Two-phase commit

Tom Anderson and Doug Woos

Two Generals Problem

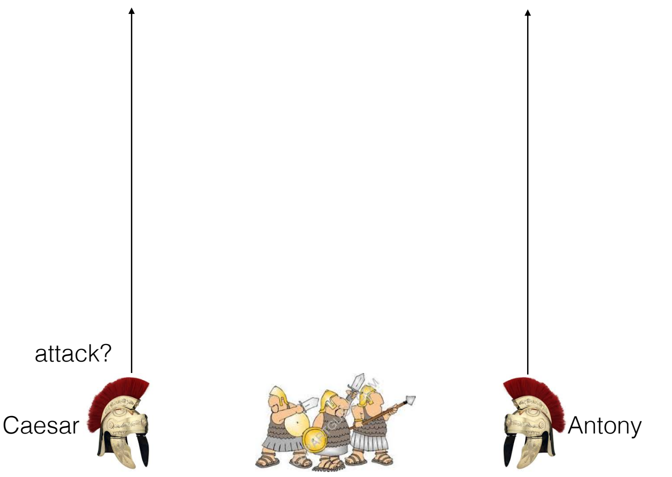
Two generals want to coordinate a time to attack

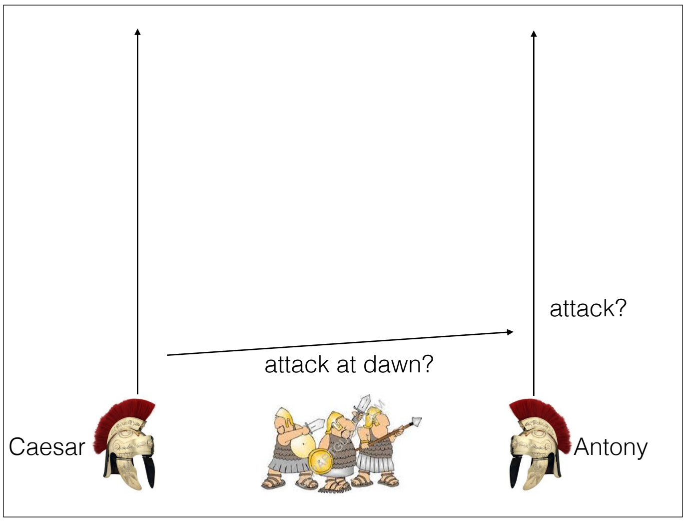
Messengers can be killed, arbitrarily delayed

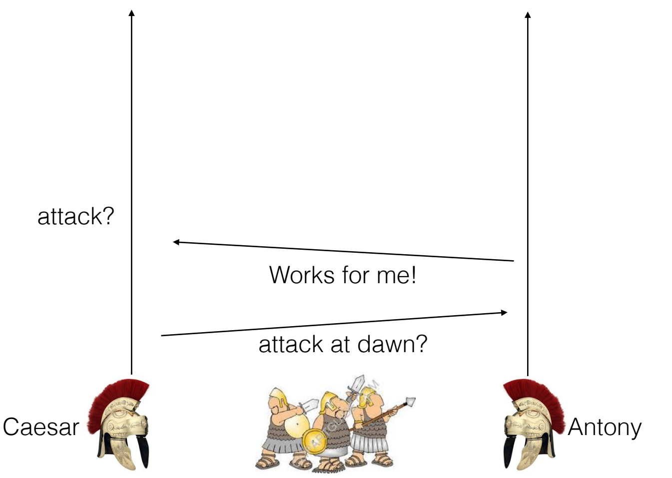
No other communication

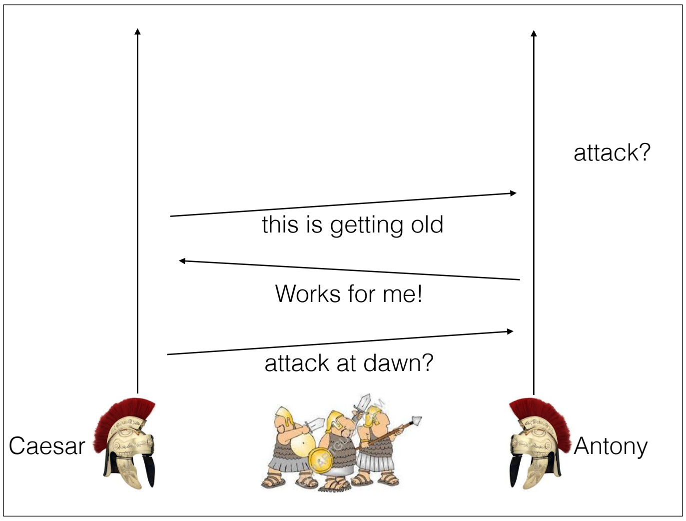
If either attacks alone, army will be destroyed

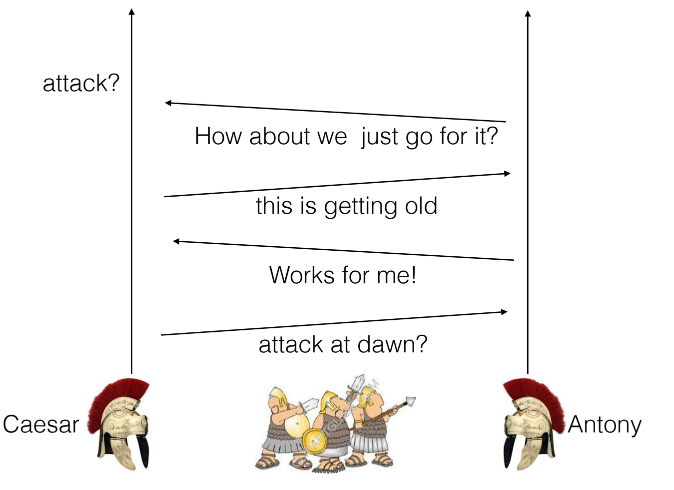
Design a protocol to coordinate an attack











Fisher Lynch Paterson (FLP)

Impossible to reach consensus in an asynchronous distributed system with unreliable messages

Even when all the messages are delivered!

