Announcements

- Midterm 2 on Thursday, Dec 2 in class
- 1 page (2 sides) cheat sheet allowed for the exam
  - To be submitted along with your exam
- P3 final due on Friday, Dec 3
- Please fill out the FCEs before its due date

Distributed Systems

15-440/640

Byzantine Fault Tolerance

Readings: Tanenbaum pages 449 - 460. PBFT paper.

Failure Models

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure *Receive omission* *Send omission*	A server fails to respond to incoming requests A server fails to receive incoming messages A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure *Value failure* *State transition failure*	A server's response is incorrect The value of the response is wrong The server deviates from the correct flow of control
Arbitrary failure	A server may produce arbitrary responses at arbitrary times

Previous lectures: specific types of fail-stop behavior

From now on: specific types of **Byzantine/adversarial** behavior

What do Arbitrary Failures Look Like?

Many things can go wrong…

Communication

- Messages lost or delayed for arbitrary time
- Adversary can intercept messages and corrupt it

Processes

- Can fail or team up to produce wrong results

Agreement very hard, sometime impossible, to achieve!

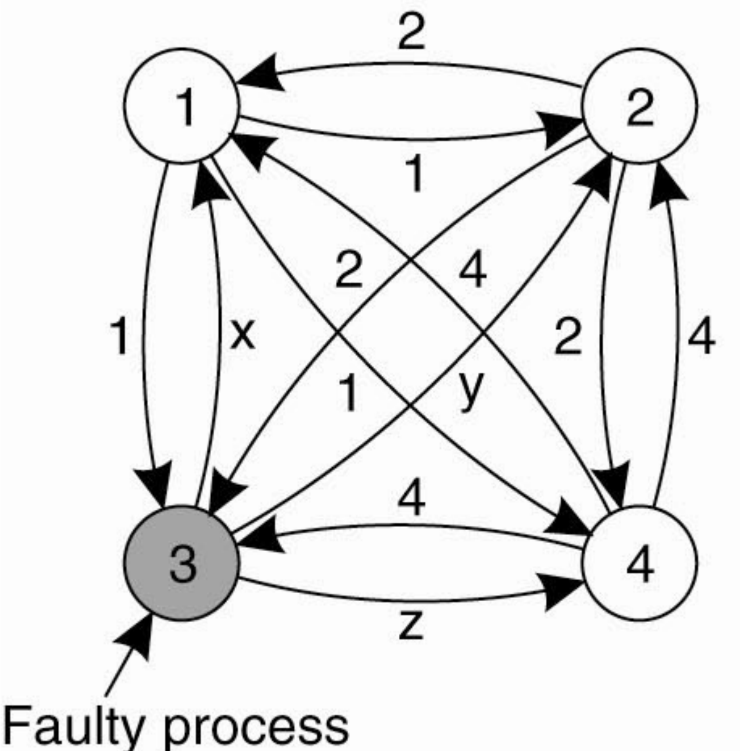
Fault Tolerance

- Terminology & Background
- Byzantine Fault Tolerance (Lamport)
- Async. BFT (Liskov)

Byzantine Agreement Problem

- System of N processes, where each process *i* will provide a value v_i to each other.
- Some number of these processes may be incorrect (or malicious)

Goal:
Each **nonfaulty** process learn the true values sent by each of the nonfaulty processes



Three nonfaulty and one faulty process.

Byzantine General's Problem

The Problem: "*Several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. After observing the enemy, they must decide upon a common plan of action. Some of the generals may be traitors, trying to prevent the loyal generals from reaching agreement*."

Goal:

- All loyal generals decide upon the same plan of action.
- A small number of traitors cannot cause the loyal generals to adopt a bad plan.

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

So far: tolerating fail-stop failures

- Traditional replicated state machine (RSM) tolerates benign failures
  - Node crashes
  - Network partitions

Given 2f+1 replicas, how many simultaneous fail-stop failures can RSM tolerate?

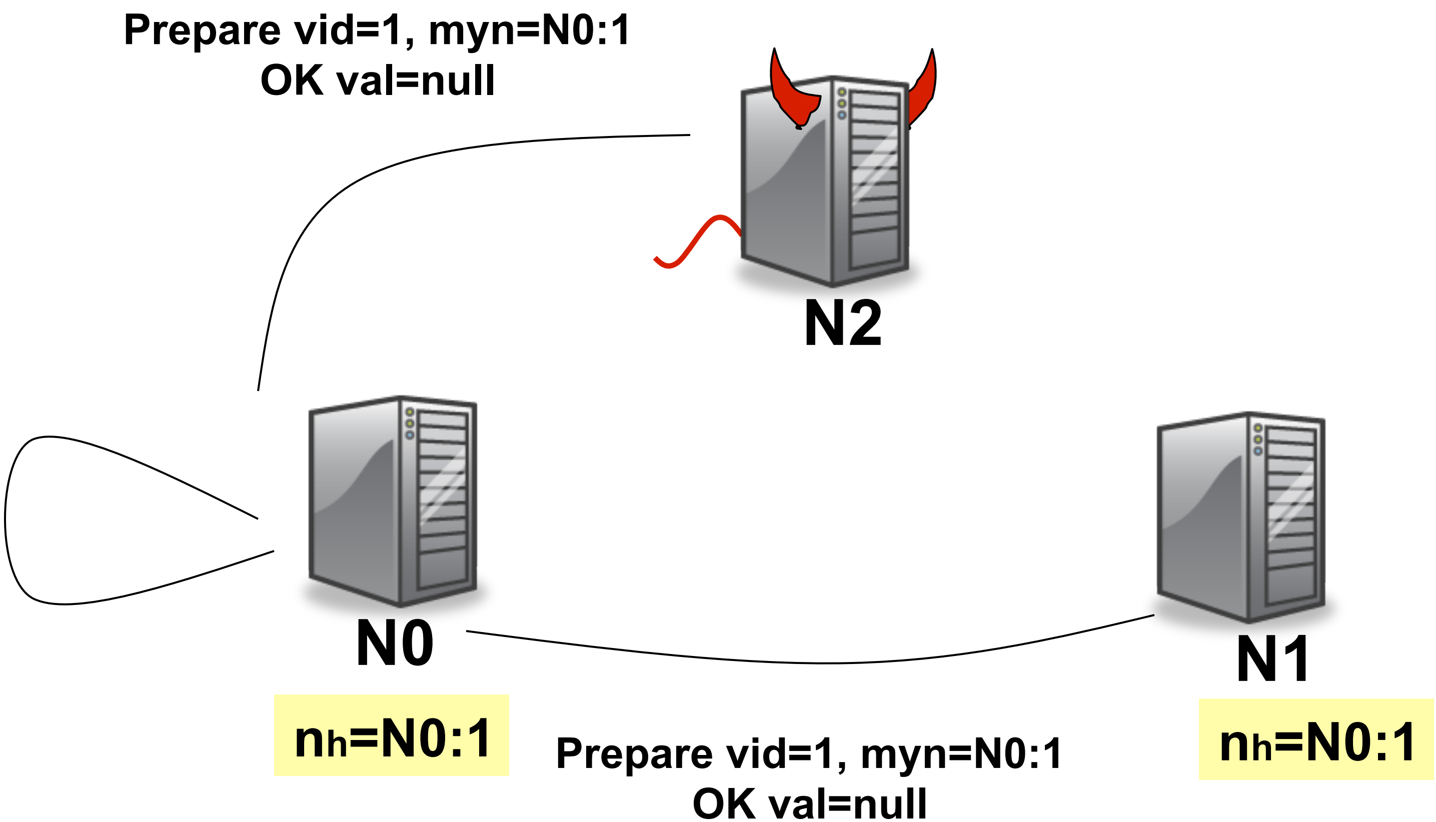
- A RSM w/ 2f+1 replicas can tolerate f simultaneous fail-stop failures

Question to ponder until next lecture: How many Byzantine/arbitrary failures can RSM (like Raft/Paxos) tolerate?

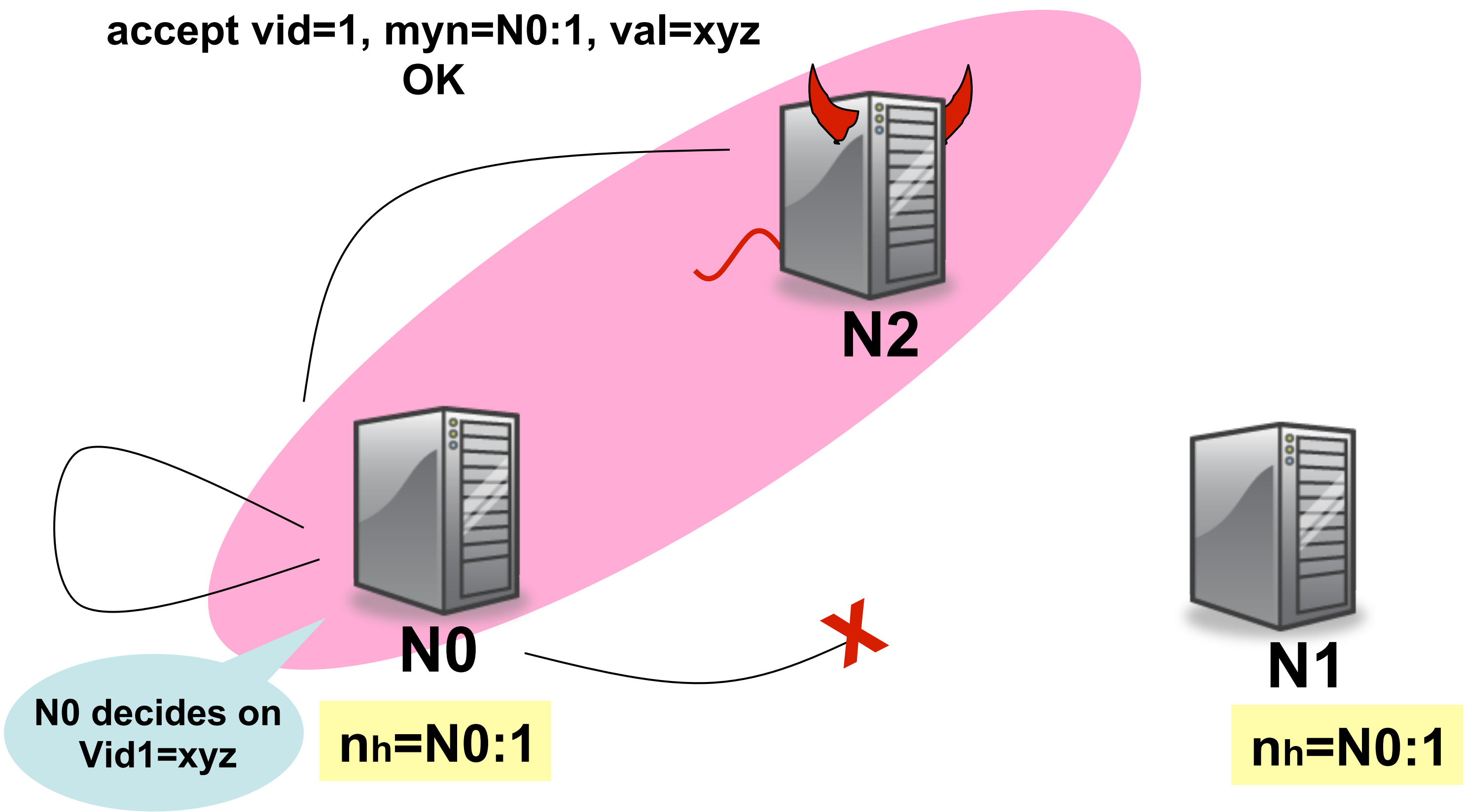
Why doesn't traditional RSM work with Byzantine nodes?

