Distributed Systems 1

CUCS Course 4113
https://systems.cs.columbia.edu/ds1-class/

Instructor: Roxana Geambasu

Byzantine Fault Tolerance

So far: Fail-stop failures

- Machine crashes, network breaks, partitions.
- Nodes are always “aware” of their own crashes, and execute a designated recovery protocol.

- Q: how many replicas are needed to tolerate f simultaneous fail-stop failures in consensus?

So far: Fail-stop failures

- Machine crashes, network breaks, partitions.
- Nodes are always “aware” of their own crashes, and execute a designated recovery protocol.

- Need 2f+1 replicas to tolerate f simultaneous fail-stop failures in consensus.

- Paxos, RAFT are two fault tolerance protocols that work correctly (though don’t guarantee progress) in the context of fail-stop failures.

Byzantine Faults

- Nodes can fail **arbitrarily**, including deviate from the protocol
  - may perform incorrect computation
  - may give conflicting information to different parts of the system
  - may collude with other failed nodes

- Potential causes:
  - software bugs
  - hardware failures
  - malicious attacks

Today: Byzantine Fault Tolerance (BFT)

- Can we provide state machine replication for a service in the presence of Byzantine faults?

- This is just one specific topic related to the broader domain of **security and privacy in distributed systems**!

- Like with everything else, lots more to learn and keep up with, so your advanced (non-foundational) DS learning only **begins** with this class!

Traditional State Machine Replication (e.g., with (multi-)Paxos)

- Requires 2f+1 = 3 replicas if f=1
- Operations are totally ordered => correctness and consistency
- A two-phase protocol (last phase is “just” for catching up partitioned nodes)
- Each phase waits for >= f+1 (i.e., 2 if f=1) of the nodes so you have overlapping quora (**for what purpose?**).

Traditional State Machine Replication (e.g., with (multi-)Paxos)

- Requires 2f+1 = 3 replicas if f=1
- Operations are totally ordered => correctness and consistency
- A two-phase protocol (last phase is “just” for catching up partitioned nodes)
- Each phase waits for >= f+1 (i.e., 2 if f=1) of the nodes so you have overlapping quora (so at least one node remembers a previously passed value).

(Multi-)Paxos for BFT?
What could go wrong?

(Multi-)Paxos for BFT? What could go wrong?

1. In multi-paxos, can't rely on primary to assign seqno
   a. Example?
2. Can't just wait for f+1: the intersection of two majorities may be a lying node.
   a. Example?
3. Many other potential failure scenarios! Think of more yourselves!

Practical Byzantine Fault Tolerance

[Castro & Liskov, 1999]

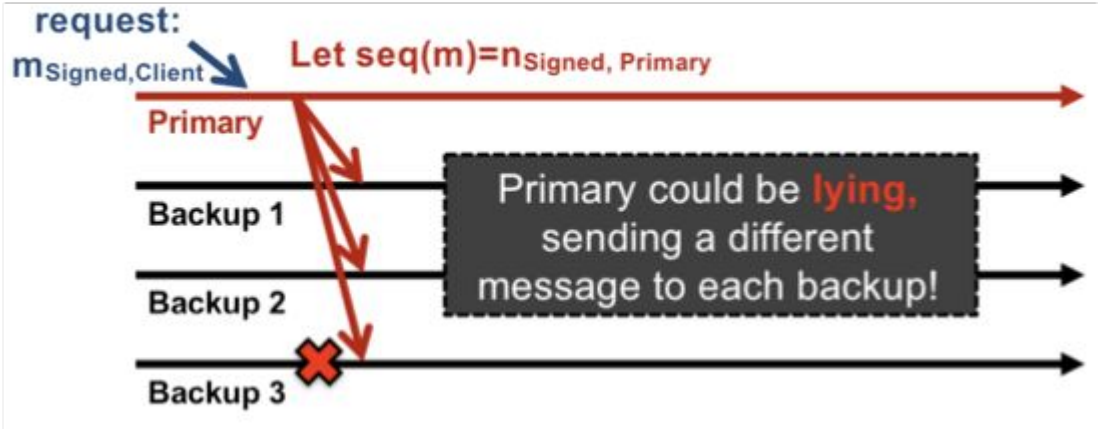
Key differences:

1. **Authentication**: Messages are authenticated (a.k.a., signed) s.t. everyone knows who sent them & can do the accounting of responses correctly.
2. **Super-majority:** Majority = 2f+1 now because you need to know that at least f+1 non-faulty nodes have responded.
    a. Intersection of any 2f+1 super-majorities formed at different times will result in *at least 1 non-faulty node who can "remember" any past agreed upon value.* This is so you can handle both f byzantine faults and potential partitioning.
3. **Broadcast**: Protocol works through broadcast b/c no individual node (such as a "leader") can be trusted.
4. **Three phases**: There are three proper phases in addition to "termination".