- Provided we don't know if they are delivered

Implies the "CAP" theorem

- Cannot have both availability and consistency
- Have to choose one!

Two Phase Commit

If we can't reach consensus, what can we do?

Central coordinator decides, tells everyone else

- One phase commit
- What if some participants can't do the request?

Two phase commit:

- Central coordinator asks
- Participants commit to commit
- Central coordinator decides, tells everyone else

Two Phase Commit Setting

Atomic read/update to multiple pieces of data, potentially stored in multiple locations

- Account transfer between banks
- Multikey update to a sharded key-value store, e.g., with local locks

Note: two phase locking, write ahead logging are related but different concepts

Calendar event creation

Doug has three advisors (Tom, Zach, Mike)

Want to schedule a meeting with all of them

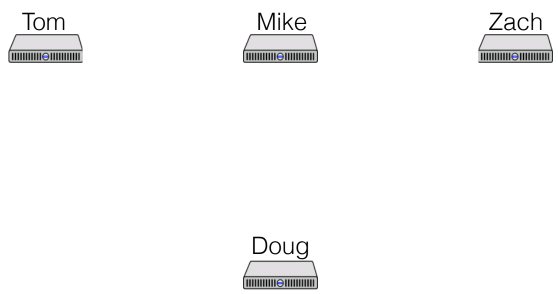
- Let's try Tues at 11, people are usually free then

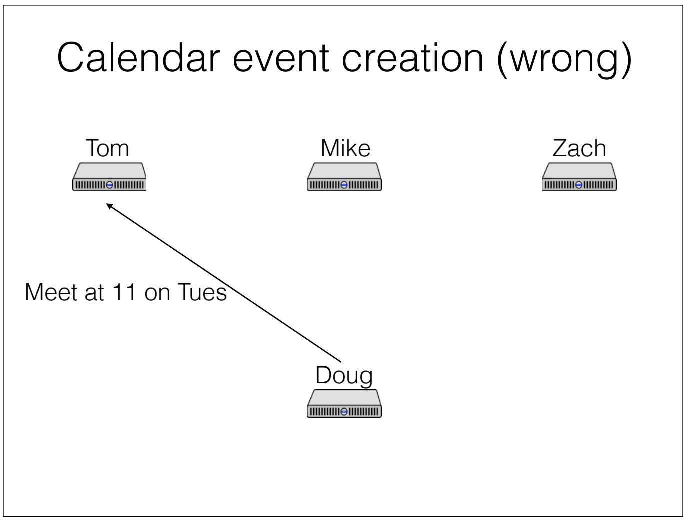
Calendars all live on different nodes!

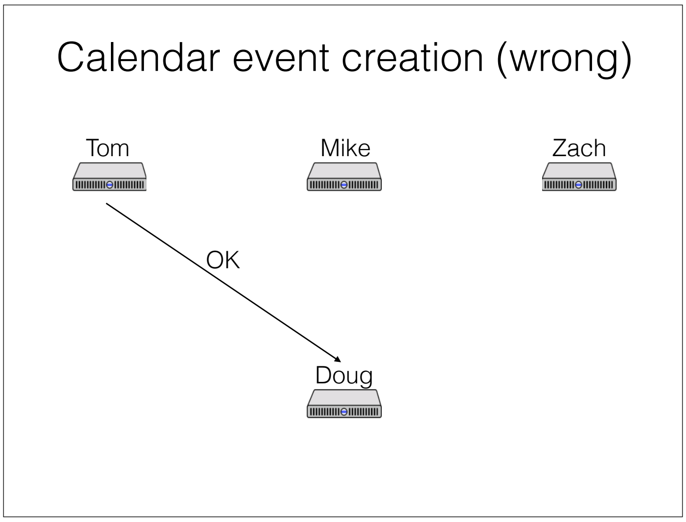
Other students also trying to schedule meetings

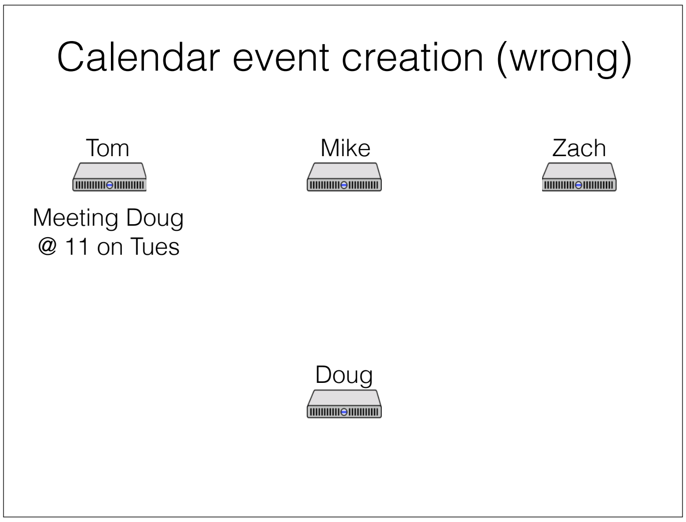
Nodes can fail, messages can be dropped (of course)