- Paxos uses a majority accept-quorum to tolerate f benign faults out of 2f+1 nodes
- Does the intersection of two quorums always contain one honest node?
- Bad node tells different things to different quorums!
  - E.g. tell N1 accept=val1 and tell N2 accept=val2

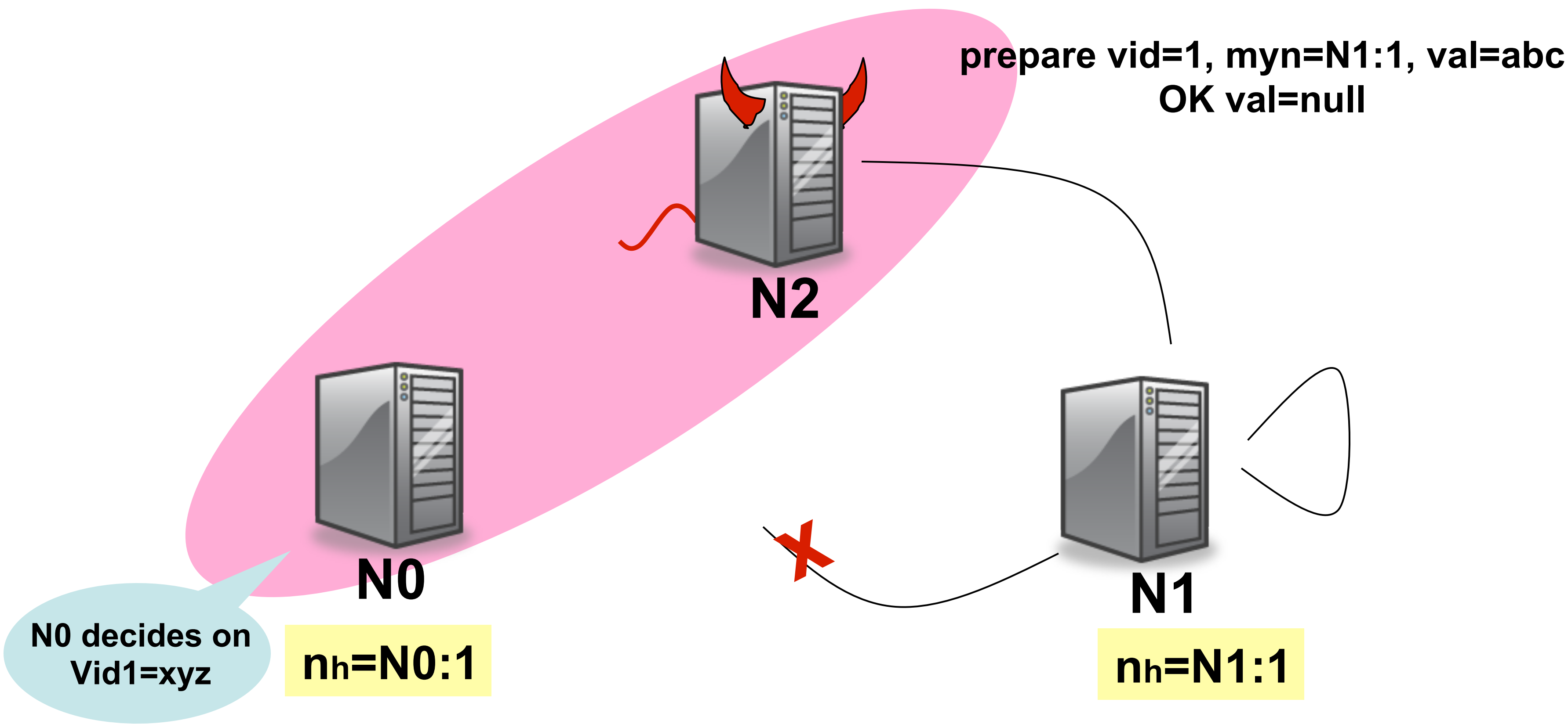
Paxos under Byzantine faults



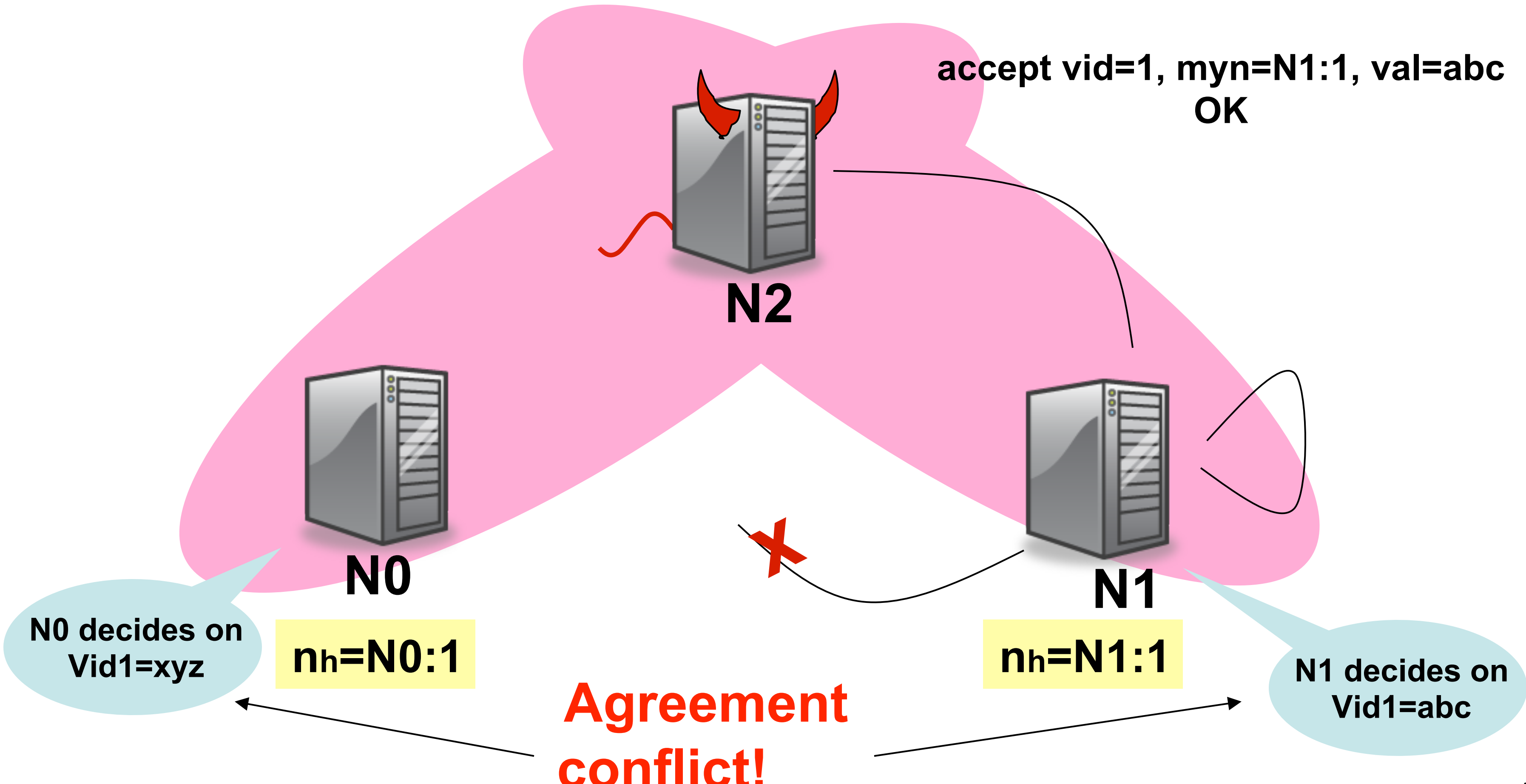
Paxos under Byzantine faults



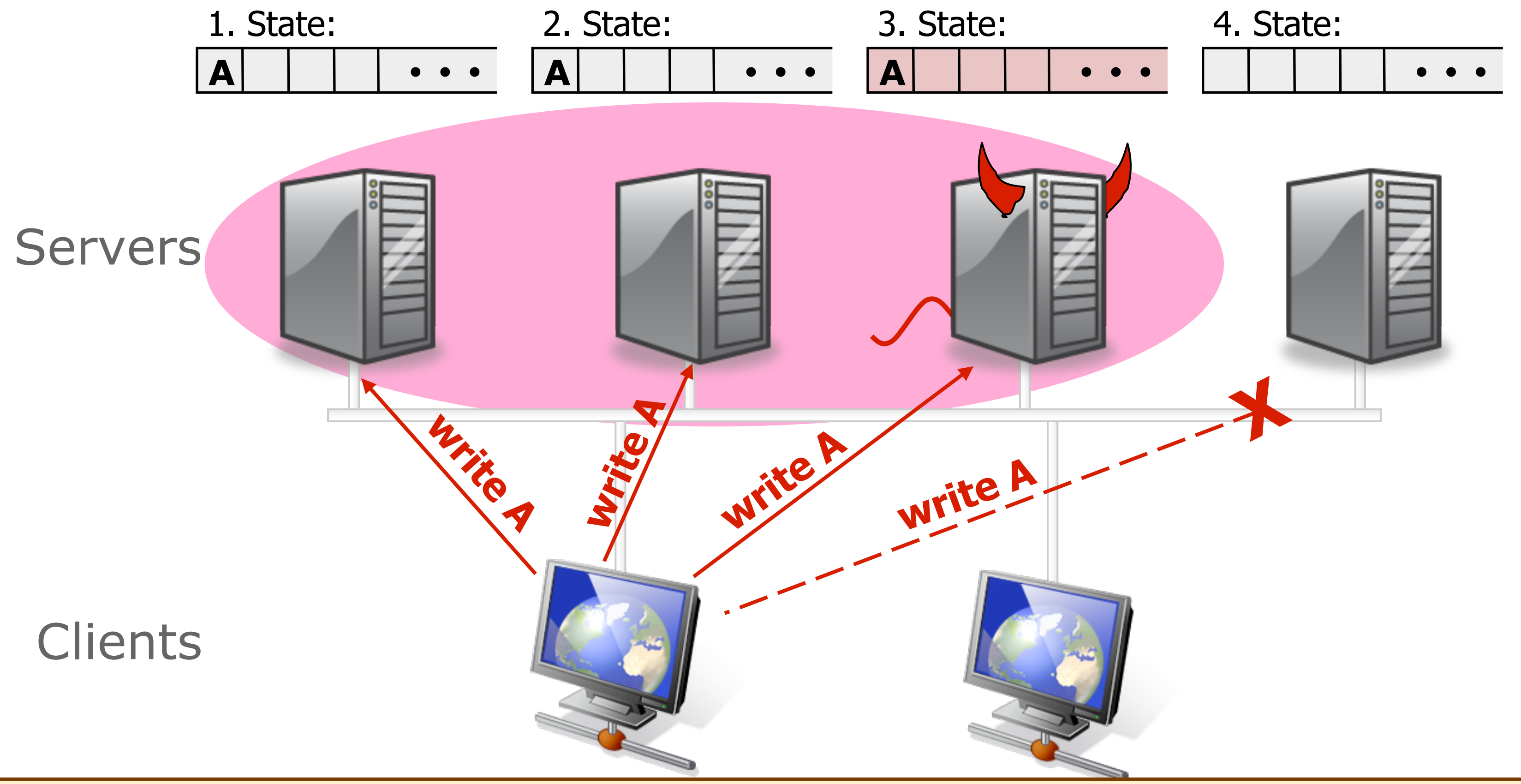
Paxos under Byzantine faults



Paxos under Byzantine faults



Quorums under Byzantine faults



For correctness, what property must the intersection of any two quorums have?

At least one honest node => intersection size at least f + 1

Byzantine General's Problem

The Problem: "*Several divisions of the Byzantine army are camped outside an enemy city, each division commanded by its own general. After