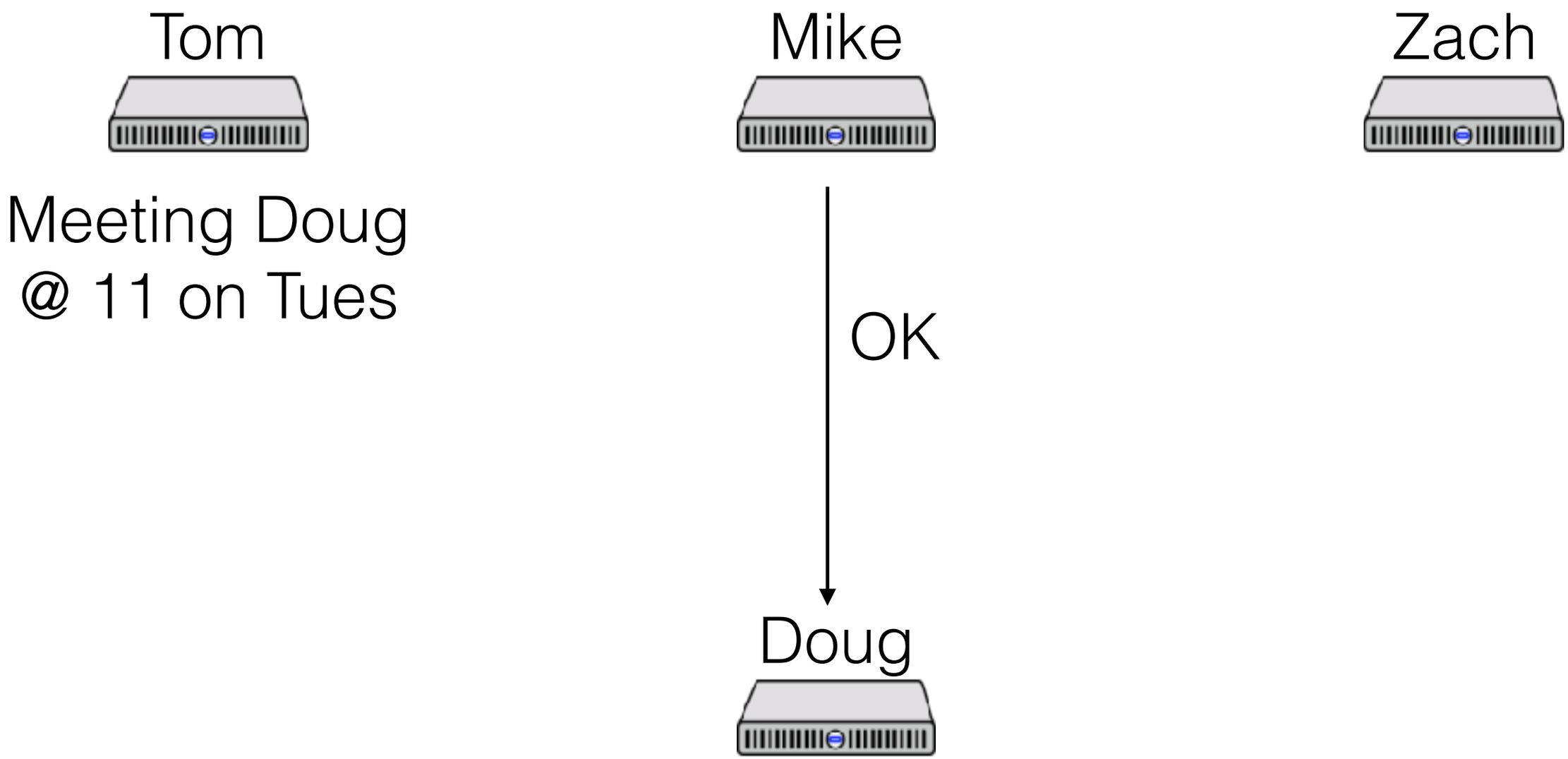
Zach

Doug

Calendar event creation (wrong)



Calendar event creation (wrong)



Calendar event creation (wrong)

Tom

Meeting Doug
@ 11 on Tues

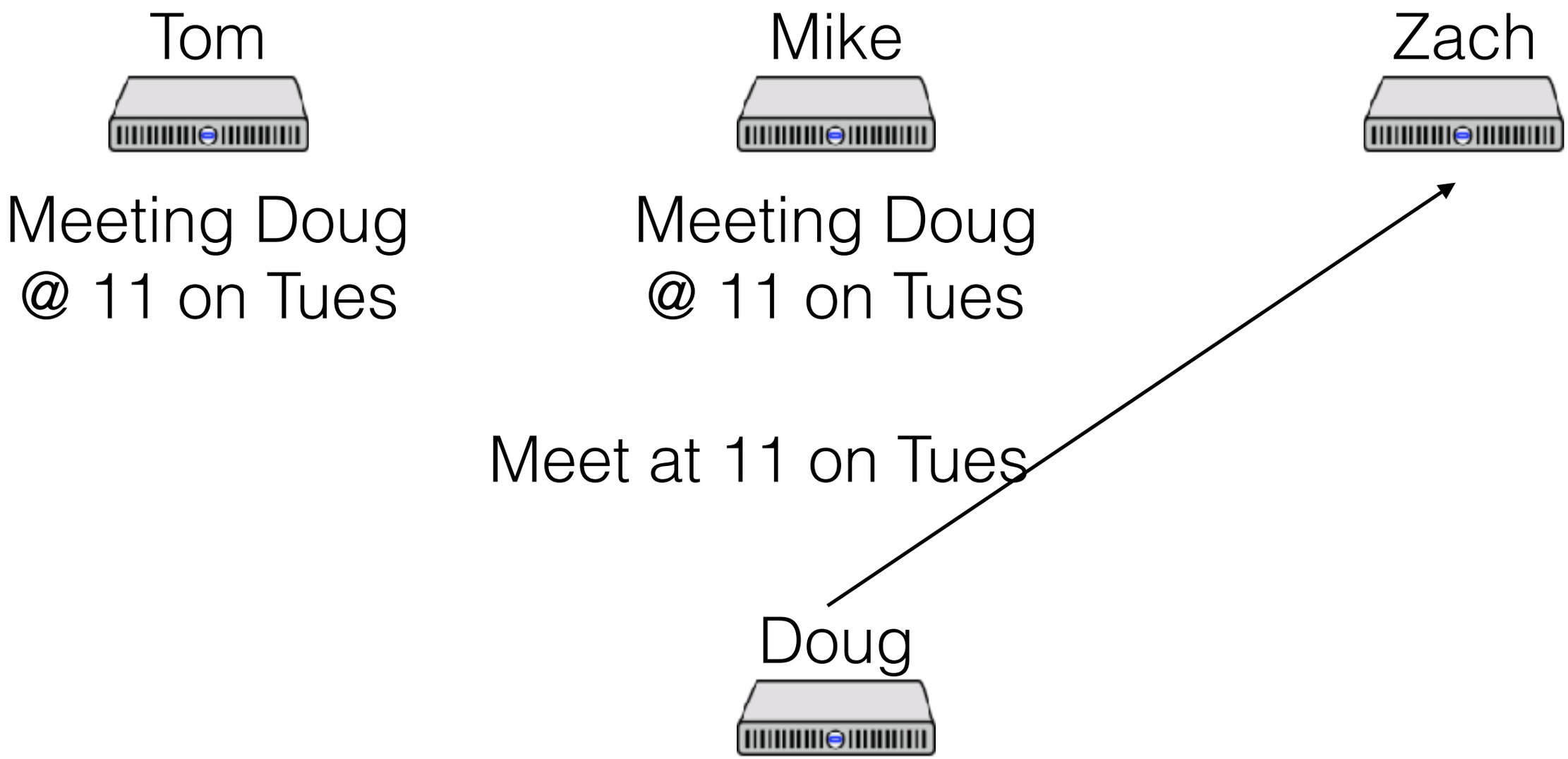
Mike

Meeting Doug
@ 11 on Tues

Zach

Doug

Calendar event creation (wrong)



Calendar event creation (wrong)

Tom

Meeting Doug
@ 11 on Tues

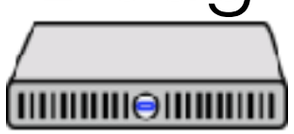
Mike

Meeting Doug
@ 11 on Tues

Zach

Busy!

