CSE 486/586 Distributed Systems
Byzantine Fault Tolerance --- 2

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

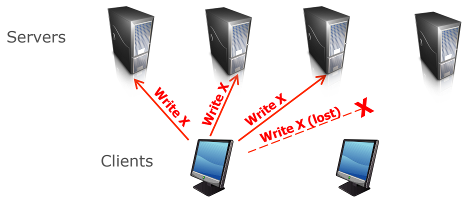
- Fault categories
  - Benign
  - Byzantine
- Consensus results
  - Paxos: *f* (benign) faulty nodes → *2f + 1* total nodes
  - BFT: *f* (Byzantine) faulty nodes → *3f + 1* total nodes
- Byzantine generals problem
  - A commanding general & *N - 1* lieutenant generals
  - All loyal lieutenants obey the same order.
  - If the commanding general is loyal, then every loyal lieutenant obeys the order the commanding general sends.

Practical Byzantine Fault Tolerance

- Byzantine fault tolerance (BFT) protocols thought to be too expensive and impractical.
- PBFT (Practical BFT) was then proposed, which showed a rather inexpensive & practical BFT protocol.
  - With asynchrony & *f* Byzantine nodes
  - This resurrected the interest in BFT protocols.
- PBFT is designed for replicated state machines

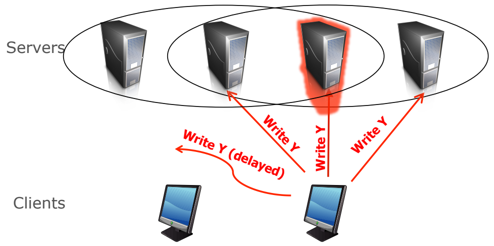
3f+1 for Replicated State Machines

- For liveness, we need to assume that we might only get N-f. We say that this N-f is our quorum size.



3f+1 for Replicated State Machines

- For correctness, any two quorums must intersect at least one honest node.
  - (N-f) + (N-f) - N >= f+1 → N >= 3f+1



PBFT

- A BFT protocol for primary-backup
- It is optimal, i.e., operates with 3f+1 nodes.
- Deal with two things (recall from last lecture)
  - Malicious primary
  - Consensus
- Everyone uses authentication to verify who they're talking with.
- How it works
  - Primary performs operations
  - Backups monitor the primary and do a view change if they detect a primary failure.

System Setting

- Each replica has an id *i* (between 0 and N-1)
- A view number *v* identifies the current primary.
  - Current primary: i = v mod N
  - If the current primary fails, the next primary is (i + 1) mod N
- Each client request has a sequence number
- All messages are authenticated using crypto-based techniques.
  - Anyone can verify who sent the message & if the message content is correct.
  - Using public-key signatures, message authentication codes, and message digests
  - Forgery is practically not possible, limiting what a faulty node can do.

Client Protocol

- A client sends a signed request to the primary.
  - The primary can still lie (later).
- All replicas reply directly to the client.
- The client waits until it receives *f + 1* replies with the same result.
- The client accepts the result.
- If the client doesn't receive replies soon enough, it multicasts the request to all replicas.

Primary-Backup Protocol

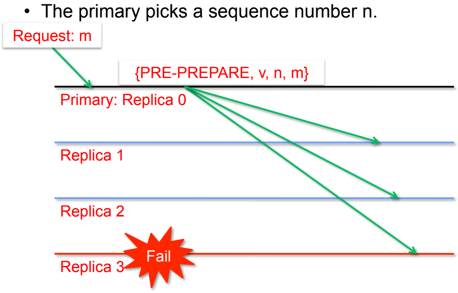
- Normal case operation
  - Three phases: Pre-prepare, prepare, commit
  - A sequence number for each operation, which is agreed and verified by all replicas to detect malicious primary
- View changes
  - When the primary fails

Normal Case Operation

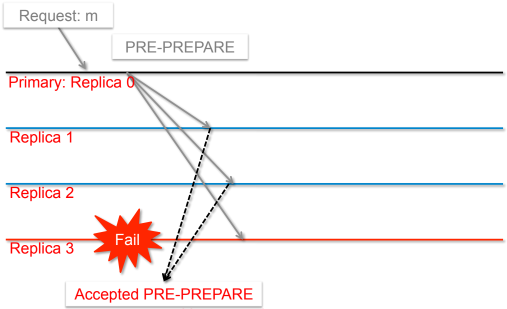
- Three phases
  - PRE-PREPARE picks order of requests
  - PREPARE ensures order within views
  - COMMIT ensures order across views
- Replicas remember messages in log
- Messages are authenticated
- The primary can still lie.
  - Send different sequence number for the same operation to different replicas
  - Use a duplicate sequence number for operation