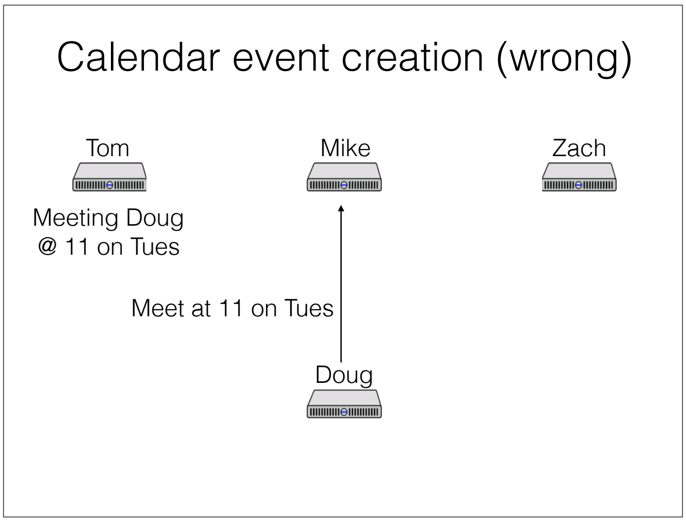
Calendar event creation (wrong)

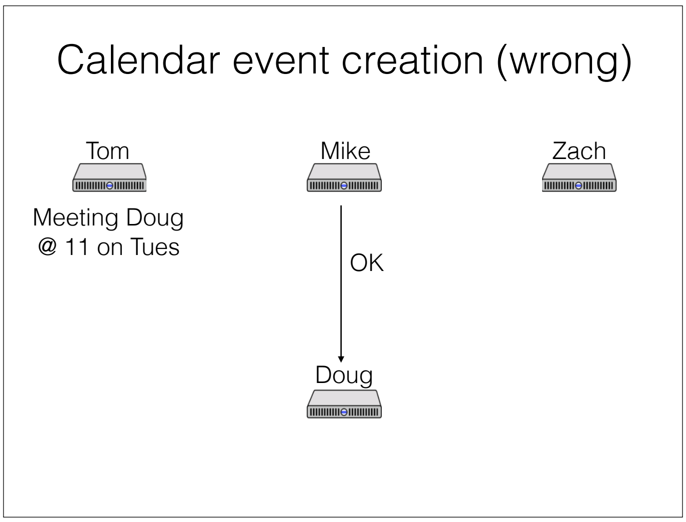


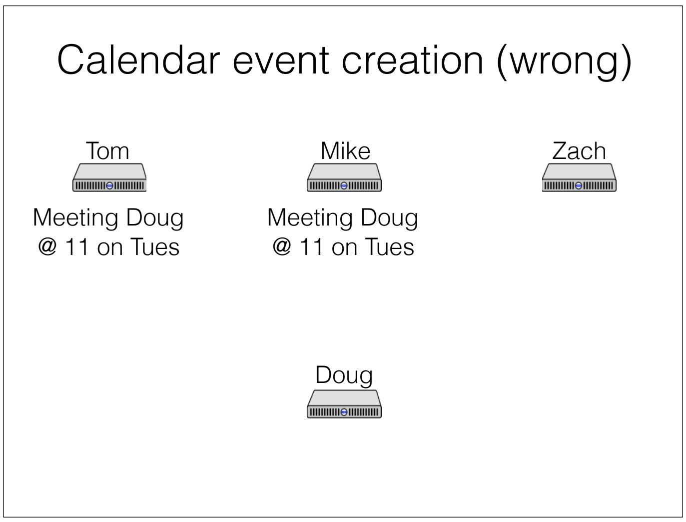

Calendar event creation (wrong)
Tom
Mike
Zach
Meet at 11 on Tues
Doug

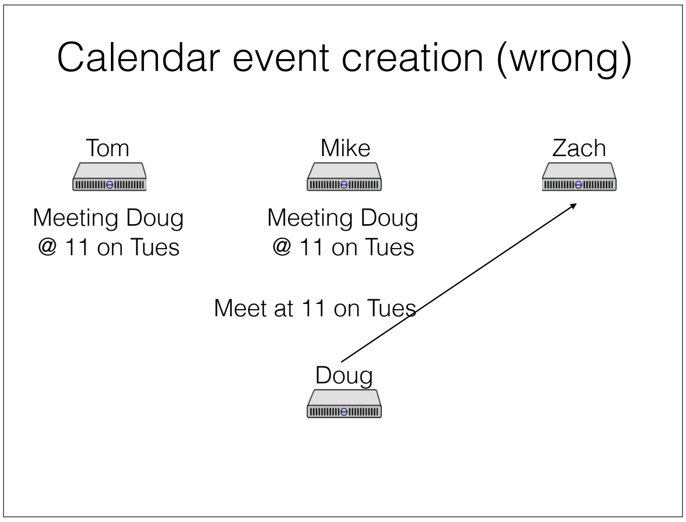

Calendar event creation (wrong)
Tom
Mike
Zach
OK
Doug

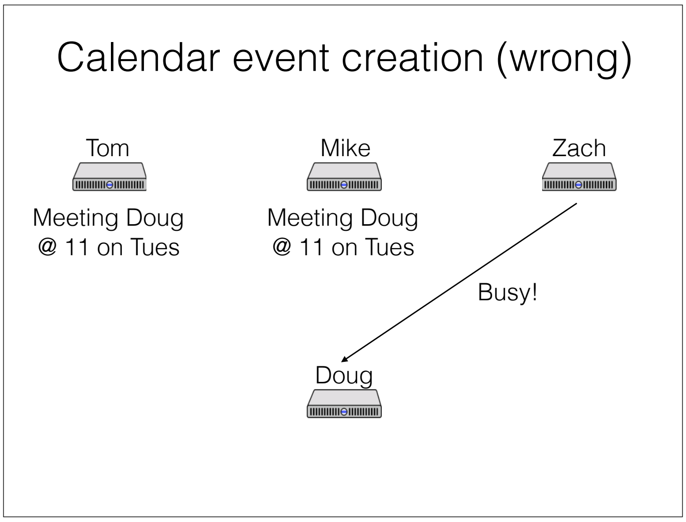