observing the enemy, they must decide upon a common plan of action. Some of the generals may be traitors, trying to prevent the loyal generals from reaching agreement*."

Goal:

- All loyal generals decide upon the same plan of action.
- A small number of traitors cannot cause the loyal generals to adopt a bad plan.
- If the commander is loyal, then all loyal lieutenants obey the commander.

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

Impossibility Results

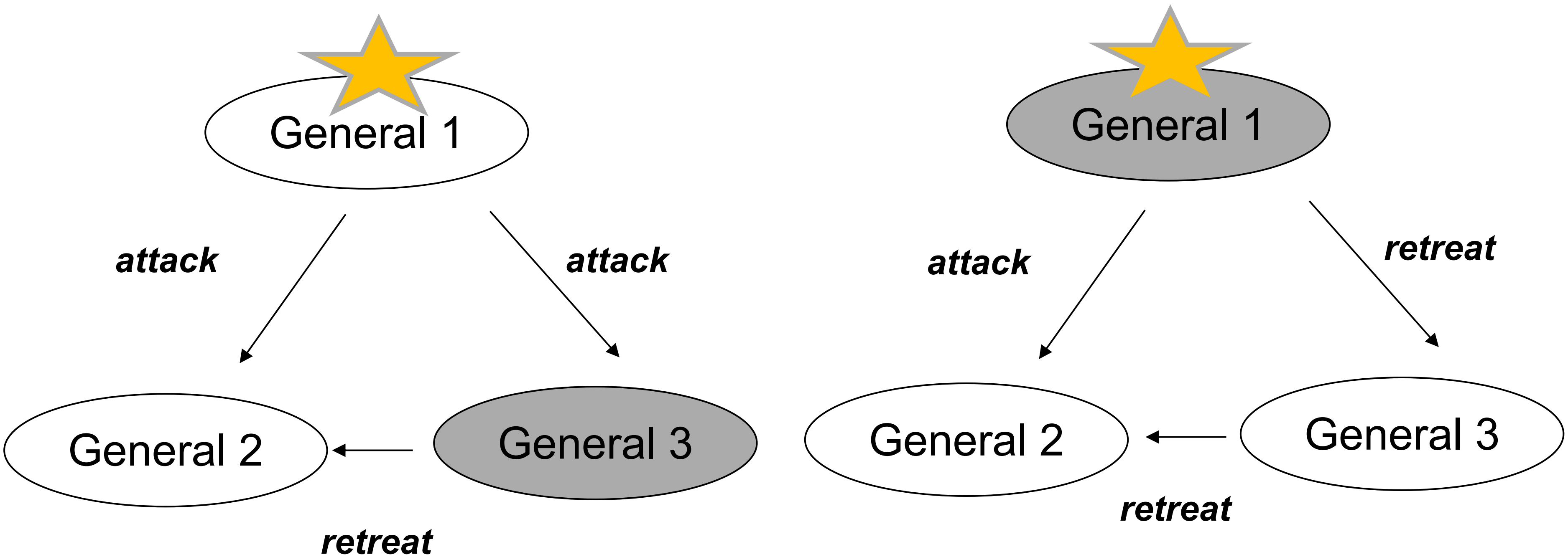
- No solution for three processes can cope with a single traitor.
- No solution with fewer than 3f + 1 generals can cope with f traitors

N >= 3f+1

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

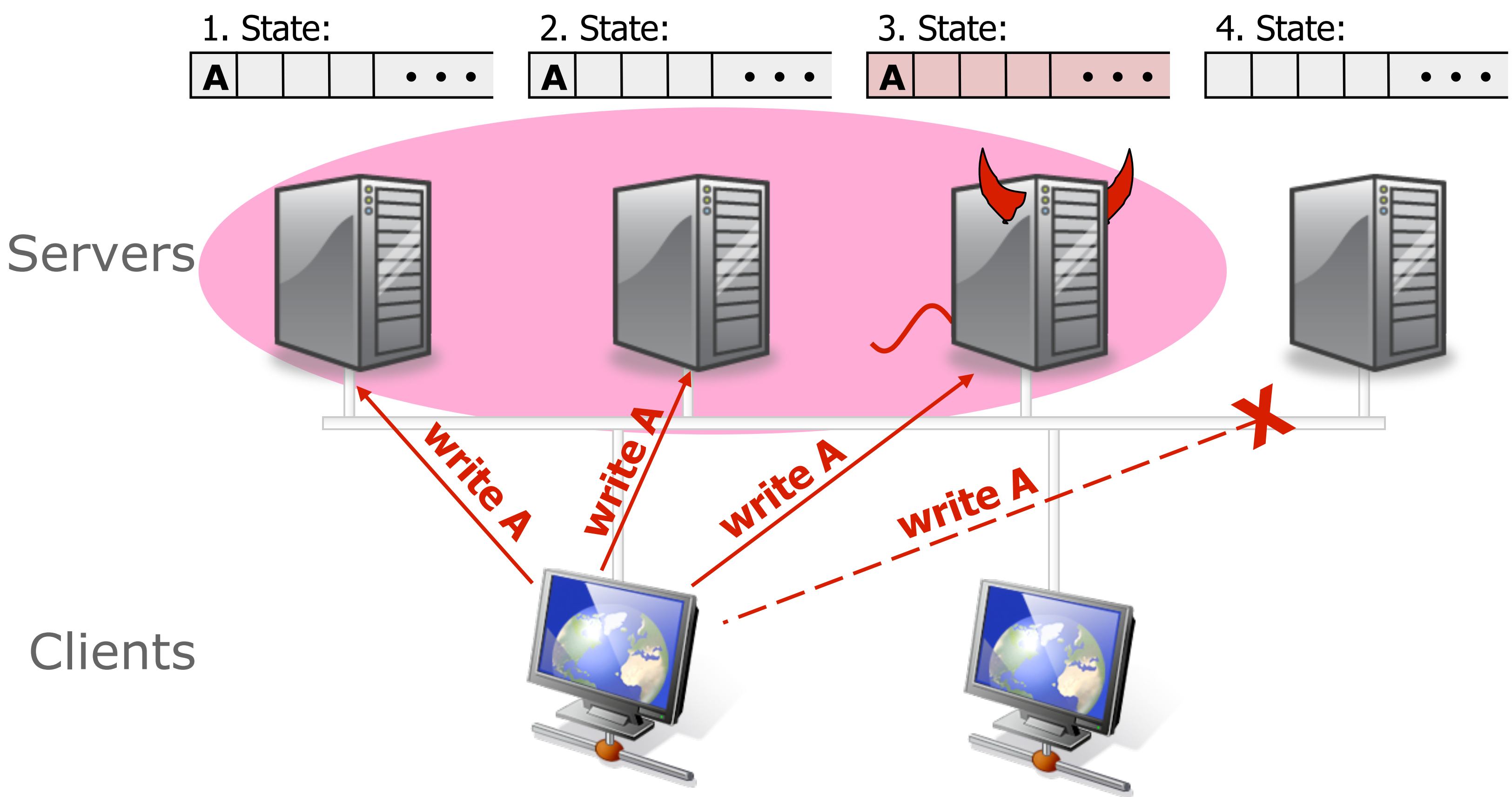
Impossibility Results

- No solution for three processes can cope with a single traitor.