Doug



Calendar event creation (wrong)

Tom

Meeting Doug
@ 11 on Tues

Mike

Meeting Doug
@ 11 on Tues

Zach

Doug

Calendar event creation (wrong)

Tom

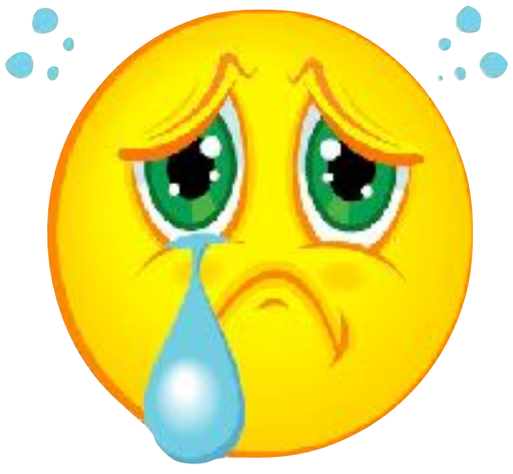
Meeting Doug
@ 11 on Tues

Mike

Meeting Doug
@ 11 on Tues

Zach

Doug



Calendar event creation (better)

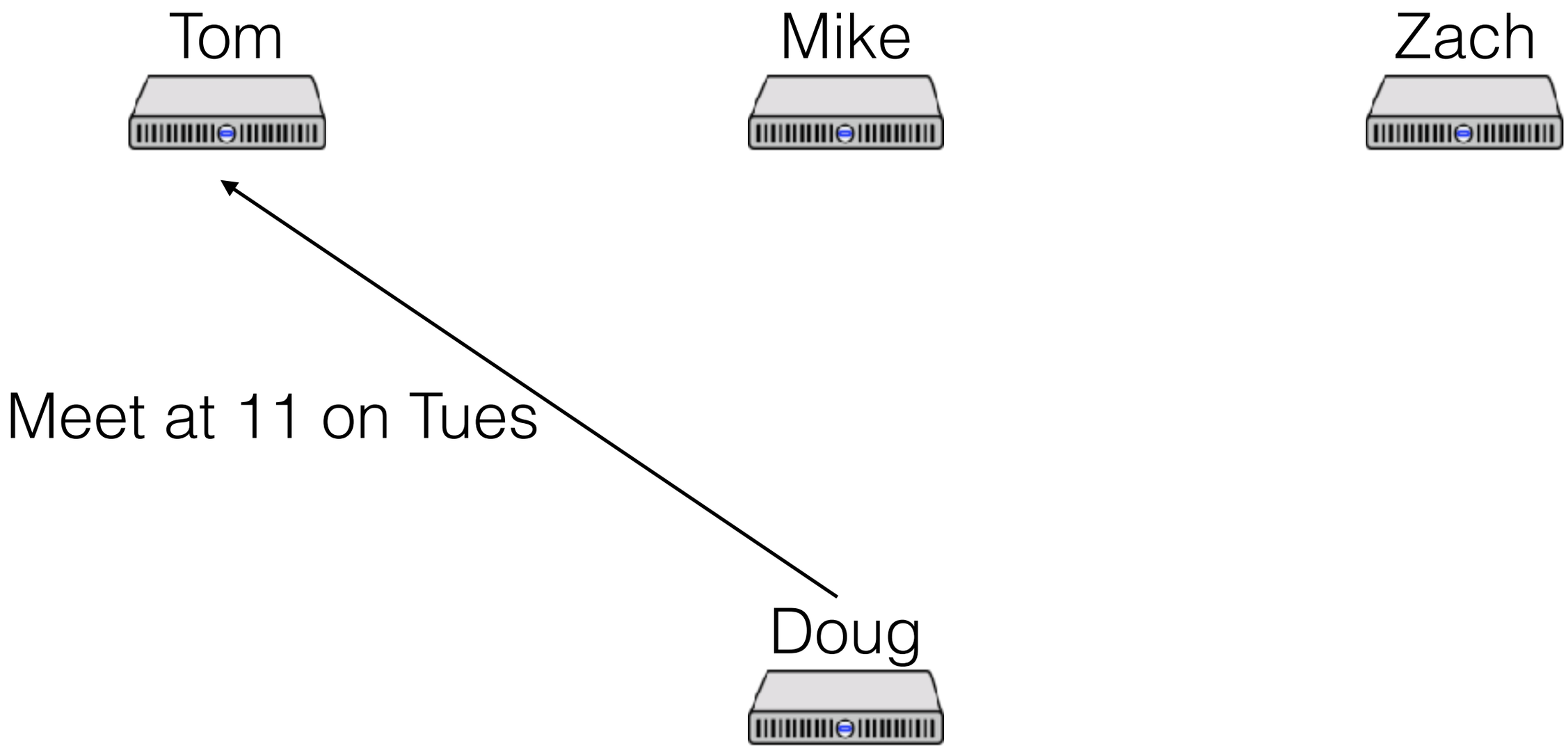
Tom

Mike

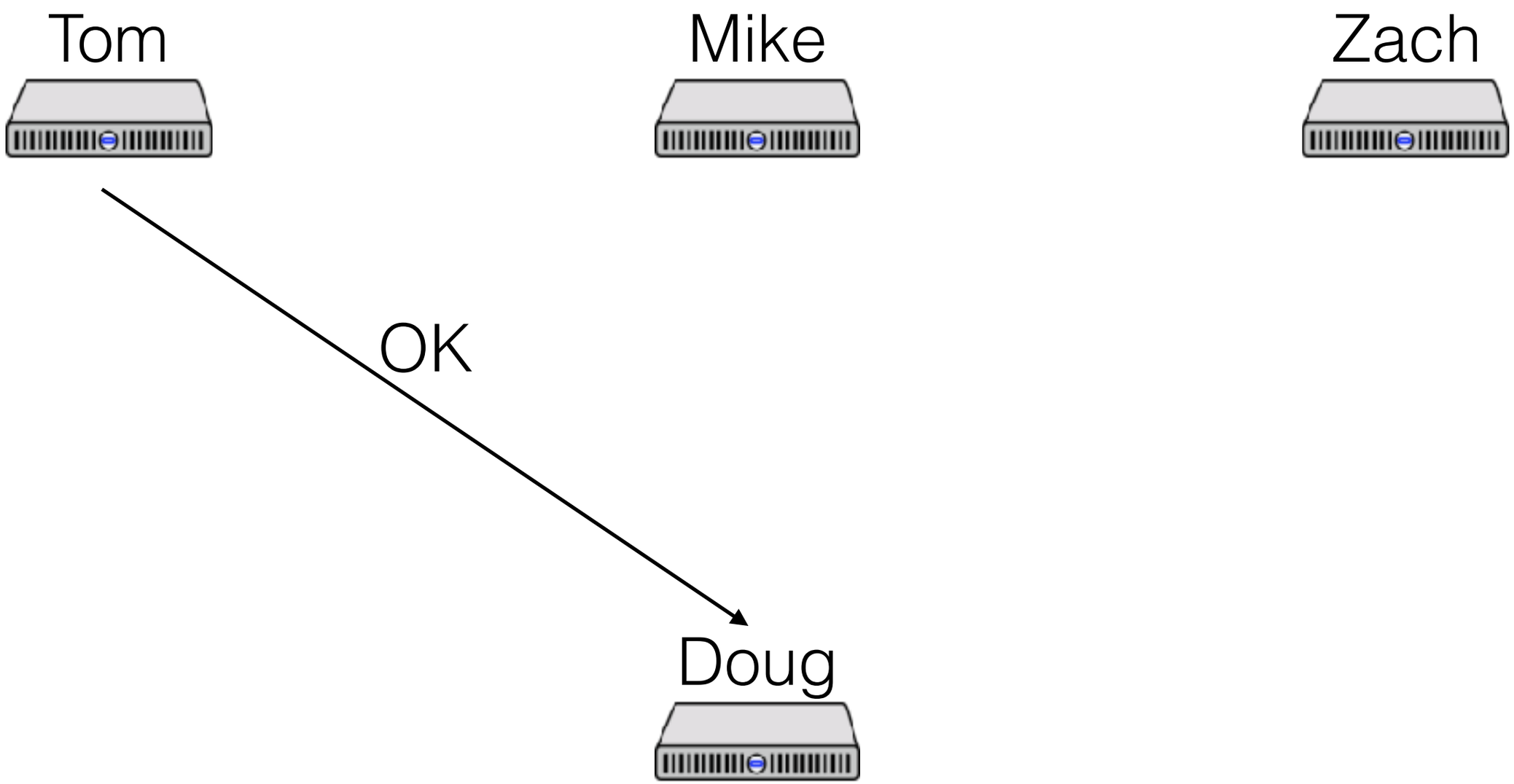
Zach

Doug

Calendar event creation (better)



Calendar event creation (better)



Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

Mike

Zach

Doug

Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

Mike

Zach

Meet at 11 on Tues

Doug

Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

Mike

OK

Zach

Doug

Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

Mike

Maybe Meeting
Doug @ 11 on Tues

Zach

Doug

Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

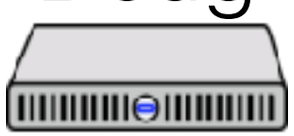
Mike

Maybe Meeting
Doug @ 11 on Tues

Zach

Meet at 11 on Tues

Doug



Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

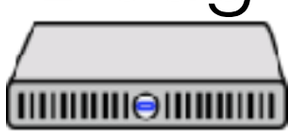
Mike

Maybe Meeting
Doug @ 11 on Tues

Zach

Busy!

Doug



Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

Mike

Maybe Meeting
Doug @ 11 on Tues