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
Doug

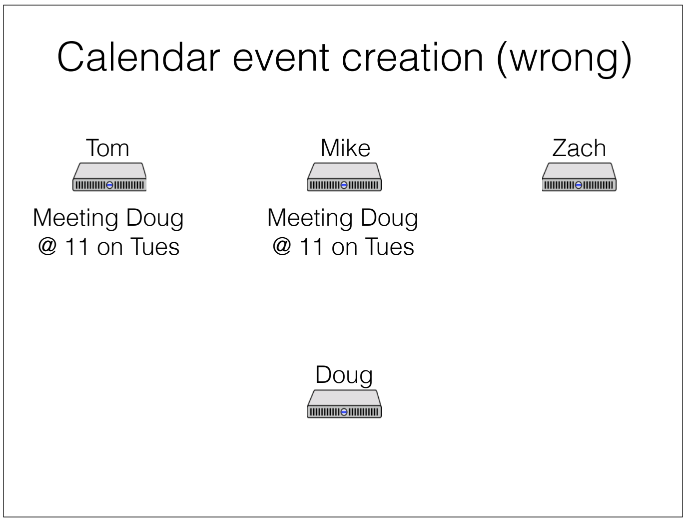

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
Meet at 11 on Tues
Doug

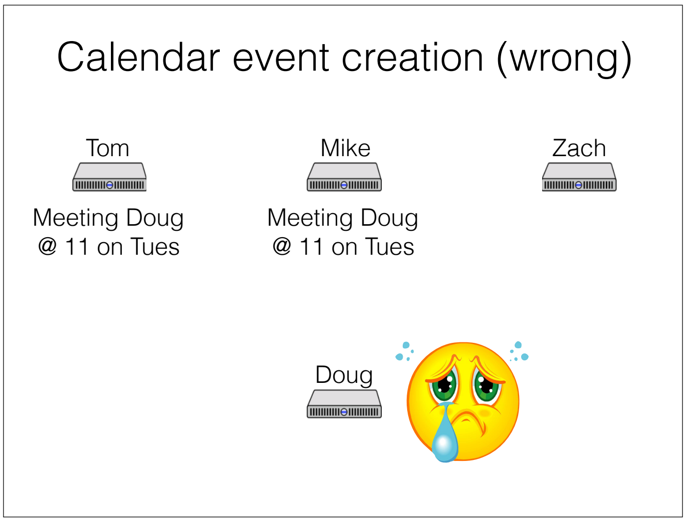

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
OK
Doug

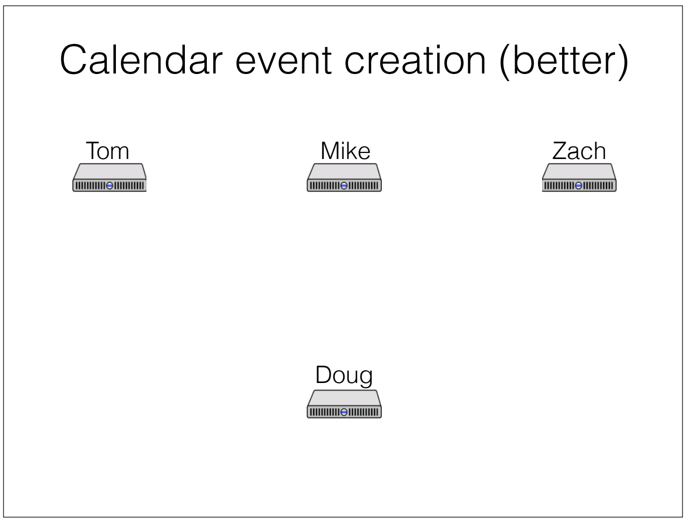

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
Meeting Doug
@ 11 on Tues
Doug