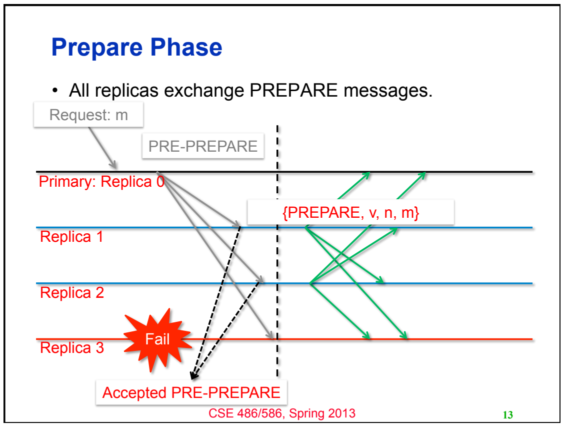
Pre-Prepare Phase

- The primary picks a sequence number n.



Prepare Phase





Prepare Phase

- All replicas exchange PREPARE messages.

Request: m

PRE-PREPARE

Primary: Replica 0

{PREPARE, v, n, m}

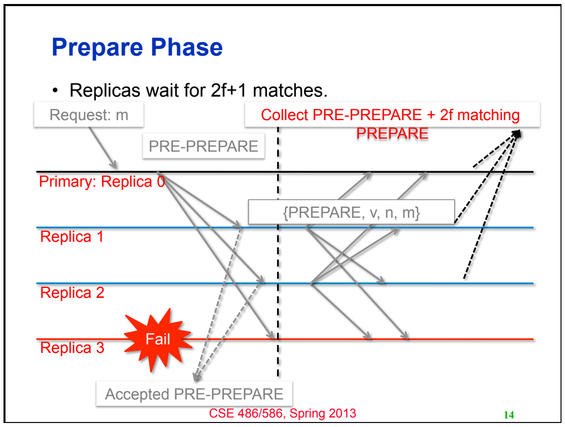
Replica 1

Replica 2

Fail

Replica 3

Accepted PRE-PREPARE



Prepare Phase

- Replicas wait for 2f+1 matches.