Zach

Doug

Calendar event creation (better)

Tom

Maybe Meeting
Doug @ 11 on Tues

Mike

Maybe Meeting
Doug @ 11 on Tues

Zach

Never mind!

Doug

Calendar event creation (better)

Tom

Mike

Zach

Maybe Meeting
Doug @ 11 on Tues

Doug

Calendar event creation (better)

Tom

Mike

Zach

Maybe Meeting
Doug @ 11 on Tues

Never mind!

Doug

Calendar event creation (better)

Tom

Mike

Zach

Doug

Two-phase commit

Atomic commit protocol (ACP)

- Every node arrives at the same decision

- Once a node decides, it never changes

- Transaction committed only if all nodes vote Yes

- In normal operation, if all processes vote Yes the transaction is committed

- If all failures are eventually repaired, the transaction is eventually either committed or aborted

Two-phase commit

Roles:

- Participants (Mike, Tom, Zach): nodes that must update data relevant to the transaction

- Coordinator (Doug): node responsible for executing the protocol (might also be a participant)

RPCs:

- **VOTE-REQ:** Can you commit this transaction?

- **COMMIT:** Commit this transaction

- **ABORT:** Abort this transaction

Failures

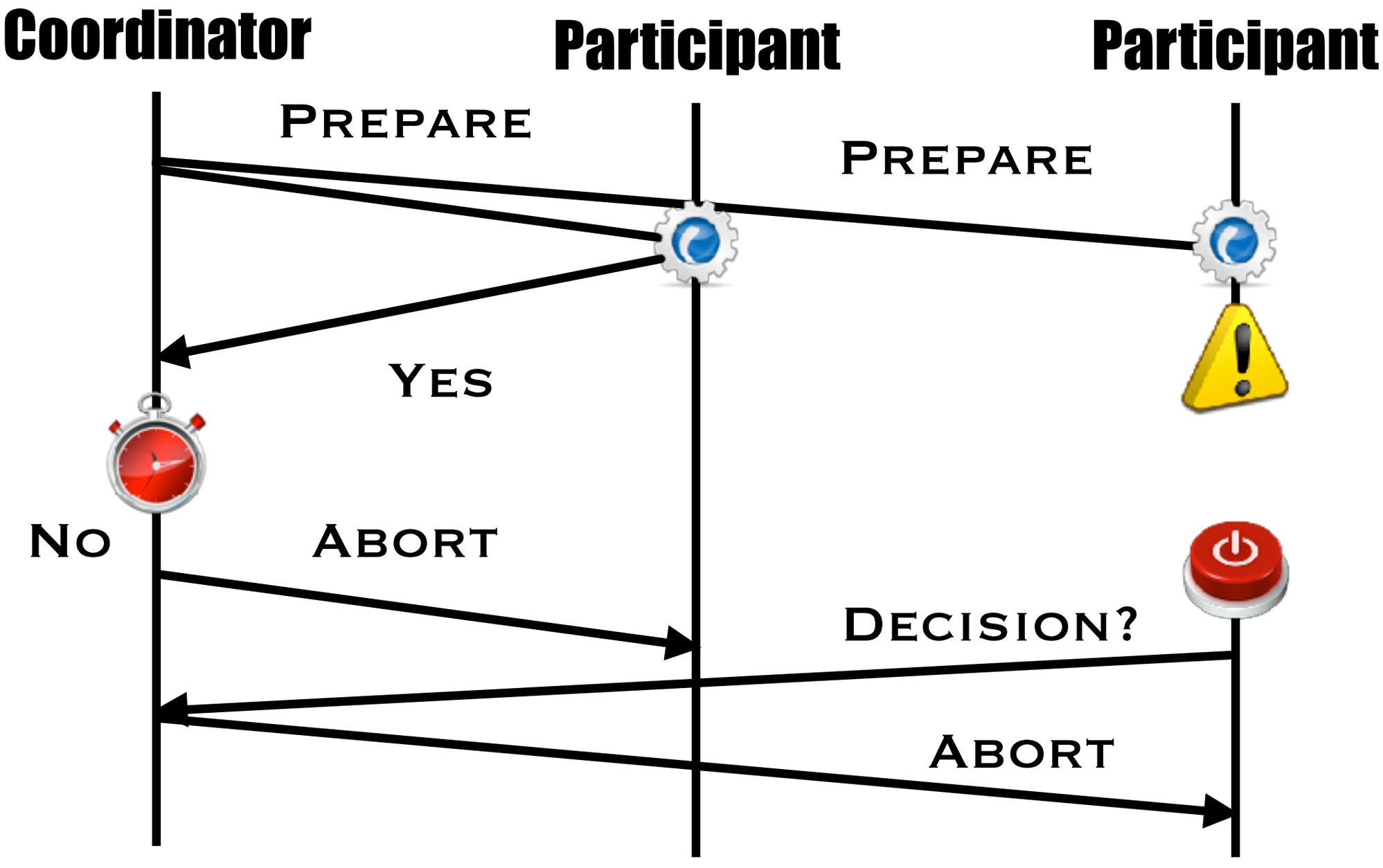
In the absence of failures, 2PC is pretty simple!

When can interesting failures happen?