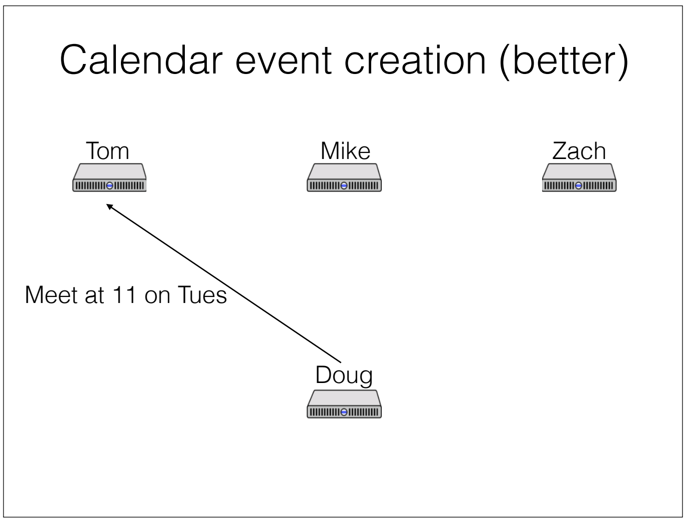

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
Meeting Doug
@ 11 on Tues
Meet at 11 on Tues
Doug

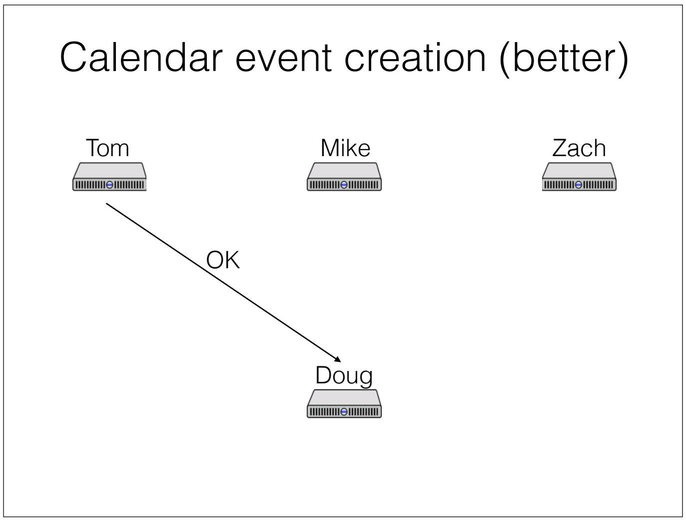

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
Meeting Doug
@ 11 on Tues
Busy!
Doug

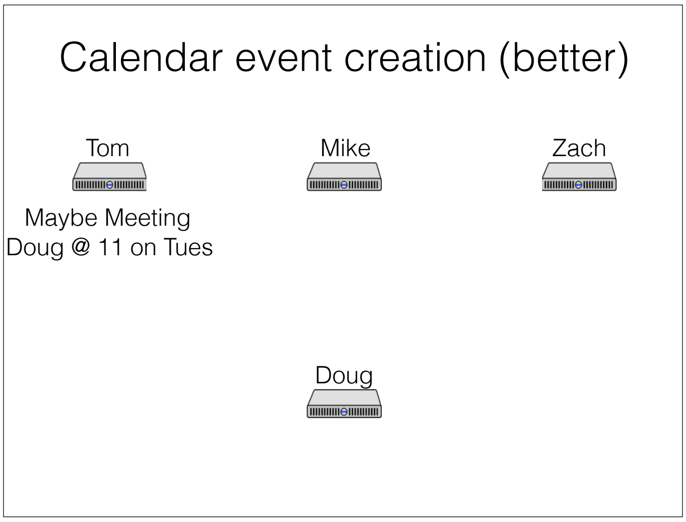

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
Meeting Doug
@ 11 on Tues
Doug

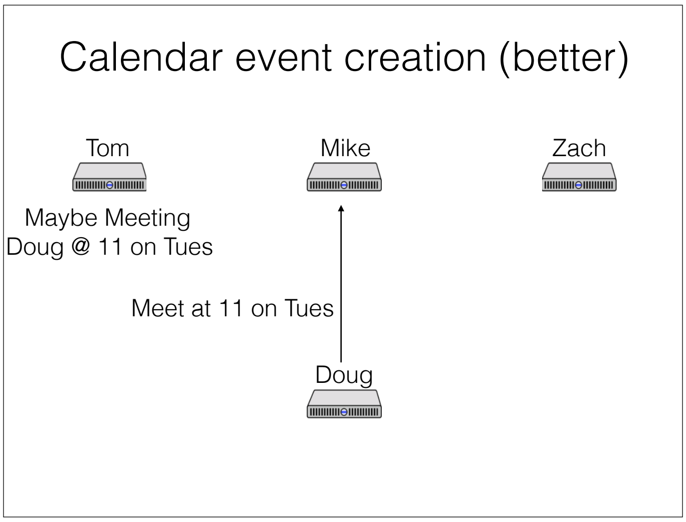

Calendar event creation (wrong)
Tom
Mike
Zach
Meeting Doug
@ 11 on Tues
Meeting Doug
@ 11 on Tues
Doug