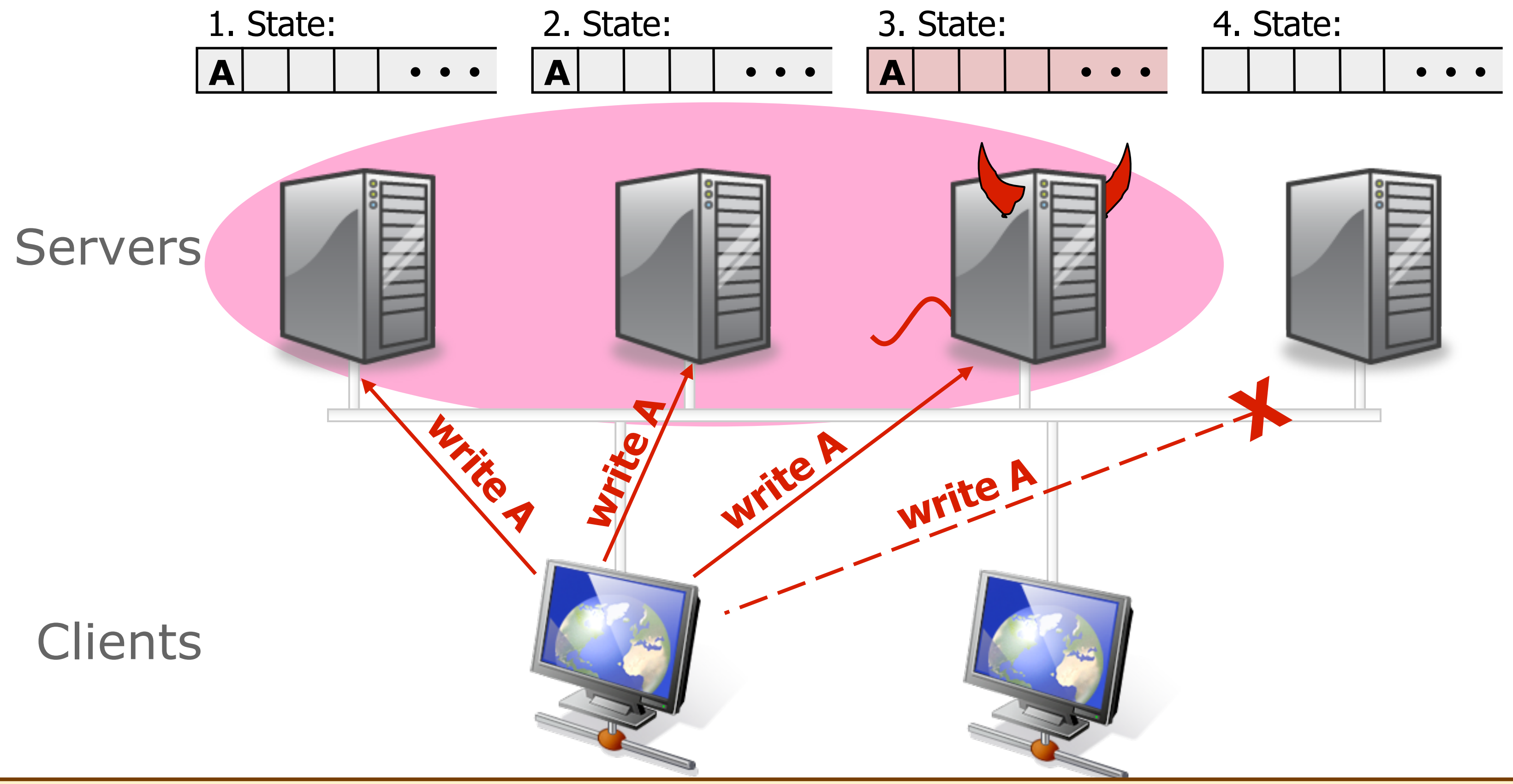
Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

Quorums under Byzantine faults



For liveness, the upper bound on the quorum size: N – f
Why?

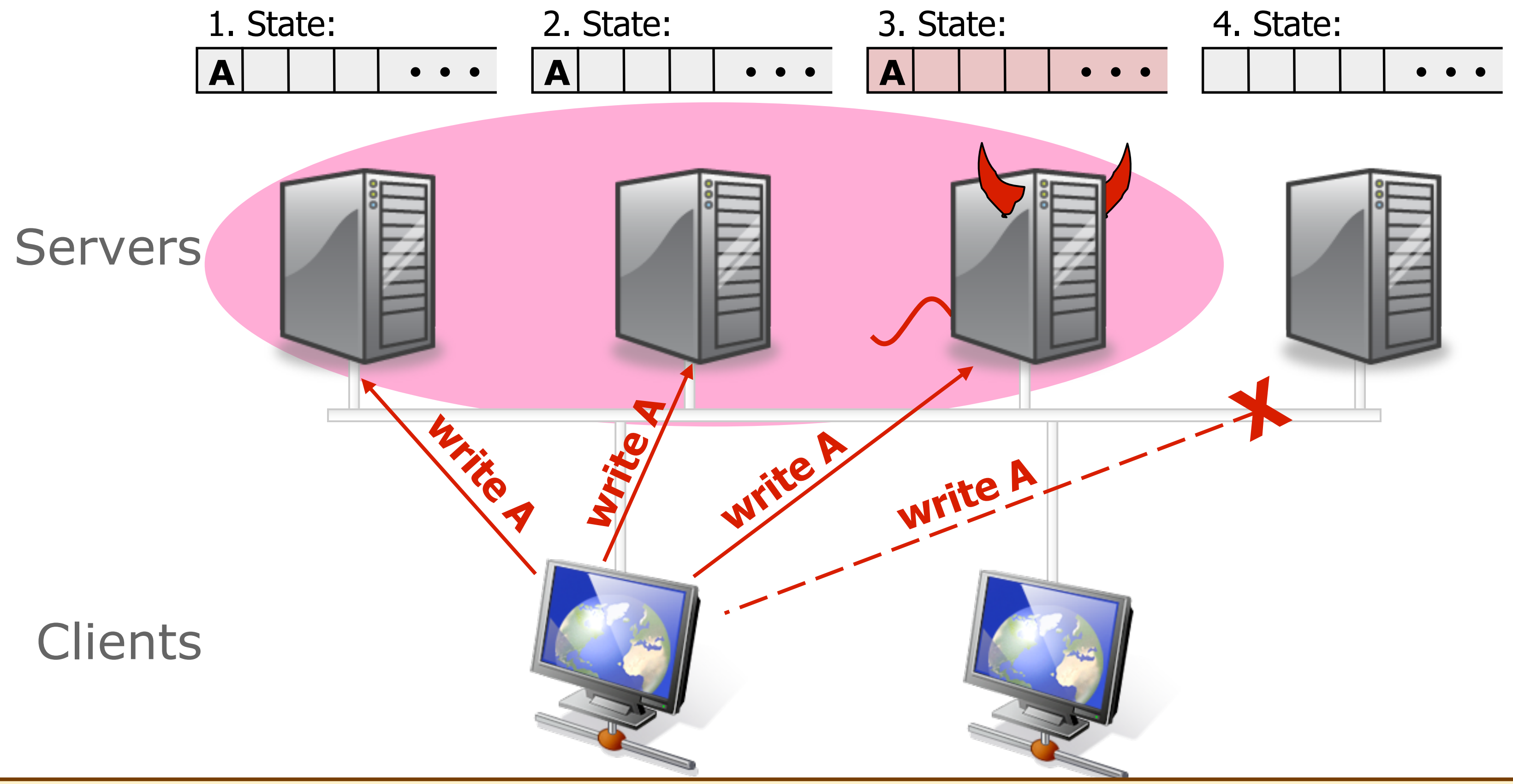
Quorums under Byzantine faults



For correctness, what property must the intersection of any two quorums have?

At least one honest node => intersection size at least f + 1

Quorums under Byzantine faults



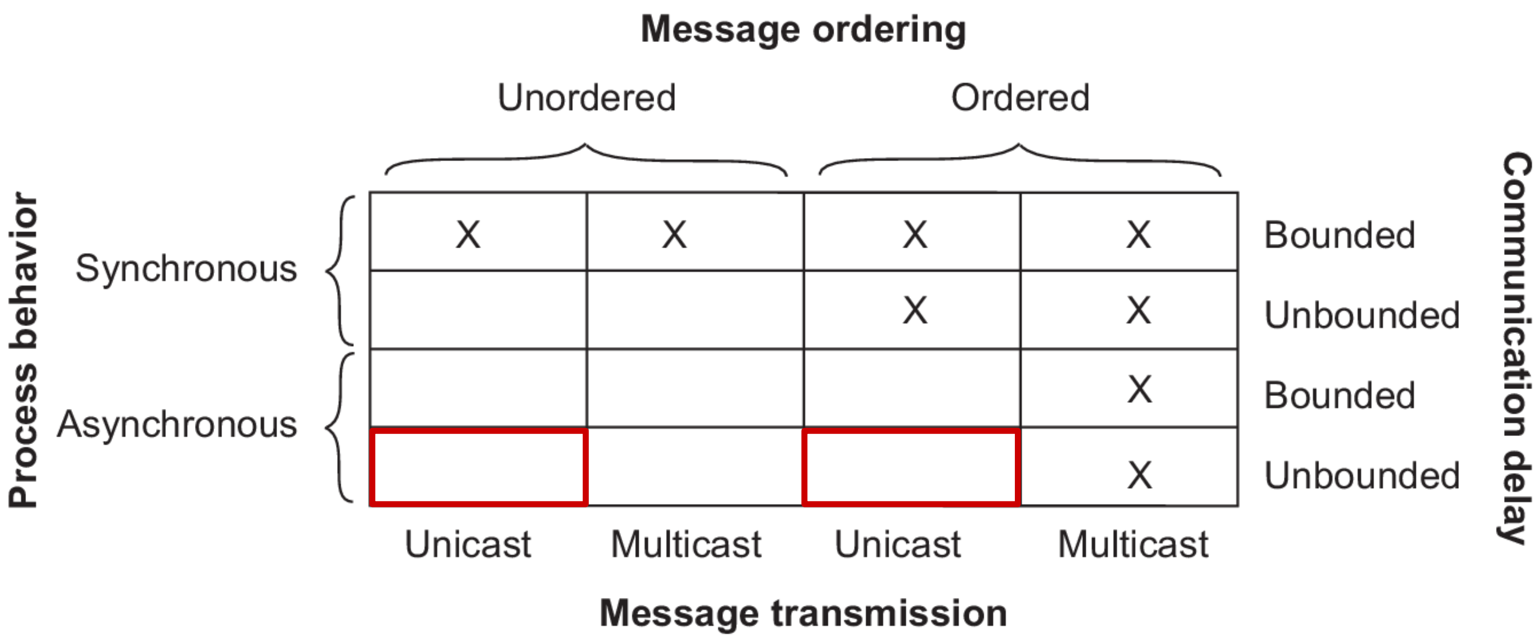
At least one honest node in the intersection =>
(N-f) + (N-f) - N >= f+1   N >= 3f+1

Agreement in Faulty Systems

Possible characteristics of the underlying system:

1. Synchronous versus asynchronous systems.
   - A system is synchronized if the process operation in lock-step mode. Otherwise, it is asynchronous.
2. Communication delay is bounded or not.
3. Message delivery is ordered or not.
4. Message transmission is done through unicasting or multicasting.