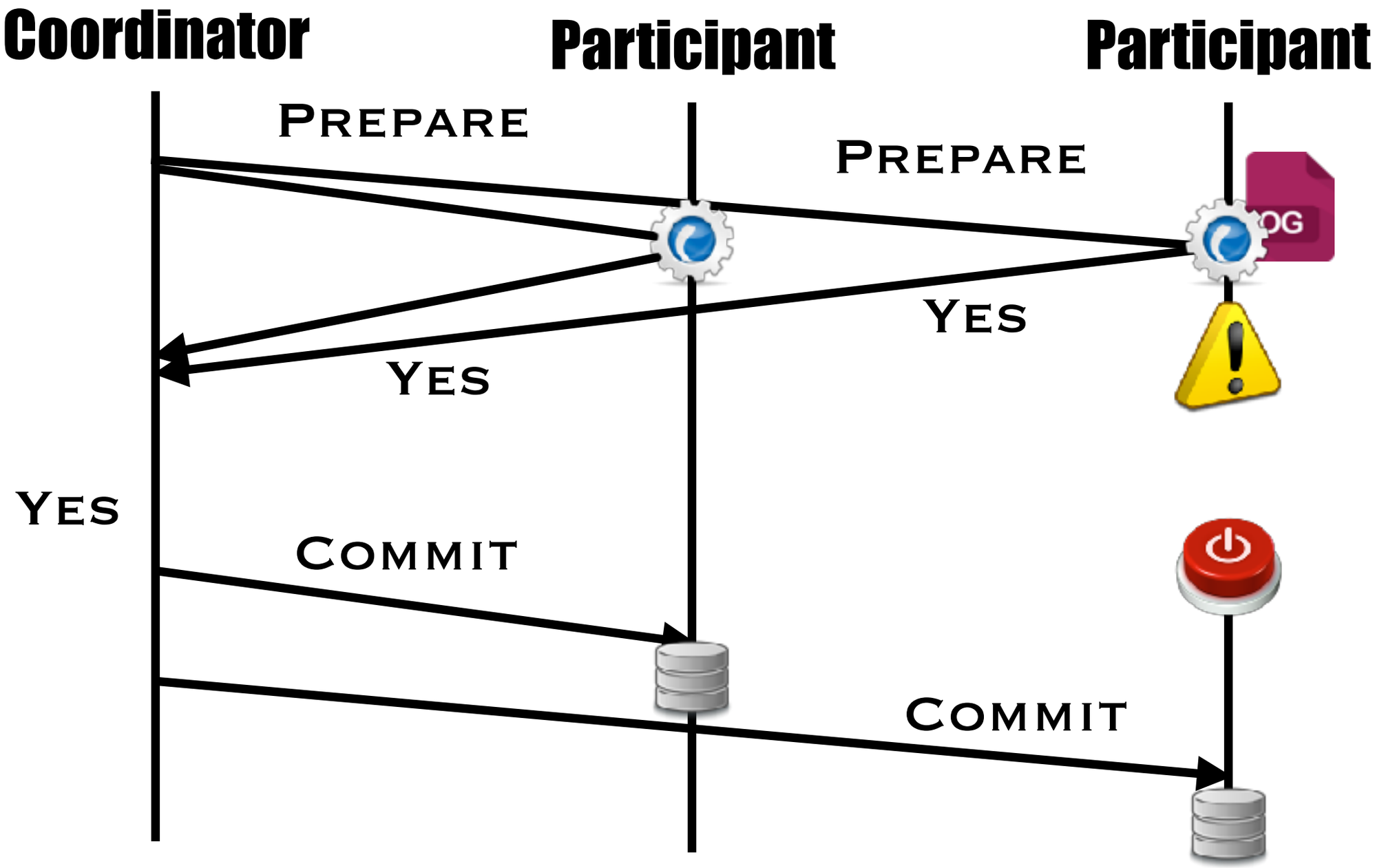
- Participant failures?
- Coordinator failures?
- Message drops?

Participant failures

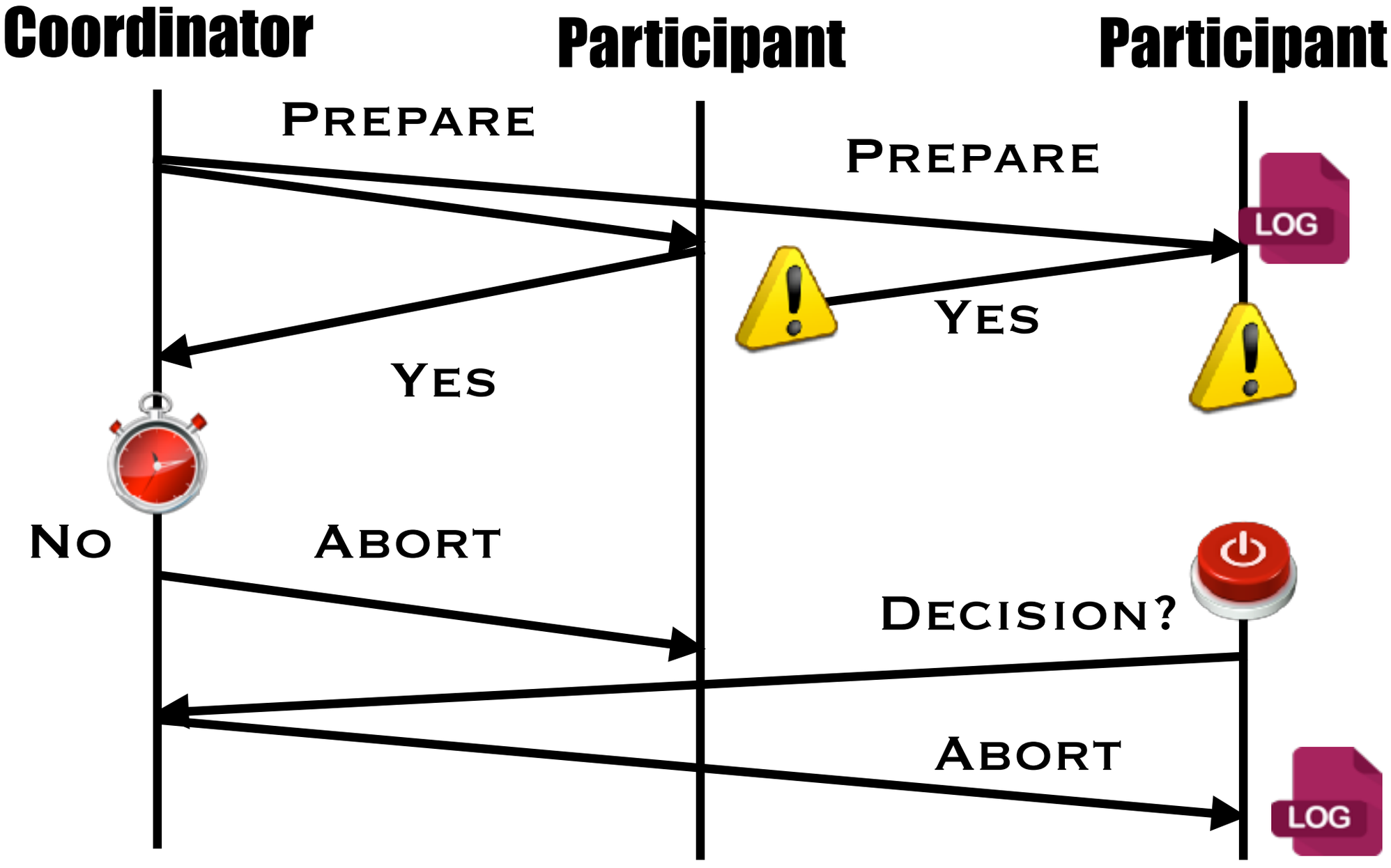
Participant failures: Before sending response?



Participant failures: After sending vote?