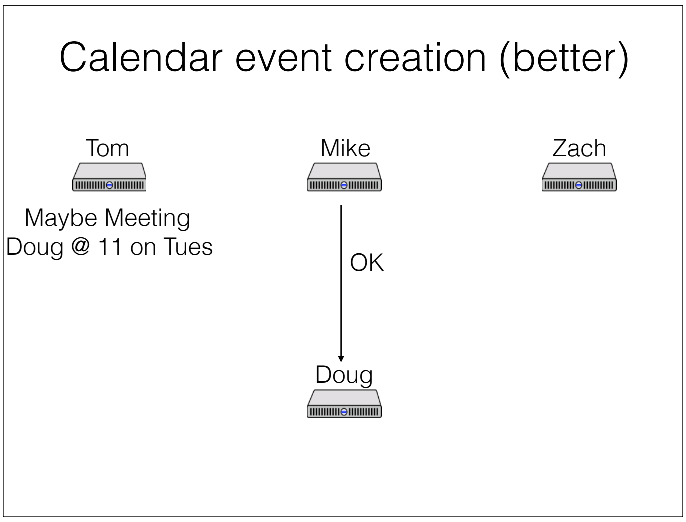

Calendar event creation (better)
Tom
Mike
Zach
Doug

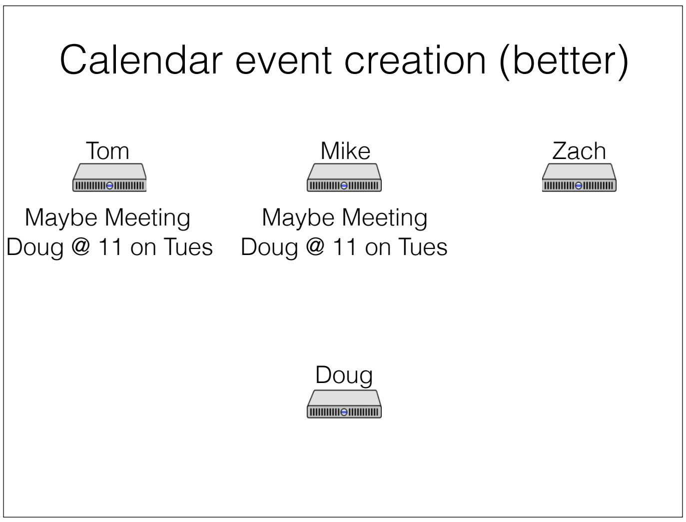

Calendar event creation (better)
Tom
Mike
Zach
Meet at 11 on Tues
Doug

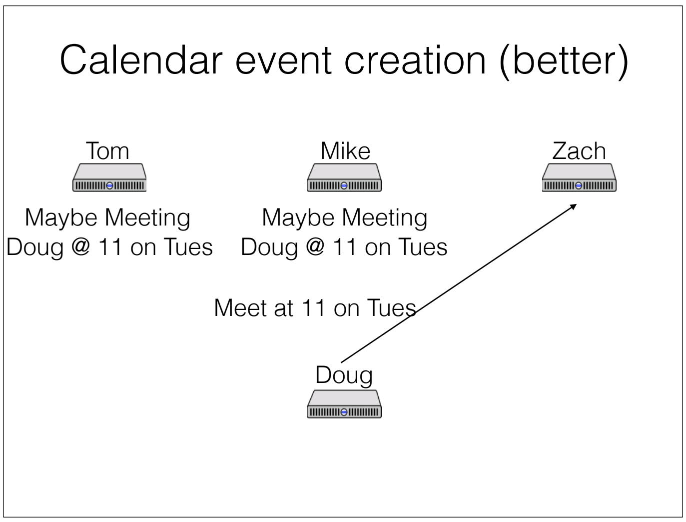

Calendar event creation (better)
Tom
Mike
Zach
OK
Doug


Calendar event creation (better)
Tom
Mike
Zach
Maybe Meeting
Doug @ 11 on Tues
Doug

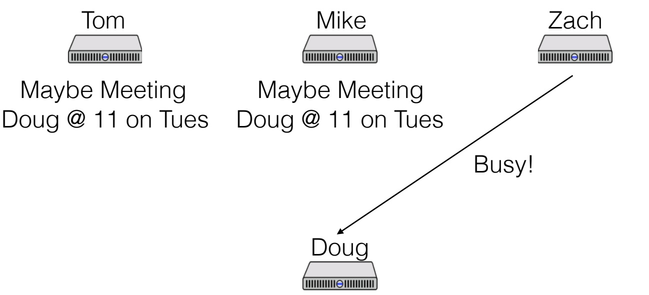

Calendar event creation (better)
Tom
Mike
Zach
Maybe Meeting
Doug @ 11 on Tues
Meet at 11 on Tues
Doug


Calendar event creation (better)
Tom
Mike
Zach
Maybe Meeting
Doug @ 11 on Tues
OK
Doug

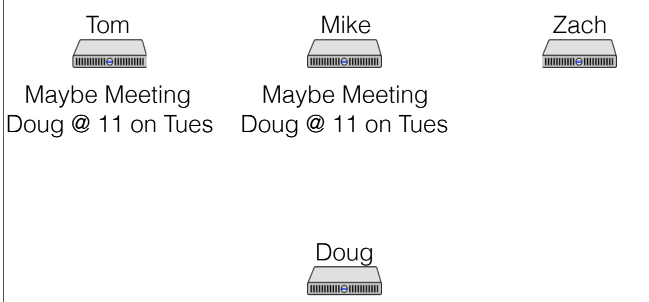

Calendar event creation (better)
Tom
Mike
Zach
Maybe Meeting
Doug @ 11 on Tues
Maybe Meeting
Doug @ 11 on Tues
Doug


Calendar event creation (better)
Tom
Mike
Zach
Maybe Meeting
Doug @ 11 on Tues
Maybe Meeting
Doug @ 11 on Tues
Meet at 11 on Tues
Doug

