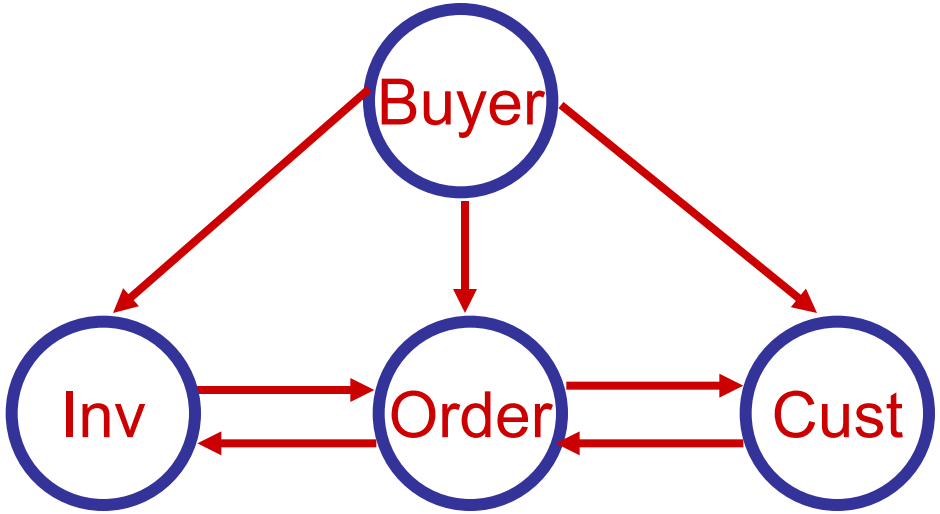
Transactions
2PC in other topologies

Intel (TX memory): Transactional Synchronization Extensions (TSX)

Trans in Distributed Systems

- A distributed transaction involves
  - updates at multiple nodes
  - and the messages between those nodes
- For example, buying widgets

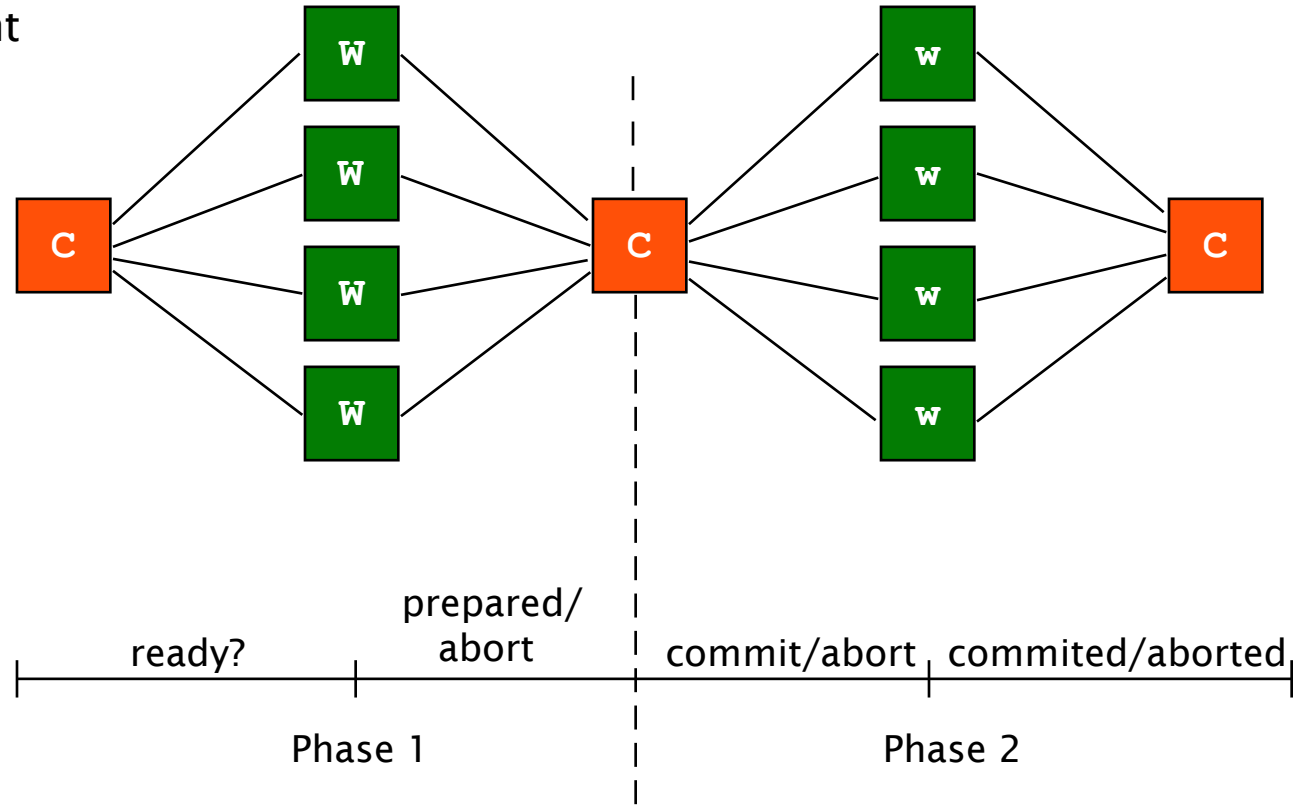


Distributed Atomic Commit Requirements

1. All workers that reach a decision reach the same one
2. Workers cannot change their decisions on commit or abort once a decision is made
3. To commit all workers must vote commit
4. If all workers vote commit and there are no failures the transaction will commit
5. If all failures are repaired and there are no more failures each worker will eventually reach a decision (In fact it will be the same decision)