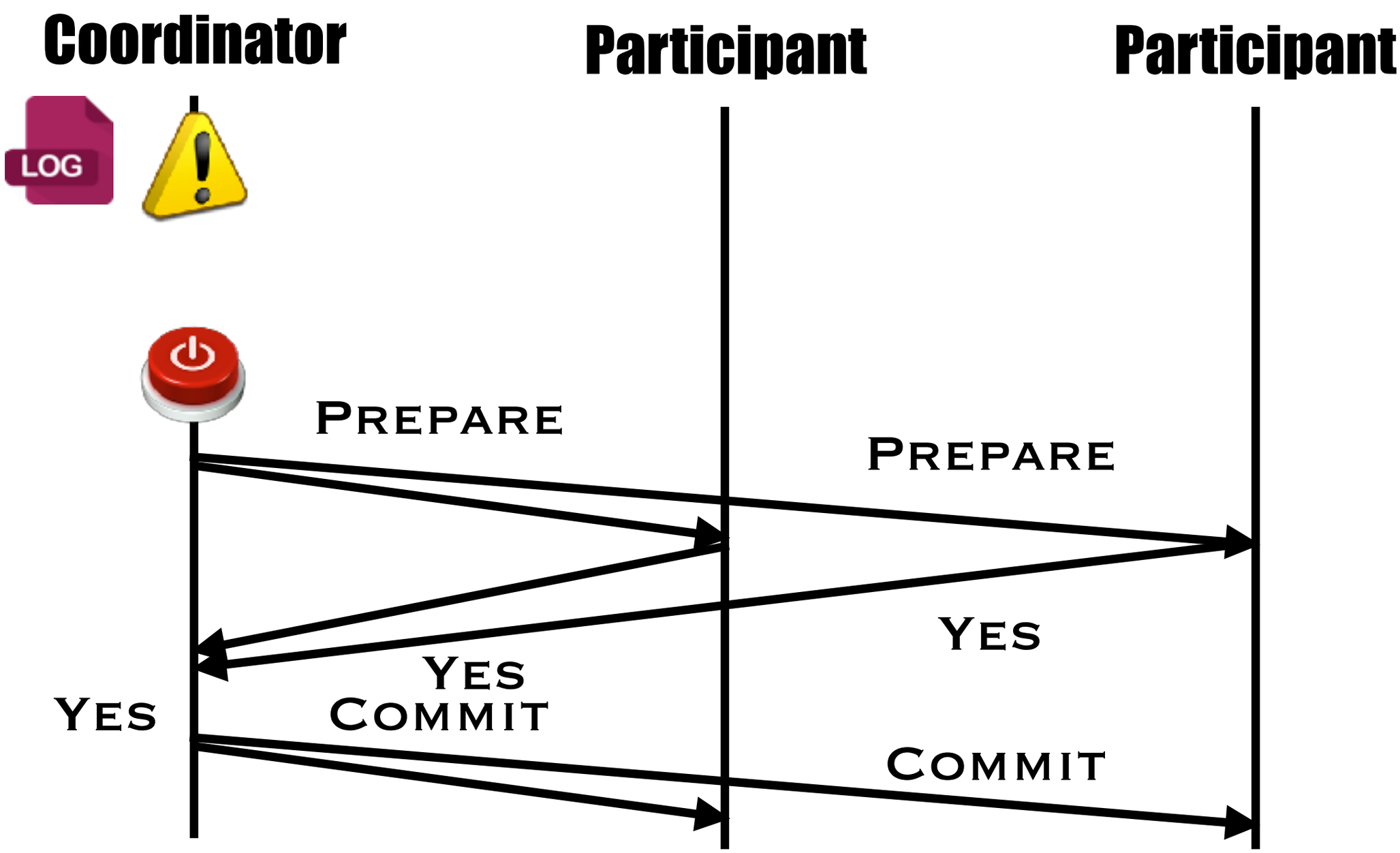


Participant failures: Lost vote?