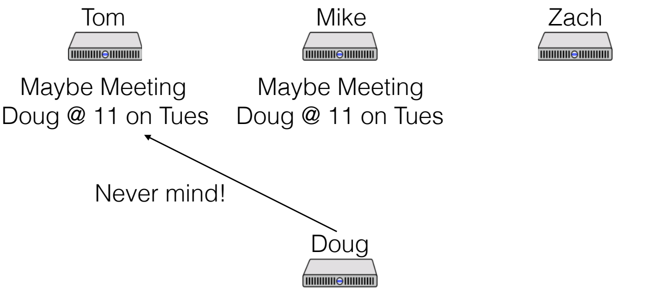
Calendar event creation (better)



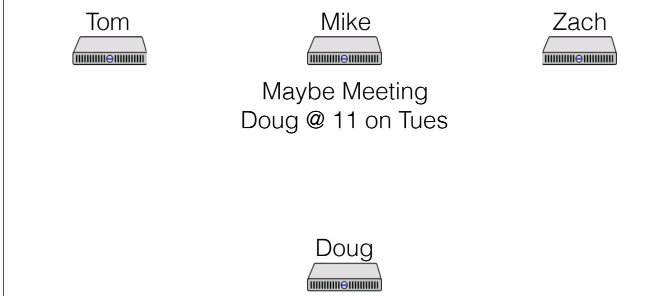
Calendar event creation (better)



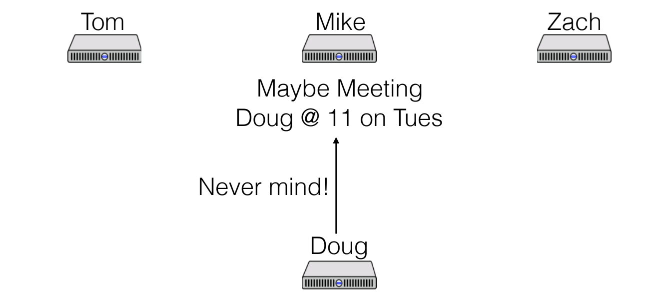
Calendar event creation (better)



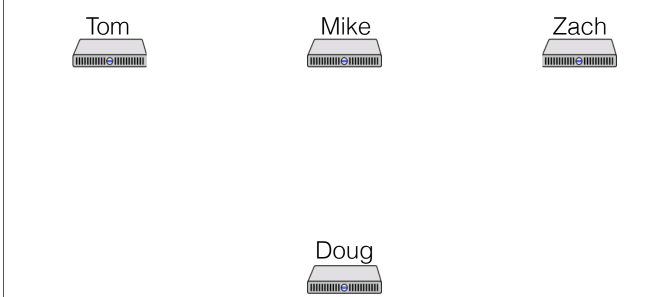
Calendar event creation (better)



Calendar event creation (better)



Calendar event creation (better)



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

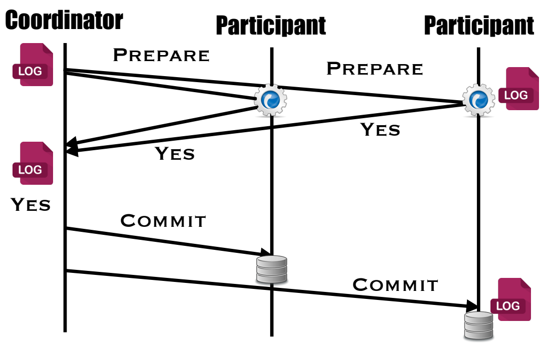
Roles:

- 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)

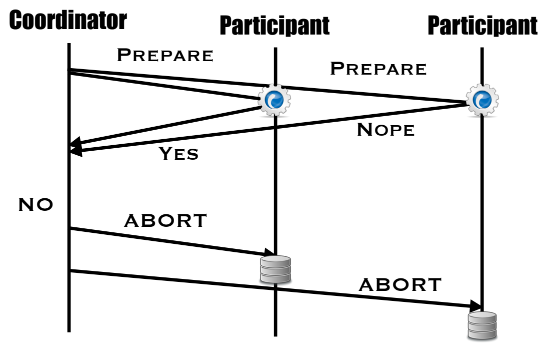
RPCs:

- **Prepare:** Can you commit this transaction?
- **Commit:** Commit this transaction
- **Abort:** Abort this transaction

2PC without failures



2PC without failures



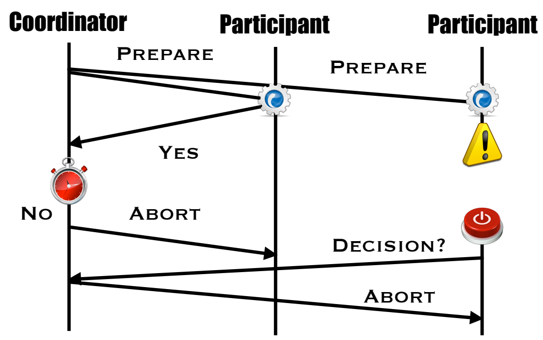
Failures

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

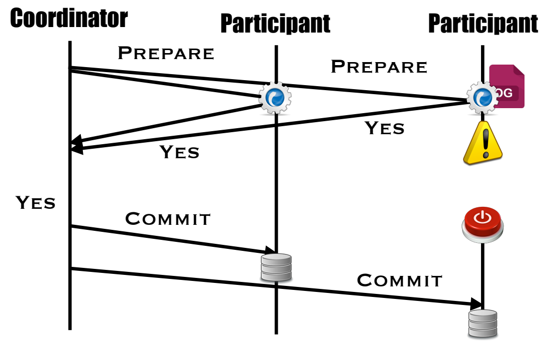
When can interesting failures happen?