Agreement in Faulty Systems



Circumstances under which distributed agreement can be reached.

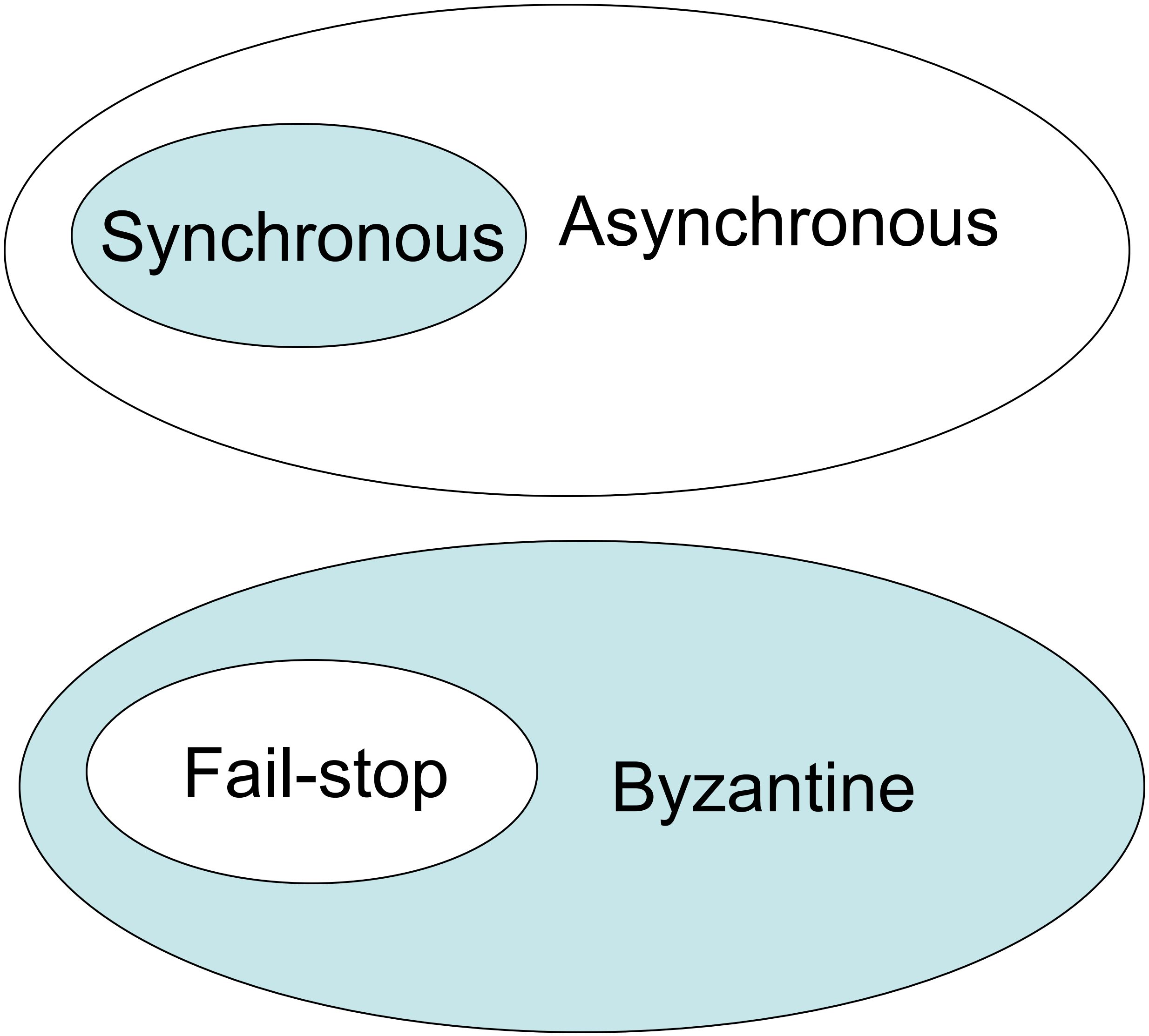
Note that most distributed systems assume that

1. processes behave asynchronously
2. messages are unicast
3. communication delays are unbounded (see red blocks)

Fault Tolerance

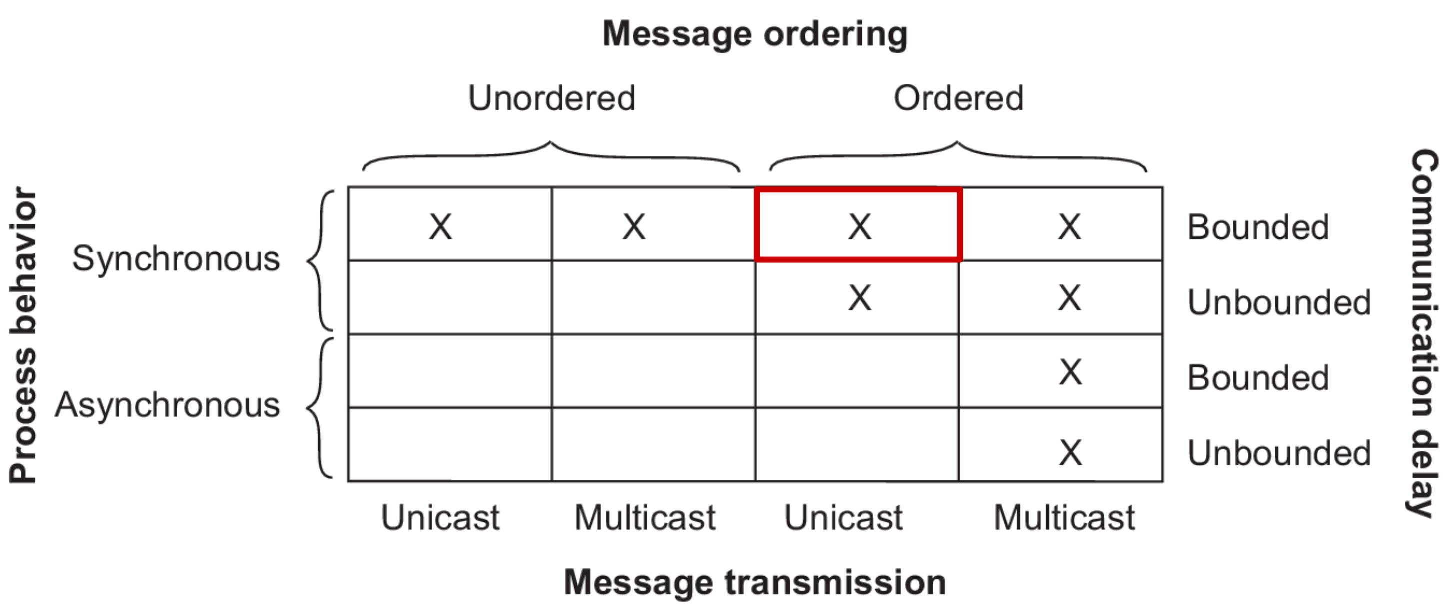
- Terminology & Background
- **Sync. Byzantine Fault Tolerance (Lamport)**
- Async. BFT (Liskov)

Synchronous, Byzantine world



Agreement in Faulty Systems

- Byzantine Agreement [Lamport, Shostak, Pease, 1982]
- Assumptions:
  - Every message that is sent is delivered correctly
  - The receiver knows who sent the message
  - Message delivery time is bounded



Byzantine Agreement Algorithm (oral messages) - Example

- 4 processes: N = 4
- At most 1 is faulty: f = 1

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

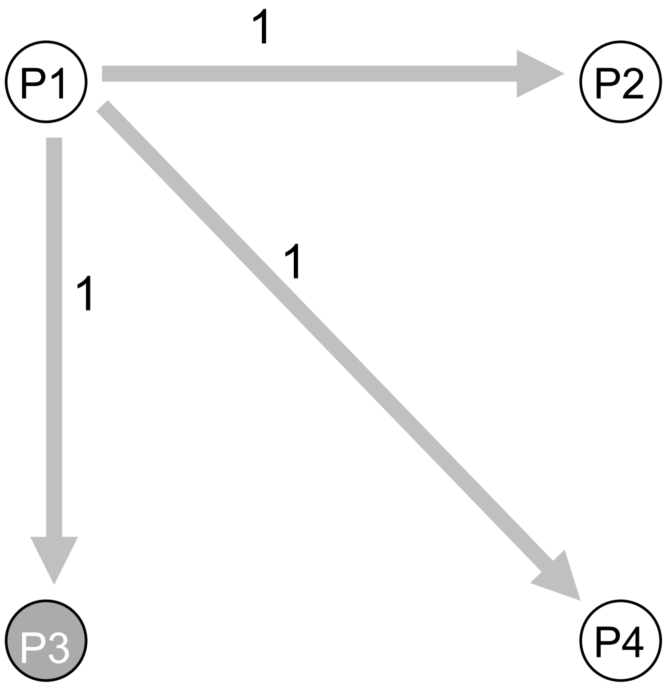
Byzantine Agreement Algorithm (oral messages) - 1

- Phase 1: Each process sends its value to the other processes.
    - Correct processes send the same (correct) value to all.
    - Faulty processes may send different values to each if desired (or no message).