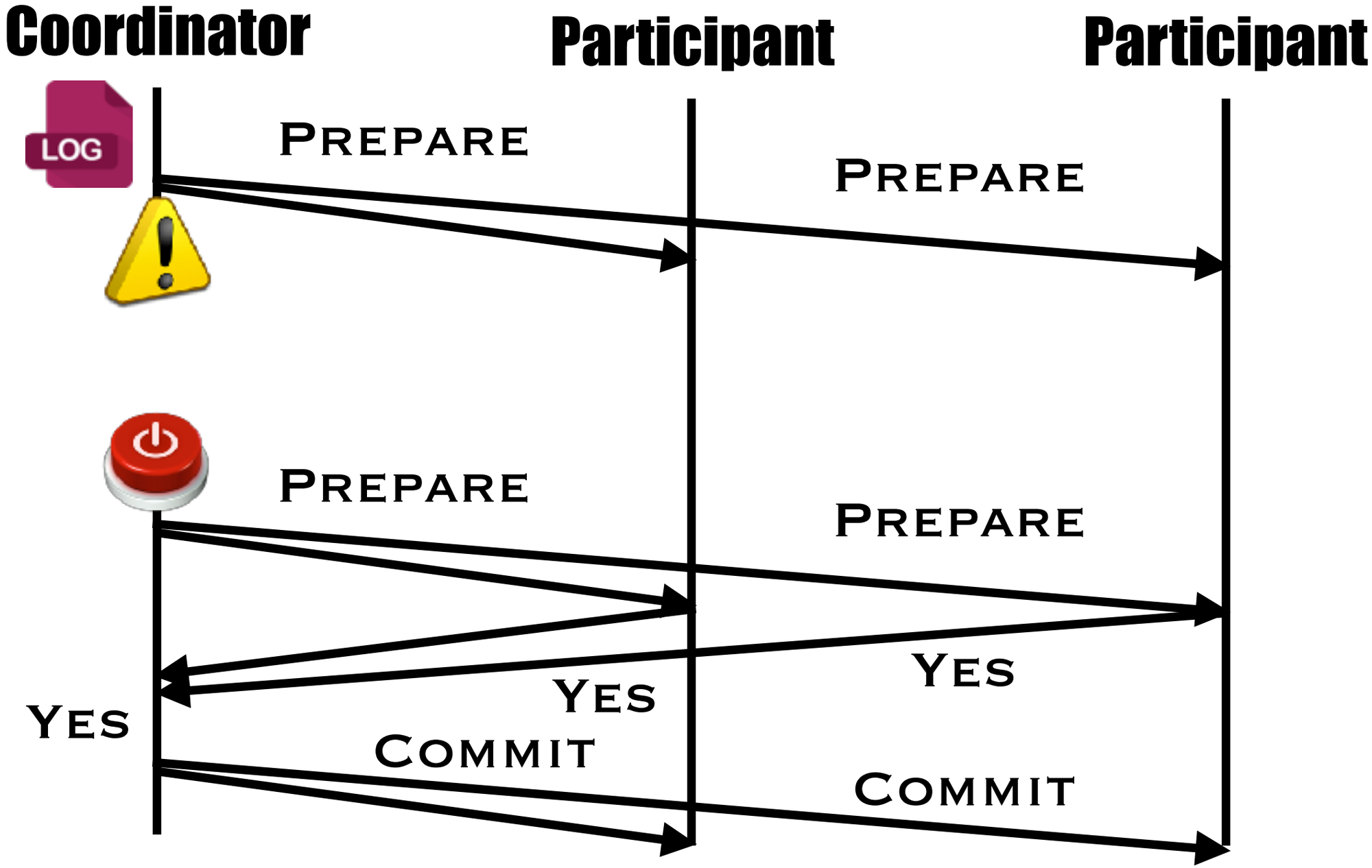


Coordinator failures

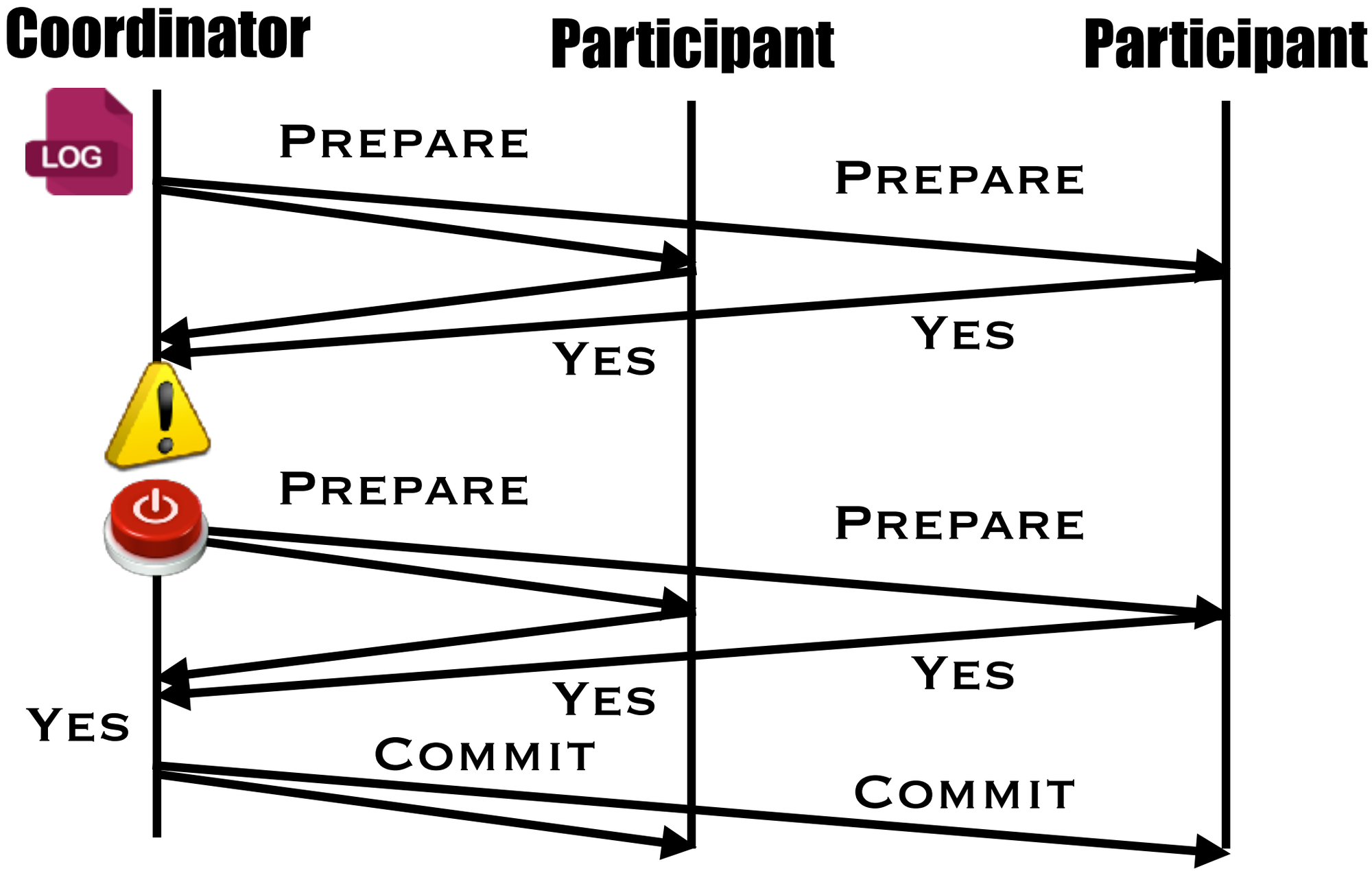
Coodinator failures:
Before sending prepare



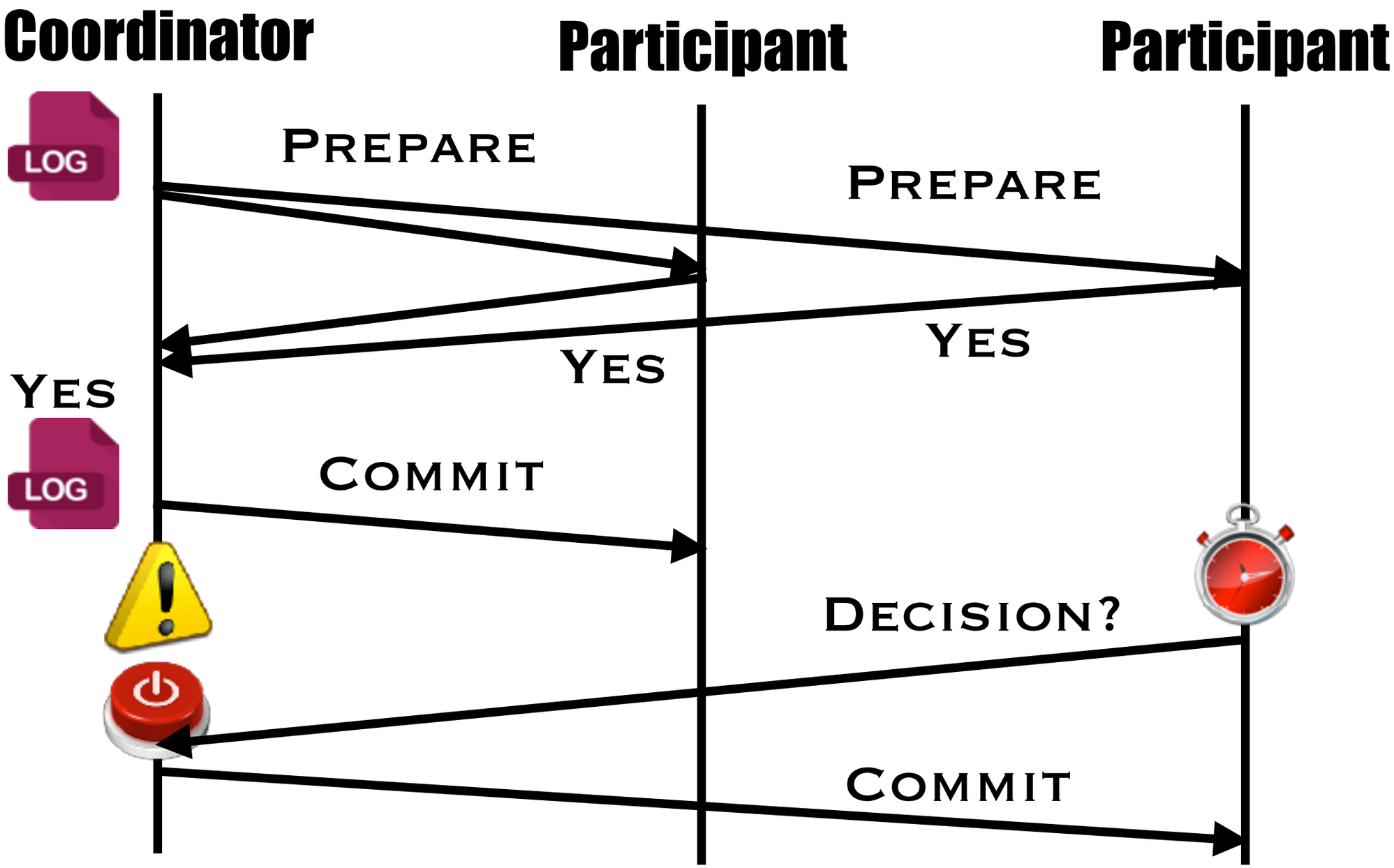
Coordinator failures: After sending prepare