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

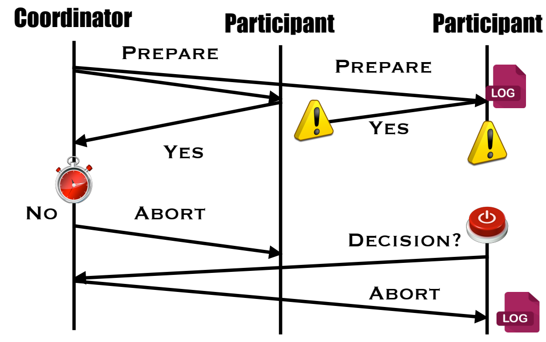
Participant failures: Before sending response?



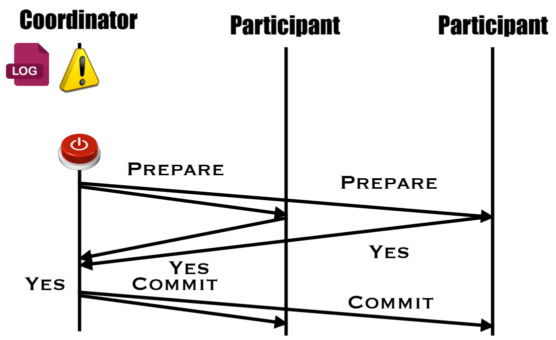
Participant failures: After sending vote?



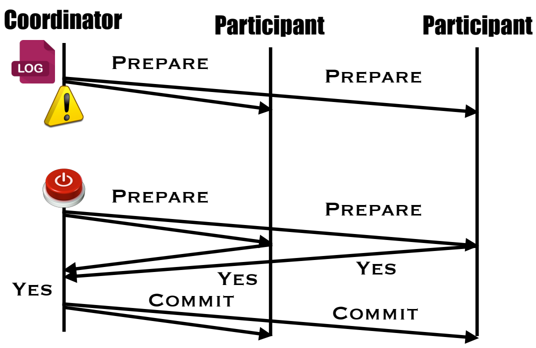
Participant failures: Lost vote?



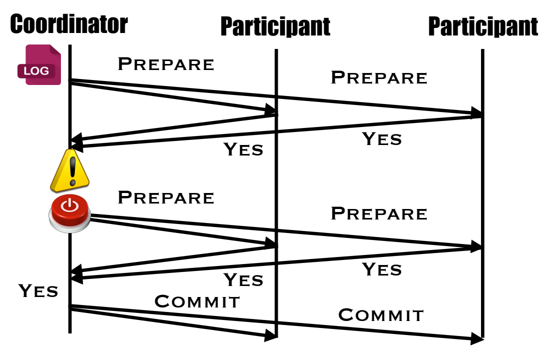
Coodinator failures: Before sending prepare



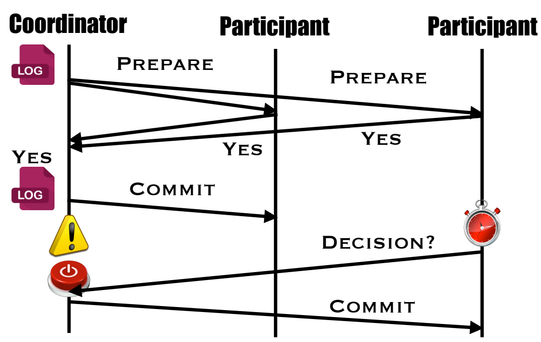
Coordinator failures: After sending prepare



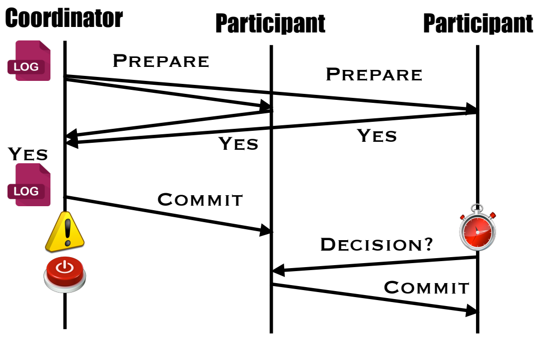
Coordinator failures: After receiving votes



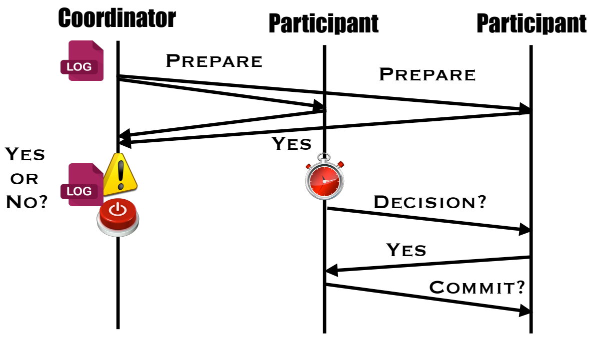
Coordinator failures: After sending decision



Do we need the coordinator?



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



2PC is a *blocking* protocol

- A blocking protocol is one that cannot make progress if some of the participants are unavailable (either down or partitioned).
- It has fault-tolerance but not *availability*.
- This limitation is fundamental (2 generals problem).