BFT Protocol (1/4)

Phase 1: Pre-prepare

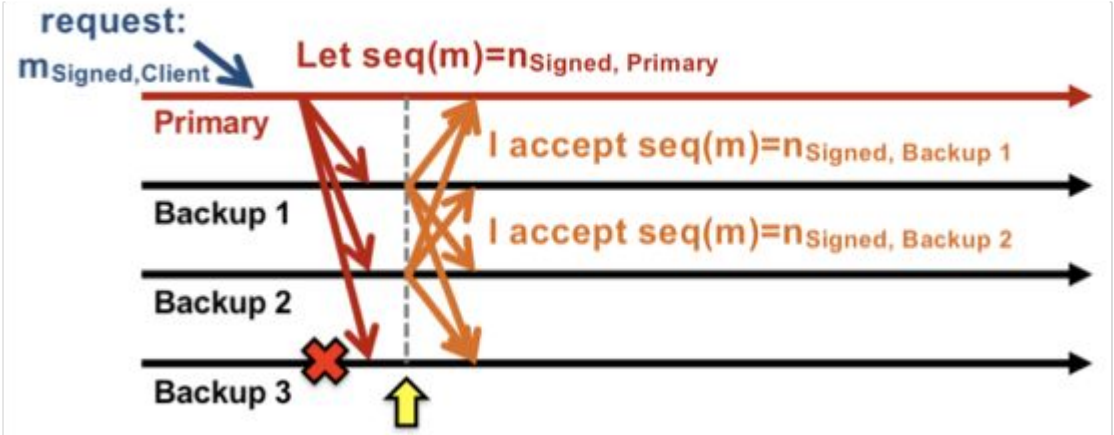
- Primary assigns a sequence number to a client request and forwards it to all the replicas



BFT Protocol (2/4)

Phase 2: Prepare

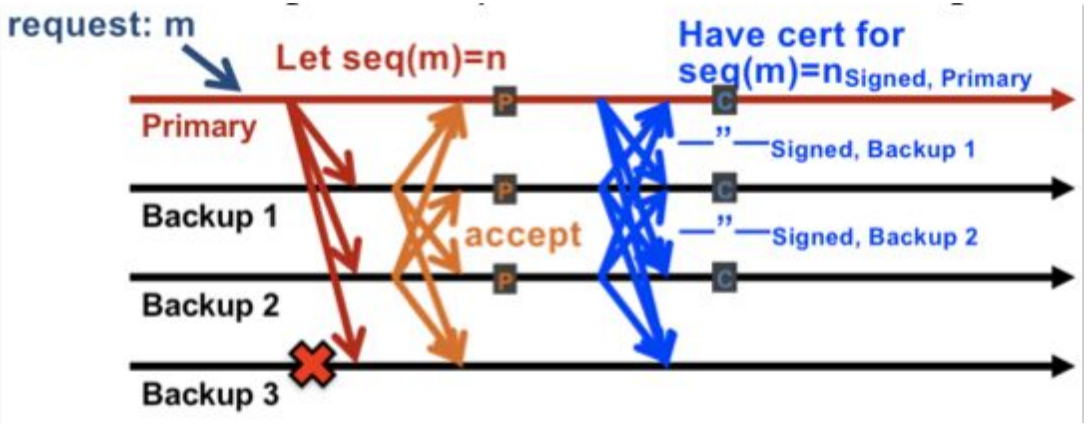
- Replicas exchange/ disseminate request they got from the primary to each other.
- Replicas wait to receive 2f+1 confirmations for the same <n,m>.
- If they get this, they are “ready to commit.”



BFT Protocol (3/4)

Phase 3: Commit

- Replicas confirm to one another that they are ready to commit.
- Replicas wait to receive “commit” confirmation from 2f+1 replicas.
- Each node that gets this, replies to the client confirming that he’s reached consensus for <n,m>.



BFT Protocol (4/4)

Termination:

- Client waits for f+1 replicas to send him confirmation of commit before he continues operation.

The End…

…But wait, there's more!

A lot of topics this course doesn't cover. Here are some keywords to search for during your next phase of DS learning:

- Pub/sub systems
- Streaming systems
- Peer-to-peer systems
- Block chains
- Security in distributed systems: authentication, key management
- Privacy and distributed data protection
- Systems for ML
- Scheduling
- Capacity planning
- Content distribution networks
- Resource discovery systems
- Serverless computing

Keep Learning!

Acknowledgement

Pictures for the BFT protocol are from Kyle Jamieson's Distributed Systems class: https://www.cs.princeton.edu/courses/archive/fall16/cos418/docs/L9-bft.pdf