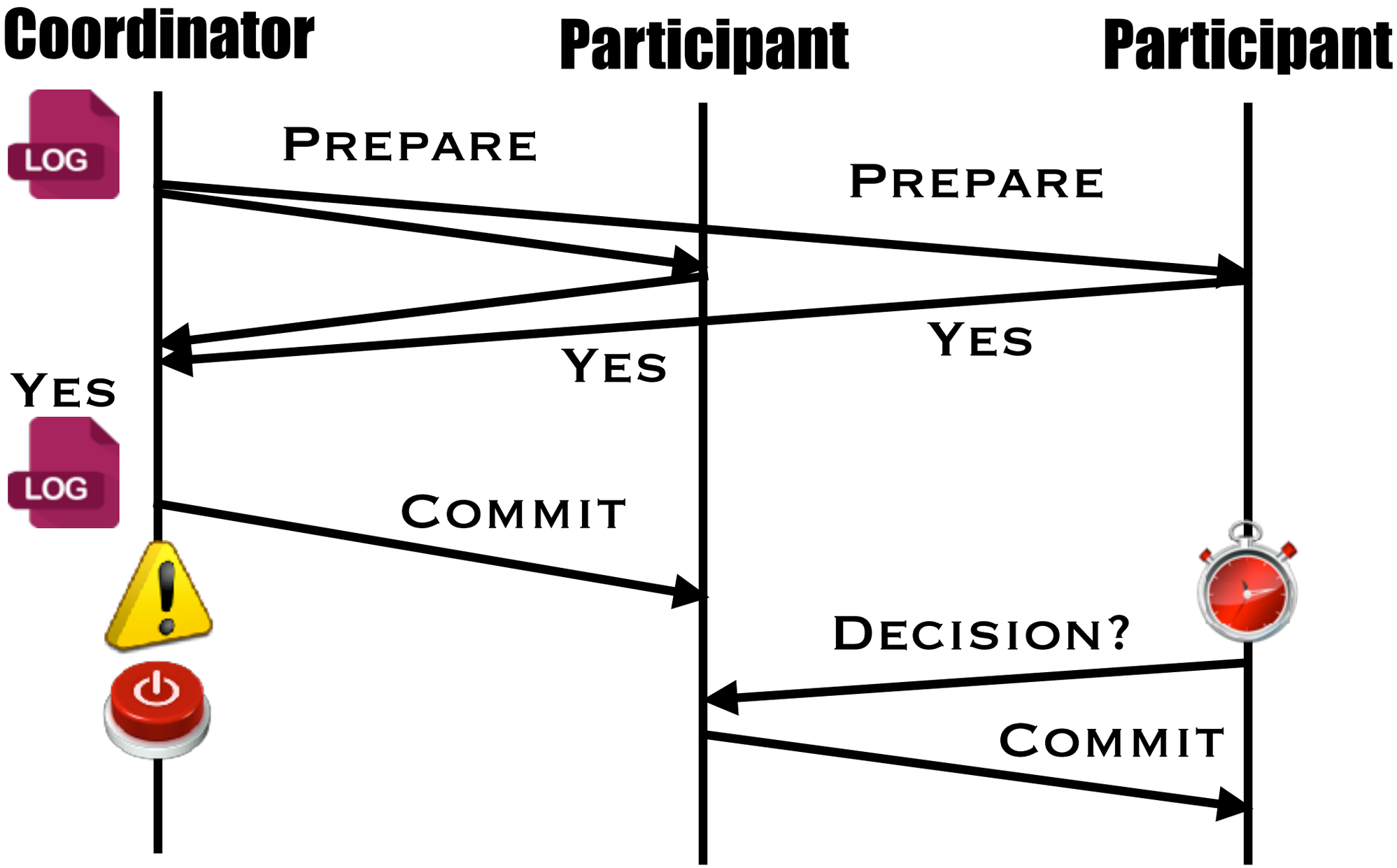
Coordinator failures: After receiving votes



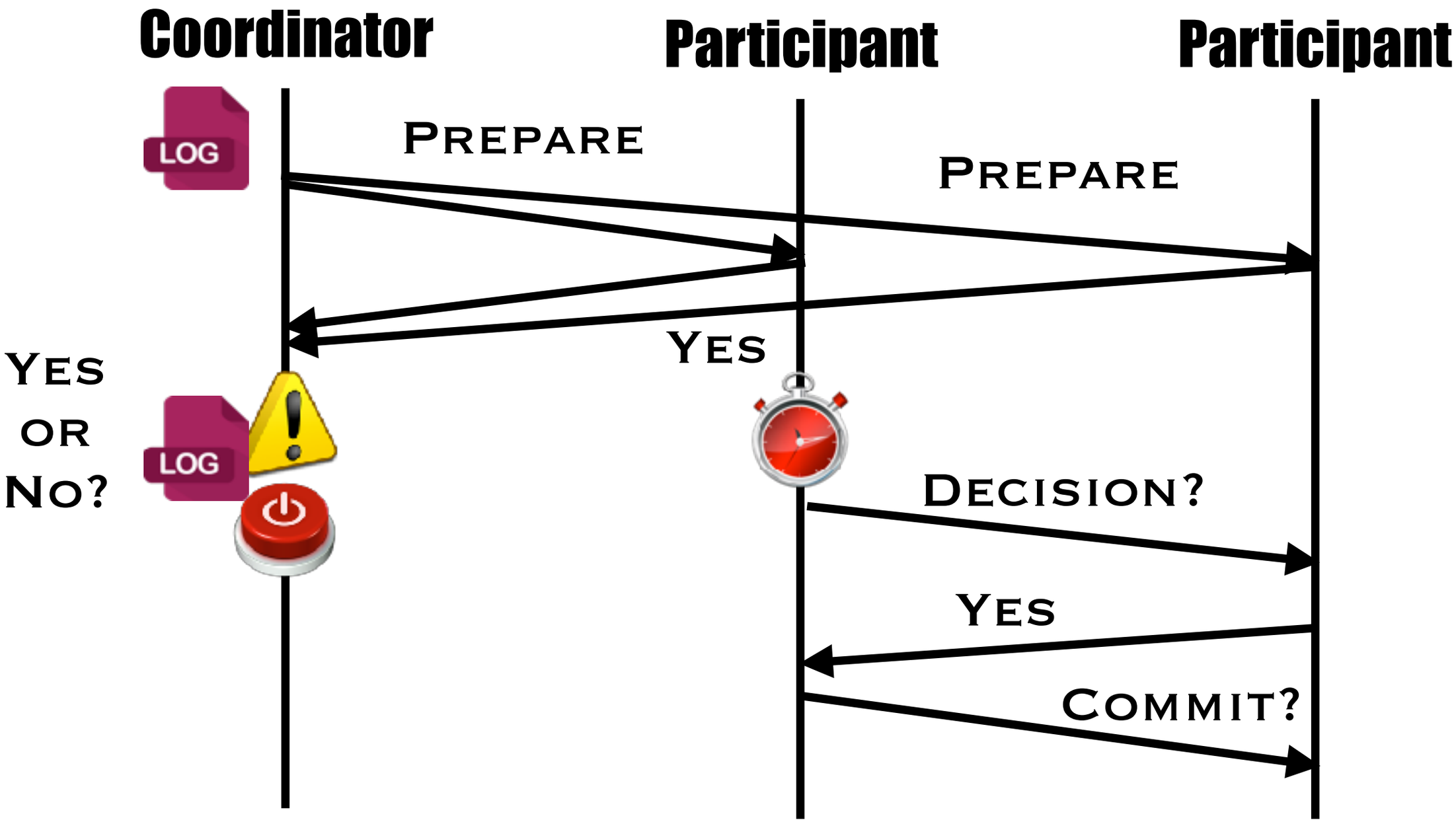
Coordinator failures: After sending decision



Do we need the coordinator?



What if we do not have the coordinator's decision?



2PC vs. SMR

Is 2PC a state machine replication protocol?