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

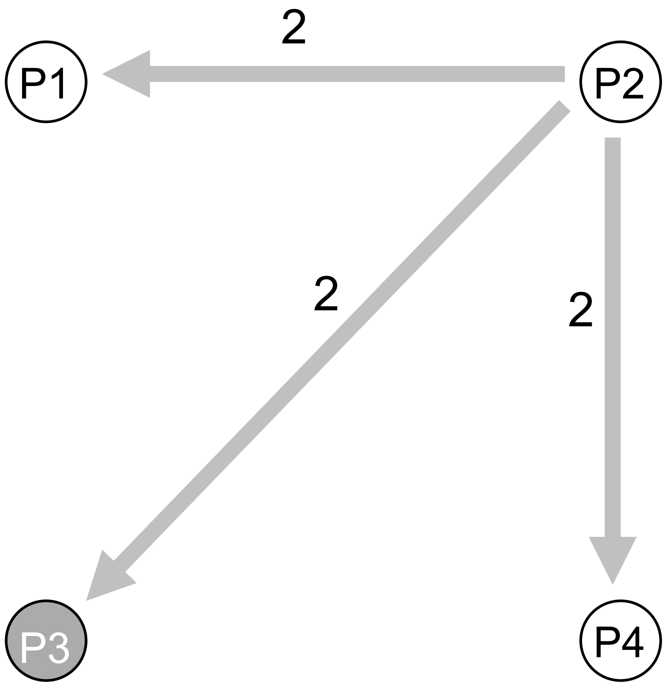
Byzantine General Problem Example - 1

- Phase 1:



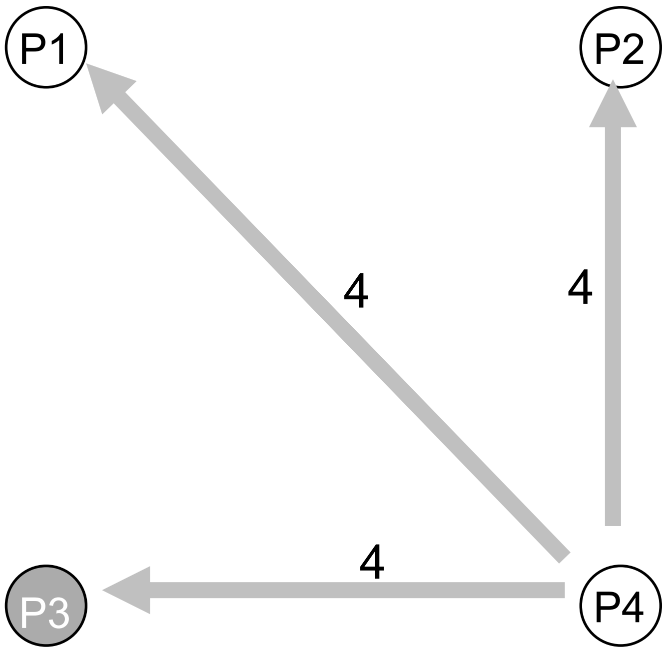
Byzantine General Problem Example - 2

- Phase 1:



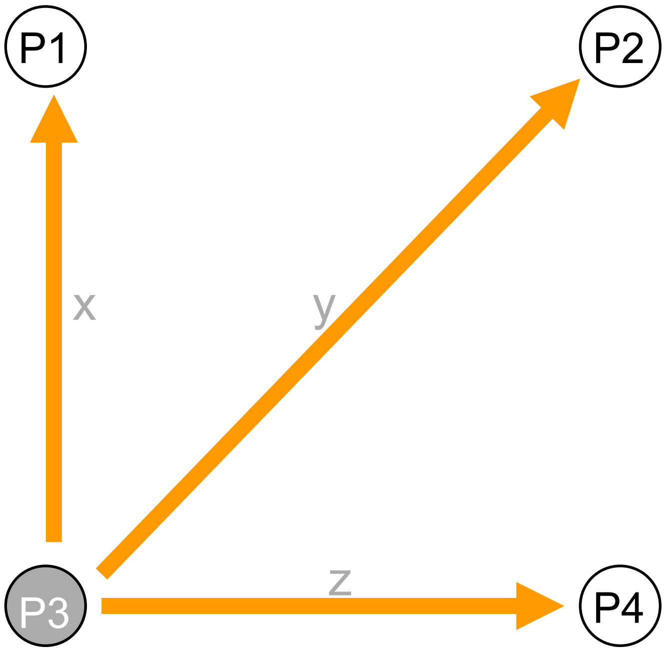
Byzantine General Problem Example - 3

- Phase 1:



Byzantine General Problem Example - 4

- Phase 1:



Byzantine Agreement Algorithm (oral messages) - 2

- Phase 2: Each process uses the messages to create a vector of responses – must be a default value for missing messages.

Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

Byzantine General Problem Example - 5

- Phase 2:

P1

P1	P2	P3	P4
1	2	x	4

P2

P1	P2	P3	P4
1	2	y	4

P3

P4

P1	P2	P3	P4
1	2	z	4

Byzantine Agreement Algorithm (oral messages) - 3

- Phase 3: Each process sends its vector to all other processes.
- Phase 4: Each process uses information received from every other process to do majority voting

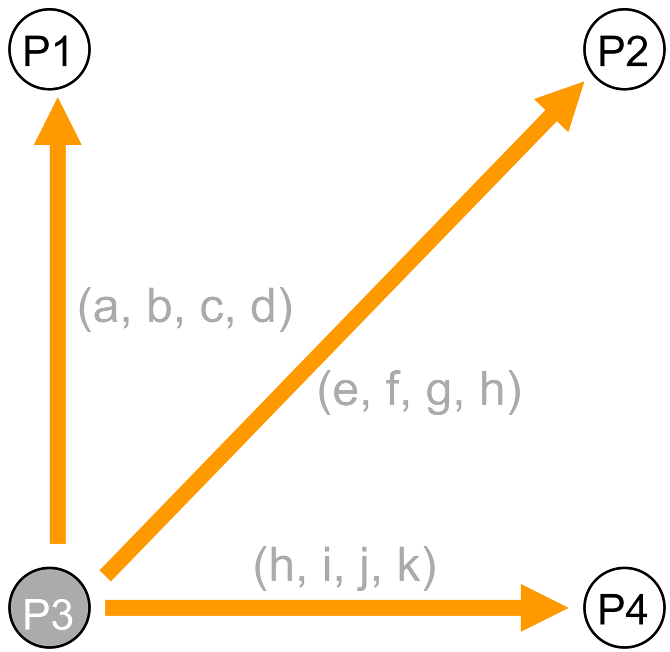
Lamport, Shostak, Pease. The Byzantine General's Problem. ACM TOPLAS, 4,3, July 1982, 382-401.

Byzantine General Problem Example - 6

- Phase 3,4:

	P1	P2	P3	P4
P2	1	2	y	4
P3	a	b	c	d
P4	1	2	z	4

(1, 2, ?, 4)



	P1	P2	P3	P4
P1	1	2	x	4
P3	e	f	g	h
P4	1	2	z	4

(1, 2, ?, 4)

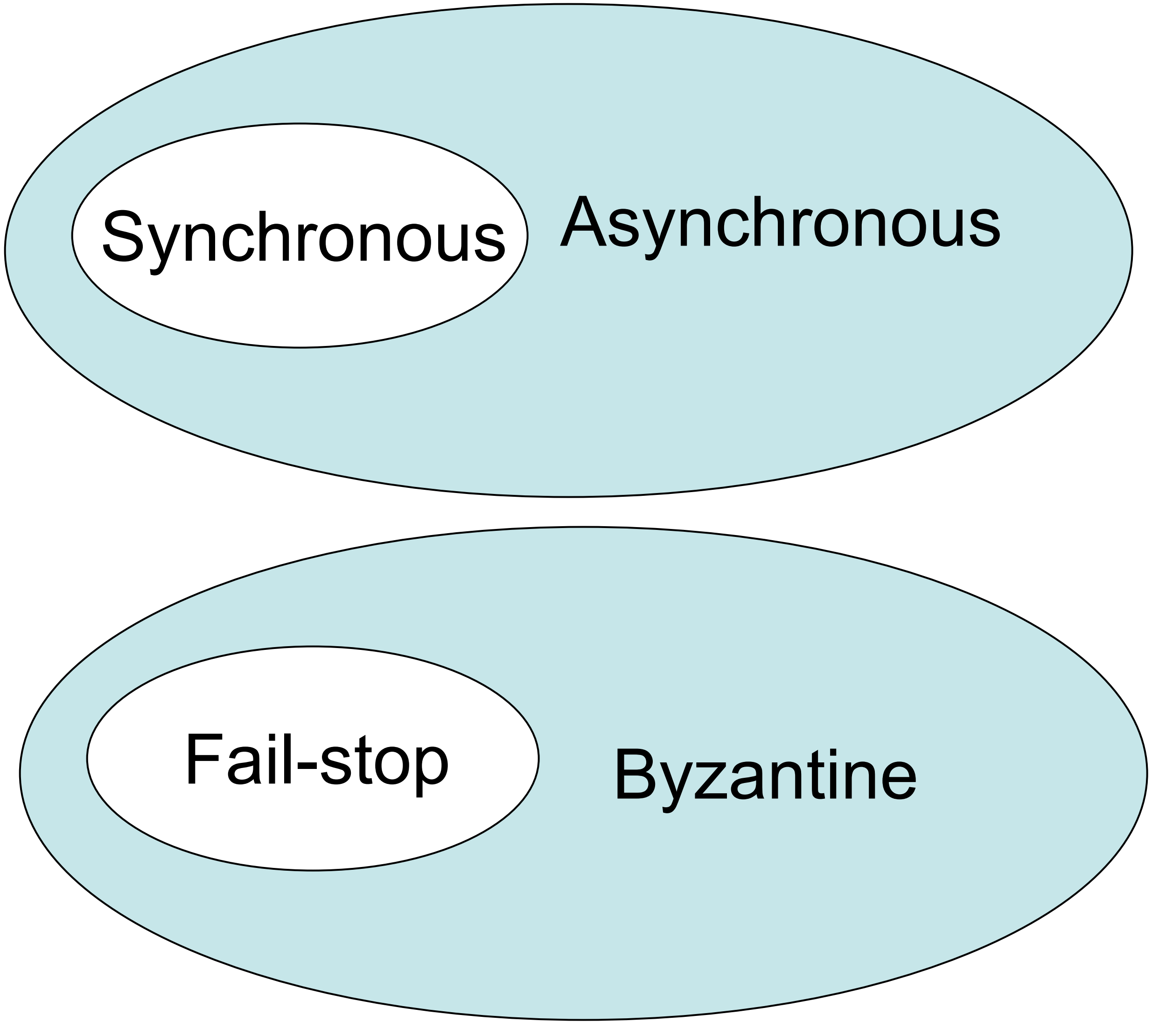
	P1	P2	P3	P4
P1	1	2	x	4
P2	1	2	y	4
P3	h	i	j	k

(1, 2, ?, 4)

Fault Tolerance

- Terminology & Background
- Byzantine Fault Tolerance (Lamport)
- **Async. BFT (Liskov)**

Practical Byzantine Fault Tolerance:Asynchronous, Byzantine



Why async?