2PC and communication topologies

- We have previously focused on centralized 2PC
  - Why funnel messages through the coordinator?
  - + None of the worker nodes can influence one another
  - + Failure of a worker node independent
  - - Put trust in coordinator
  - - Hope coordinator does not fail



2PC and communication topologies

- We have previously focused on centralized 2PC
  - Why funnel messages through the coordinator?
  - + None of the worker nodes can influence one another
  - + Failure of a worker node independent
  - - Put trust in coordinator
  - - Hope coordinator does not fail

- Nothing stopping us from considering alternative communication topologies for 2PC!
- Why? Because other topologies may reduce time or message complexity for the basic 2PC protocol
  - Time/latency ~ rounds used by a protocol
  - Bandwidth ~ messages used by a protocol

2PC in other topologies

- Two extremes: linear and decentralized

- Linear 2PC: coordinator, and all workers in a single line/chain
  * Build a protocol that has fewer messages (but more rounds!) than 2PC
  * C – W1 – W2 – W3 … – Wn

- Decentralized 2PC: all workers can communicate with one another
  * Build a protocol that has fewer rounds (but more messages!) than 2PC

2PC in other topologies

- Two extremes: linear and decentralized
- Linear 2PC: coordinator, and all workers in a single line/chain
  * Build a protocol that has fewer messages (but more rounds!) than 2PC
  * C – W1 – W2 – W3 … – Wn
- Decentralized 2PC: all workers can communicate with one another
  * Build a protocol that has fewer rounds (but more messages!) than 2PC

Linear 2PC

- Alternative communication topologies in 2PC context
- Linear 2PC: coordinator, and all workers in a single line/chain
  - C, W1, W2, W3, … Wn
  - Build a protocol that has fewer messages (but more rounds!) than 2PC
  - C sends request + its vote to W1, W1 decided commit/abort, forward decision to W2. W2, determines outcome with its own decision, forward to W3, and so on.
  - Wn receives commit and decided commit → tx commit! Forward this decision back to front of chain
  - Wn receives abort/decides abort -> tx abort! Forward this decision back
- Note: linear 2PC bundles node/site failure with communication failure
  - A kind of fate sharing (node failure takes down commucation with it)

Linear 2PC

- Important note: linear 2PC bundles node/site failure with communication failure.
- Why is this important?

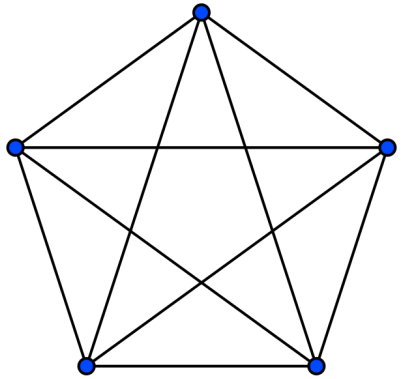
- Analysis for linear 2PC:
  * 2n rounds
  * 2n messages

2PC in other topologies

- Two extremes: linear and decentralized

- Linear 2PC: coordinator, and all workers in a single line/chain
  * Build a protocol that has fewer messages (but more rounds!) than 2PC
  * C – W1 – W2 – W3 … – Wn

- Decentralized 2PC: all workers can communicate with one another
  * Build a protocol that has fewer rounds (but more messages!) than 2PC



Decentralized 2PC

- Alternative communication topologies in 2PC context
- Decentralized 2PC: all workers can communicate with one another
  - Build a protocol that has fewer rounds (but more messages!) than 2PC
  - Complete graph communication topology
  - Coordinate votes and sends it's decision (commit/abort) along with prepareToCommit to workers
  - Workers broadcast their choice to all other workers (n^2 messages!)
  - Workers collect votes, and figure out the final transaction outcome
- 2 rounds -- Can we do better than 2 rounds?
- Approx: n+(n-1)*n messages (n=number of nodes)

Are they still susceptible to blocking?

- Centralized 2PC blocks if coordinator fails after receiving all votes and before sending decision.

- What about linear 2PC and decentral. 2PC?
  * Linear 2PC: coordinator, and all workers in a single line/chain
  * Decentralized 2PC: all workers can communicate with one another

Are they still susceptible to blocking?

- Centralized 2PC blocks if coordinator fails after receiving all votes and before sending decision.

- What about linear 2PC and decentral. 2PC?
  - Yes, both are blocking protocols!
- Linear 2PC: coordinator, and all workers in a single line/chain
  - Blocks if last node in the chain fails (outcome indeterminate)

- Decentralized 2PC: all workers can communicate with one another
  - Blocks if any node fails (or msg does not arrive: not enough information)

Comparison in one slide

	Messages	Rounds
Centralized 2PC	3n	3
Linear 2PC	2n	2n
Decentralized 2PC	n^2	2

Broader take-aways

- Most algorithms (not just 2PC!) presented for one topology, can be converted to use a different topology
- Topology matters, particularly for performance: rounds and communication complexity
- Topology matters (a lot) for failures
- Topology rarely changes fundamental properties of the algorithm, such as blocking in 2PC