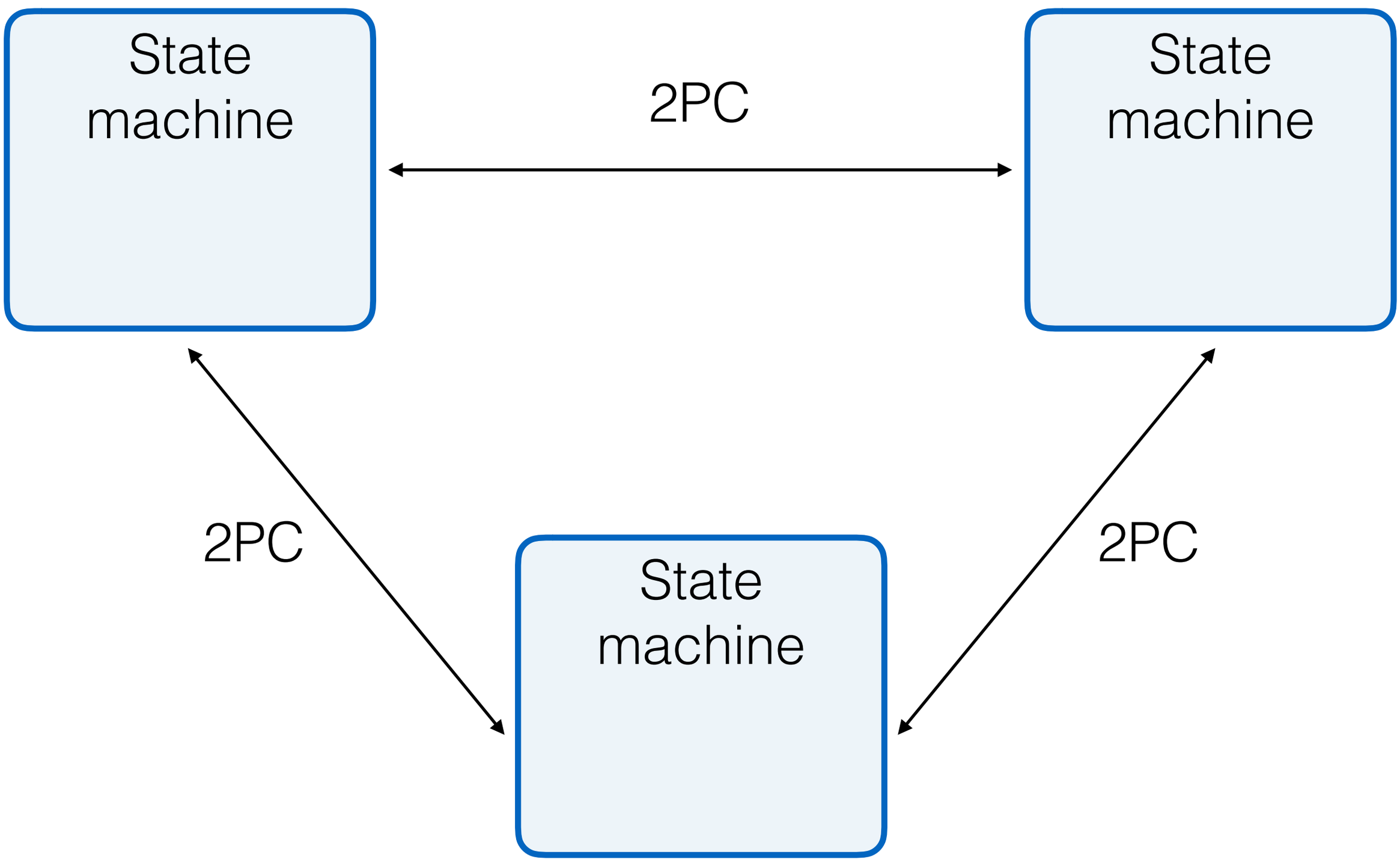
2PC vs. SMR

Is 2PC a state machine replication protocol?

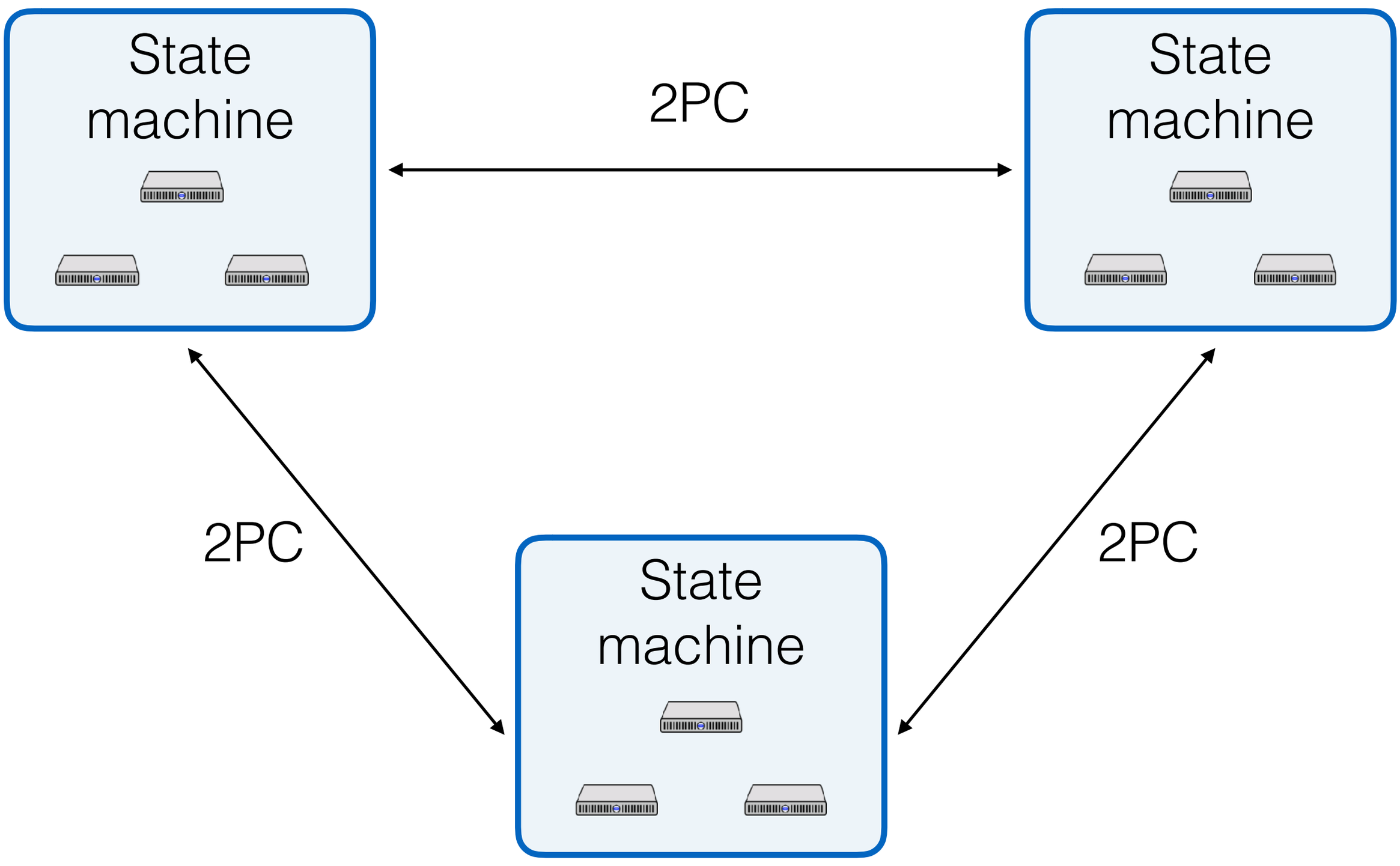
- Could be—but isn't a very good one!

2PC is for coordinating an operation between multiple nodes, where *every* node has to agree in order for the system to make progress!

Typical architecture



Typical architecture



Two generals

Does 2PC solve Two Generals?

Why not?