Faulty network can violate timing assumptions

PBFT ideas

- PBFT, “Practical Byzantine Fault Tolerance”, M. Castro and B. Liskov, SOSP 1999
- Replicate service across many nodes
  - Assumption: only a small fraction of nodes are Byzantine
  - Rely on a super-majority of votes to decide on correct computation.
  - Makes some weak synchrony (message delay) assumptions to ensure liveness
    - Why?
    - Would violate FLP impossibility otherwise

- PBFT property: tolerates <=f failures

  using a RSM with 3f+1 replicas

PBFT main ideas

- Static configuration (same 3f+1 nodes)
- Primary-Backup Replication + Quorums
- To deal with malicious primary
  - Use a 3-phase protocol to agree on sequence number
- To deal with loss of agreement
  - Use a bigger quorum (2f+1 out of 3f+1 nodes)
- New primary (new “view”)
- Need to authenticate communications (MACs)

Replica state

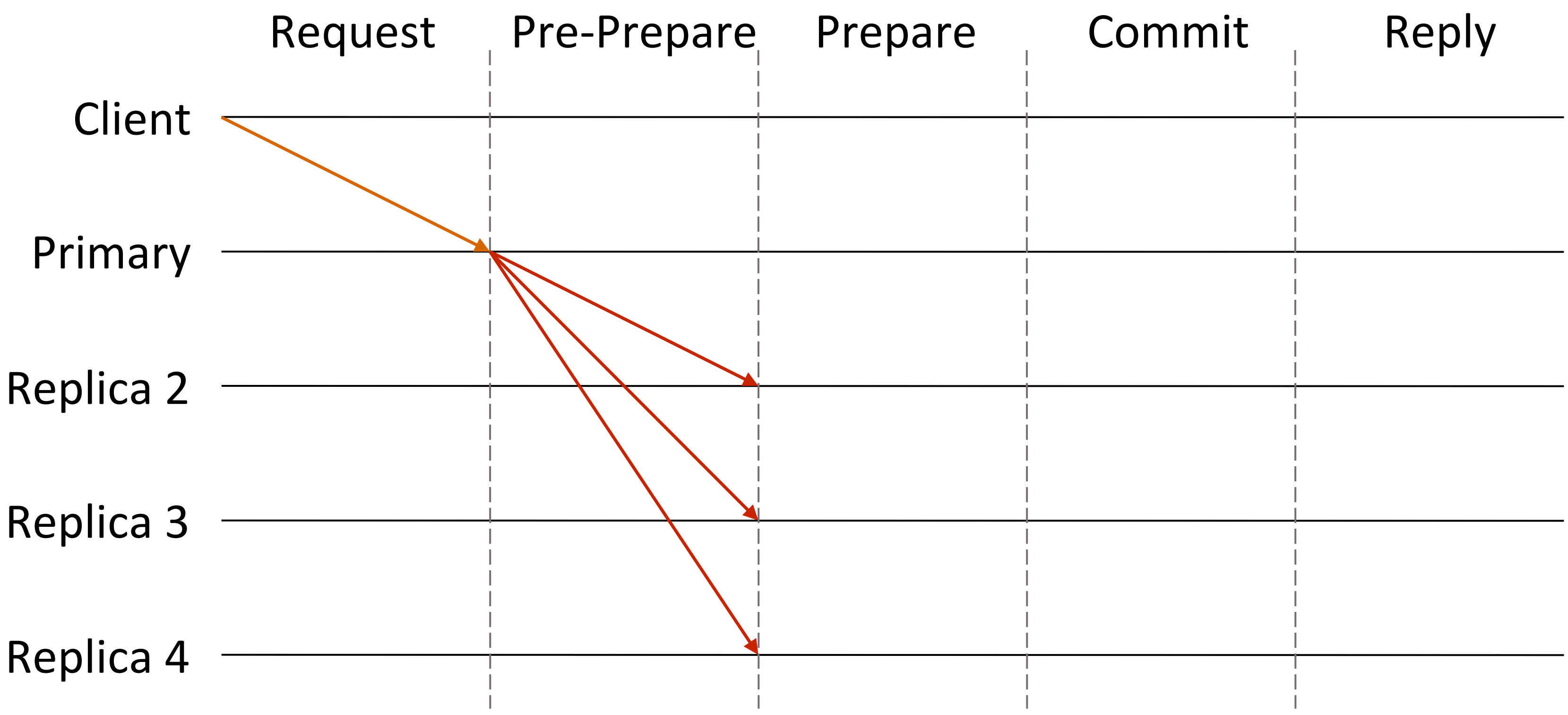
- A replica id i (between 0 and N-1)
  - Replica 0, replica 1, …
- A view number v#, initially 0
- Primary is the replica with id
  i = v# mod N
- A log of <op, seq#, status> entries
  - Status = pre-prepared or prepared or committed

Normal Case

- Client sends request to Primary
- Primary sends pre-prepare message to all

  Pre-prepare contains <v#,seq#,op>
  - Records operation in log as pre-prepared

PBFT



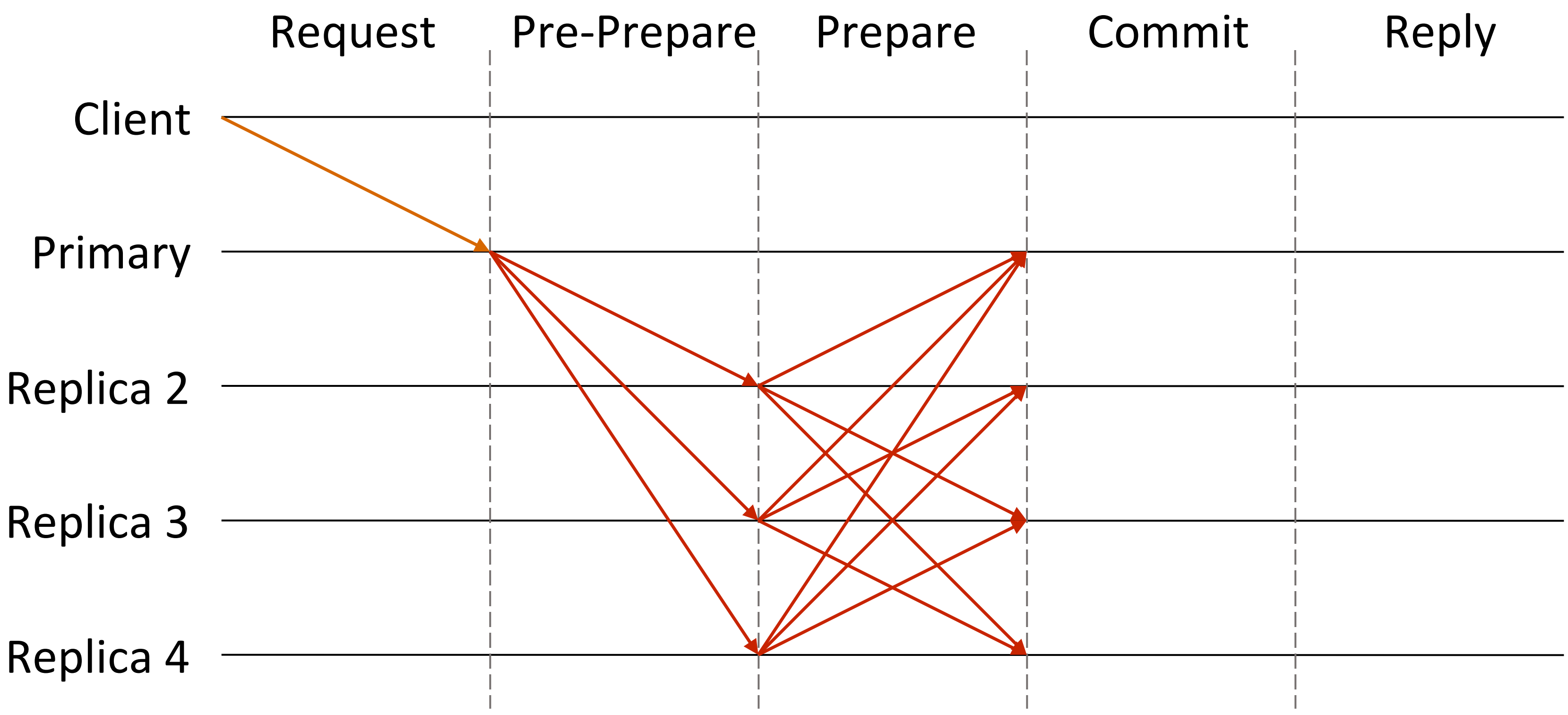
Request
Pre-Prepare
Prepare
Commit
Reply
Client
Primary
Replica 2
Replica 3
Replica 4

Normal Case

- Replicas check the pre-prepare message
- If pre-prepare is ok:
  - Record operation in log as pre-prepared
  - Send prepare messages to all
  - Prepare contains <i,v#,seq#,op>