Request: m

Collect PRE-PREPARE + 2f matching PREPARE

PRE-PREPARE

Primary: Replica 0

{PREPARE, v, n, m}

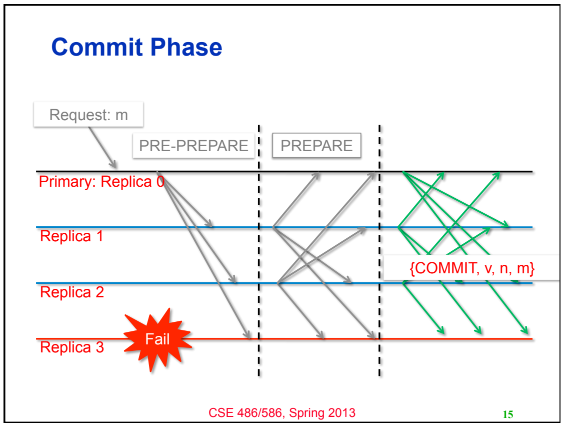
Replica 1

Replica 2

Fail

Replica 3

Accepted PRE-PREPARE



Commit Phase

Request: m

PRE-PREPARE

PREPARE

Primary: Replica 0

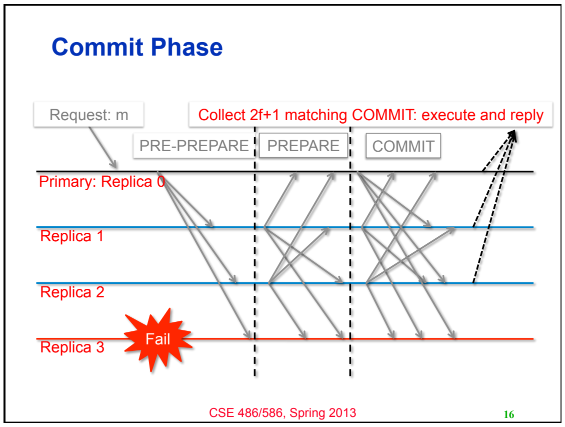
Replica 1

{COMMIT, v, n, m}

Replica 2

Fail

Replica 3



Commit Phase

Request: m

Collect 2f+1 matching COMMIT: execute and reply

PRE-PREPARE

PREPARE

COMMIT

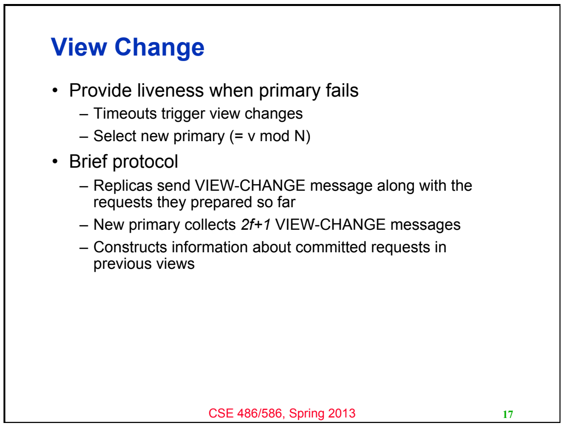
Primary: Replica 0

Replica 1

Replica 2

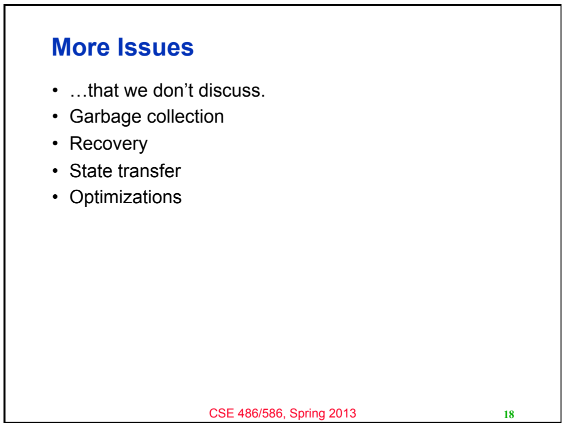
Fail

Replica 3



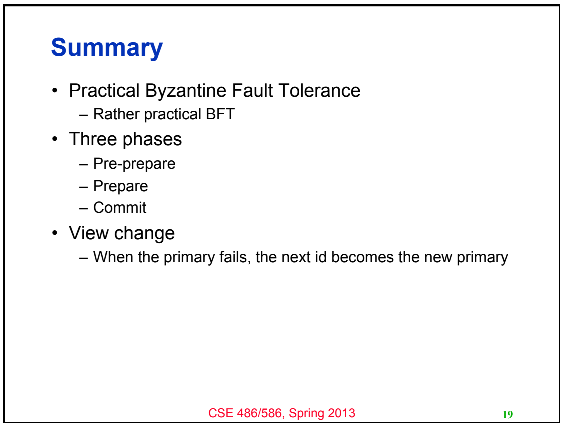
View Change

- Provide liveness when primary fails
  - Timeouts trigger view changes
  - Select new primary (= v mod N)
- Brief protocol
  - Replicas send VIEW-CHANGE message along with the requests they prepared so far
  - New primary collects *2f+1* VIEW-CHANGE messages
  - Constructs information about committed requests in previous views



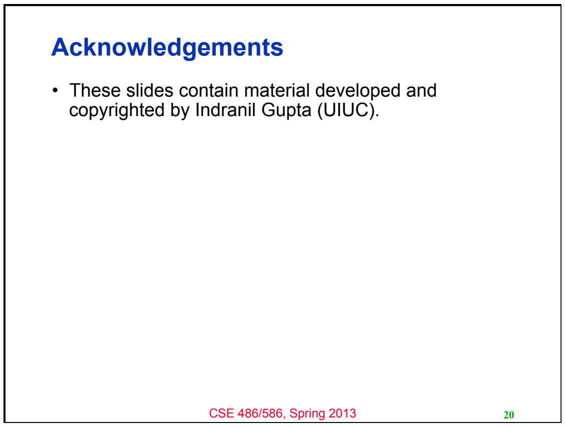
More Issues

- …that we don't discuss.
- Garbage collection
- Recovery
- State transfer
- Optimizations



Summary

- Practical Byzantine Fault Tolerance
  - Rather practical BFT
- Three phases
  - Pre-prepare
  - Prepare
  - Commit
- View change
  - When the primary fails, the next id becomes the new primary



Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).