- All to all communication

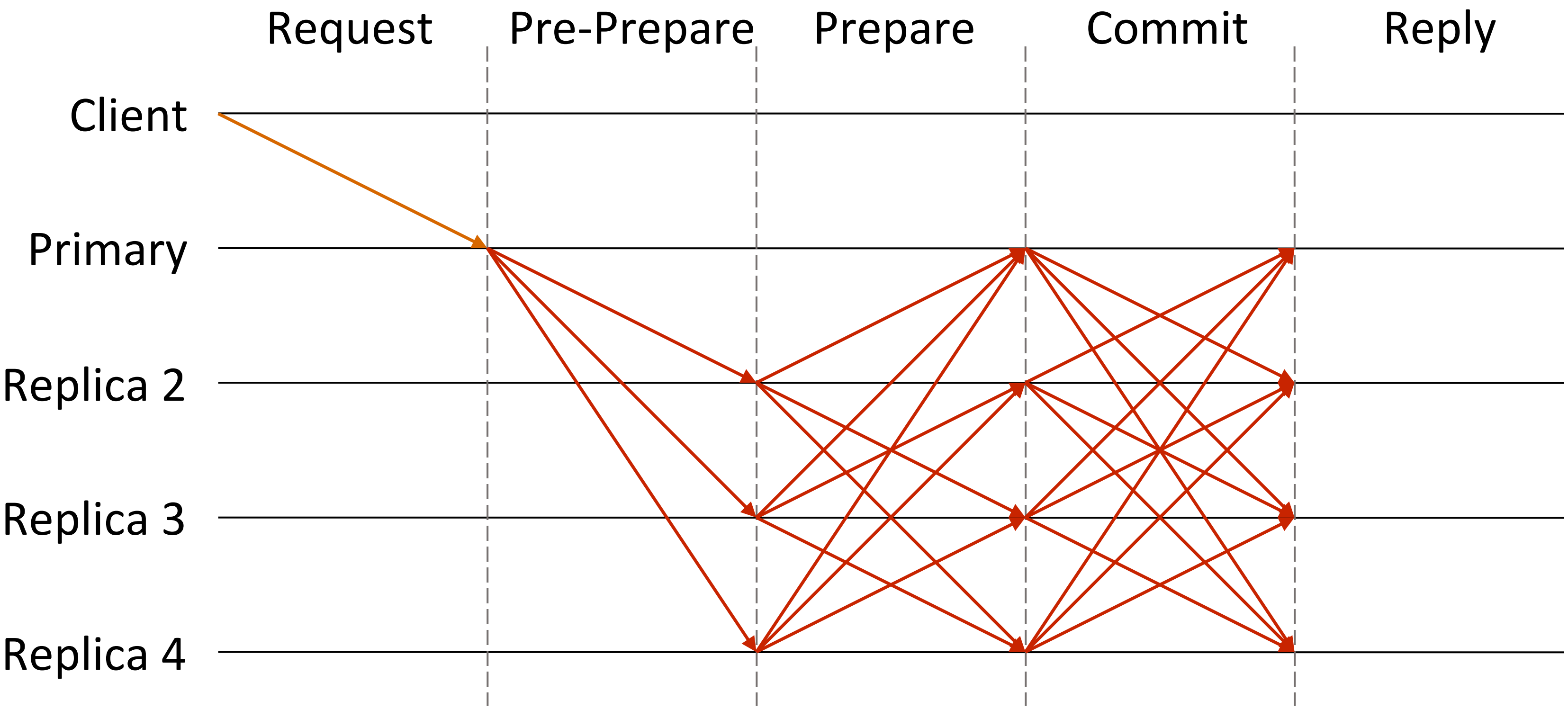
PBFT


Request
Pre-Prepare
Prepare
Commit
Reply
Client
Primary
Replica 2
Replica 3
Replica 4

Normal Case:

- Replicas wait for 2f+1 matching prepares
  - Record operation in log as prepared
  - Send commit message to all
  - Commit contains <i,v#,seq#,op>
- What does this stage achieve:
  - All honest nodes that are prepared prepare the same value
  - At least f+1 honest nodes have sent prepare/pre-prepare

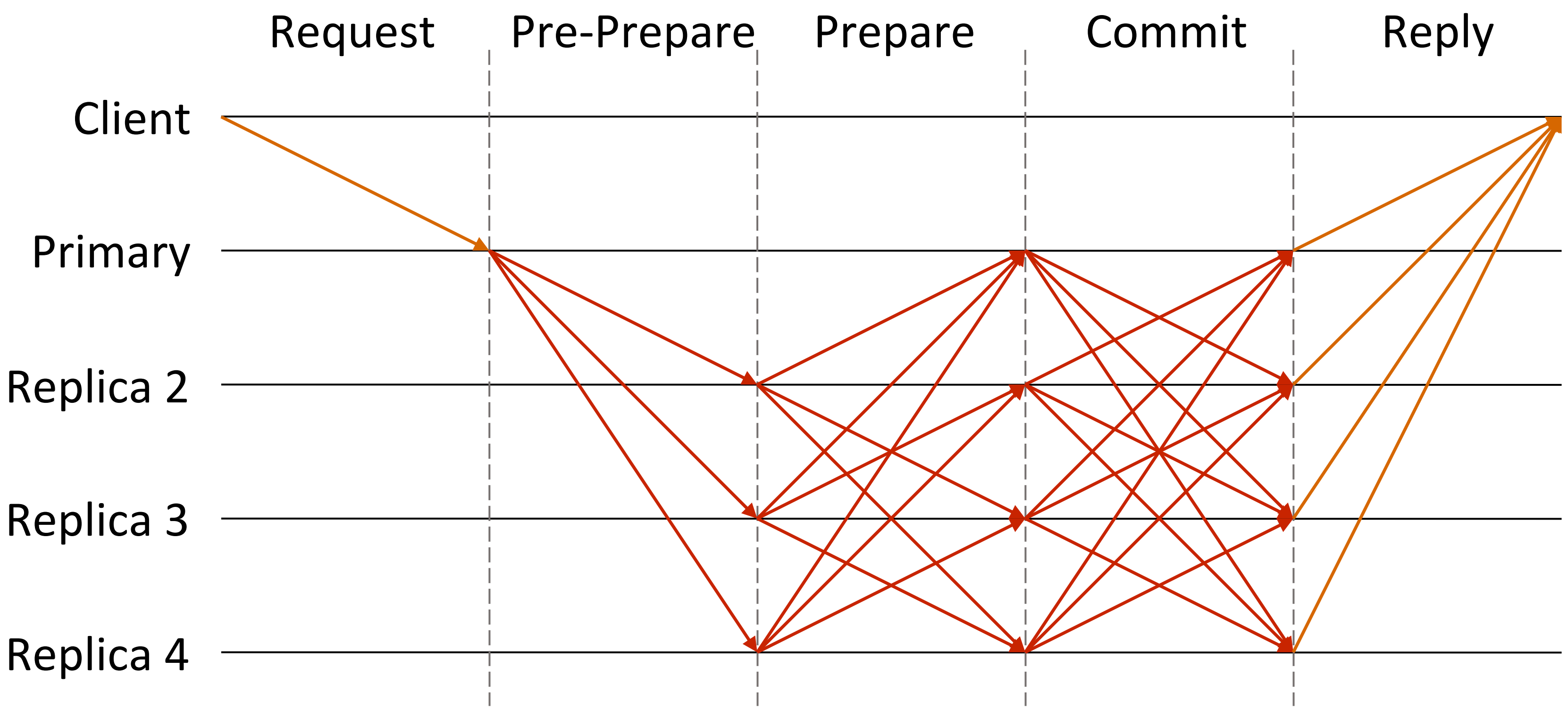
PBFT


Request
Pre-Prepare
Prepare
Commit
Reply
Client
Primary
Replica 2
Replica 3
Replica 4

Normal Case:

- Replicas wait for 2f+1 matching commits
  - Absent view-change, a node only needs f+1 matching commits, but under the view change logic (not discussed), 2f+1 ensures eventual convergence even if operations were committed in different views.
- Record operation in log as committed
  - Execute the operation
  - Send result to the client

PBFT



Request
Pre-Prepare
Prepare
Commit
Reply
Client
Primary
Replica 2
Replica 3
Replica 4

Normal Case

- Client waits for f+1 matching replies

Why f+1? What does this ensure?

- Ensures that at least one honest node is among these nodes

Practical limitations of BFTs

- Expensive

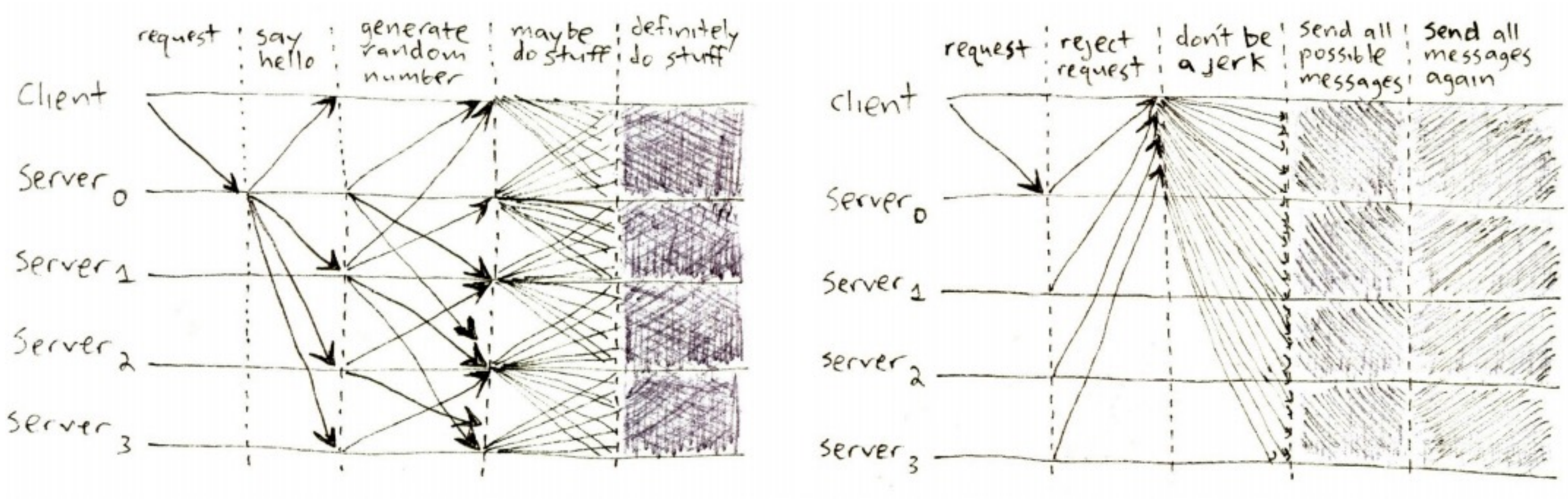


Figure 2: Our new protocol is clearly better.

- Protection is achieved only when <= f nodes fail
  - How to know in advance: how many nodes will fail?

Figure from “The Saddest Moment” article by James Mickens.

Practical Application of BFTs

- While very expensive, still need to deal with arbitrary failures
- "Small" safety-critical systems

SpaceX Dragon requirement for ISS docking procedure.

[Robert Rose, SpaceX, Embedded Linux Conference, 2013]

Boeing 777/ 787 flight control systems

[Zurawski, Richard. Industrial Communication Technology, 2nd ed, 2015]

- "Large" (but low-throughput) distributed ledgers

Bitcoin
Ripple
Ethereum
NXT
Litecoin
ZCASH
Ethereum Classic
Dogercoin