# Practical Byzantine Fault Tolerance

Academic Study presentation for Distributed systems

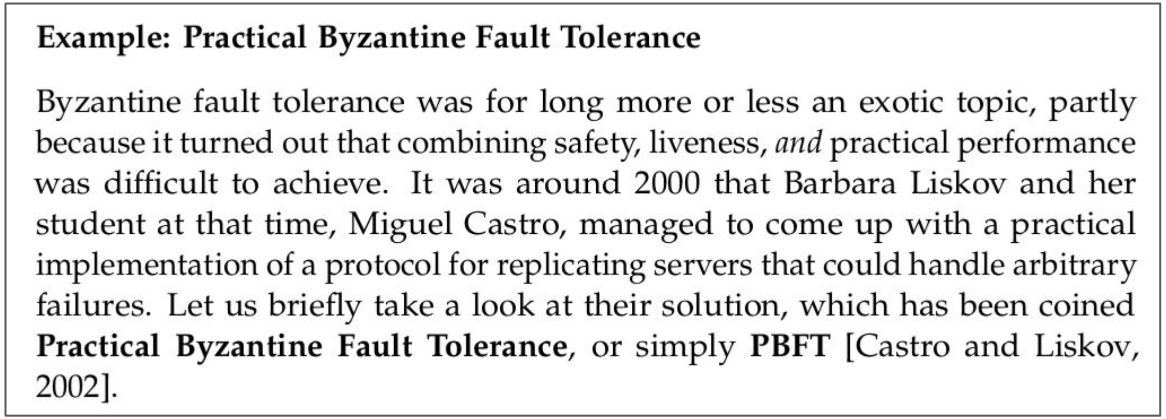
By Altanai Bisht
Dept of Computer Science , Seattle University ( 2021)

## About

Practical Byzantine Fault Tolerance and Proactive Recovery - MIGUEL CASTRO Microsoft Research

And BARBARA LISKOV MIT Laboratory for Computer Science

ACM Transactions on Computer Systems, 20(4):398–461, Nov. 2002. Cited on 458, 460



**Example: Practical Byzantine Fault Tolerance**

Byzantine fault tolerance was for long more or less an exotic topic, partly because it turned out that combining safety, liveness, *and* practical performance was difficult to achieve. It was around 2000 that Barbara Liskov and her student at that time, Miguel Castro, managed to come up with a practical implementation of a protocol for replicating servers that could handle arbitrary failures. Let us briefly take a look at their solution, which has been coined **Practical Byzantine Fault Tolerance**, or simply **PBFT** [Castro and Liskov, 2002].

## Challenges in Distributed Systems

Distributed systems are subject to a variety of failures and attacks -Hacker break-in, Data corruption, Software/hardware failure. But in Byzantine failure model: Faulty nodes may exhibit arbitrary behavior - Malicious attacks

Consensus Protocol Goals

- Liveness
- Clients receive replies to requests
- Safety
- Replicated service is linearizable
  i.e. it appears centralized w/ atomic ops

We need n > 3f nodes

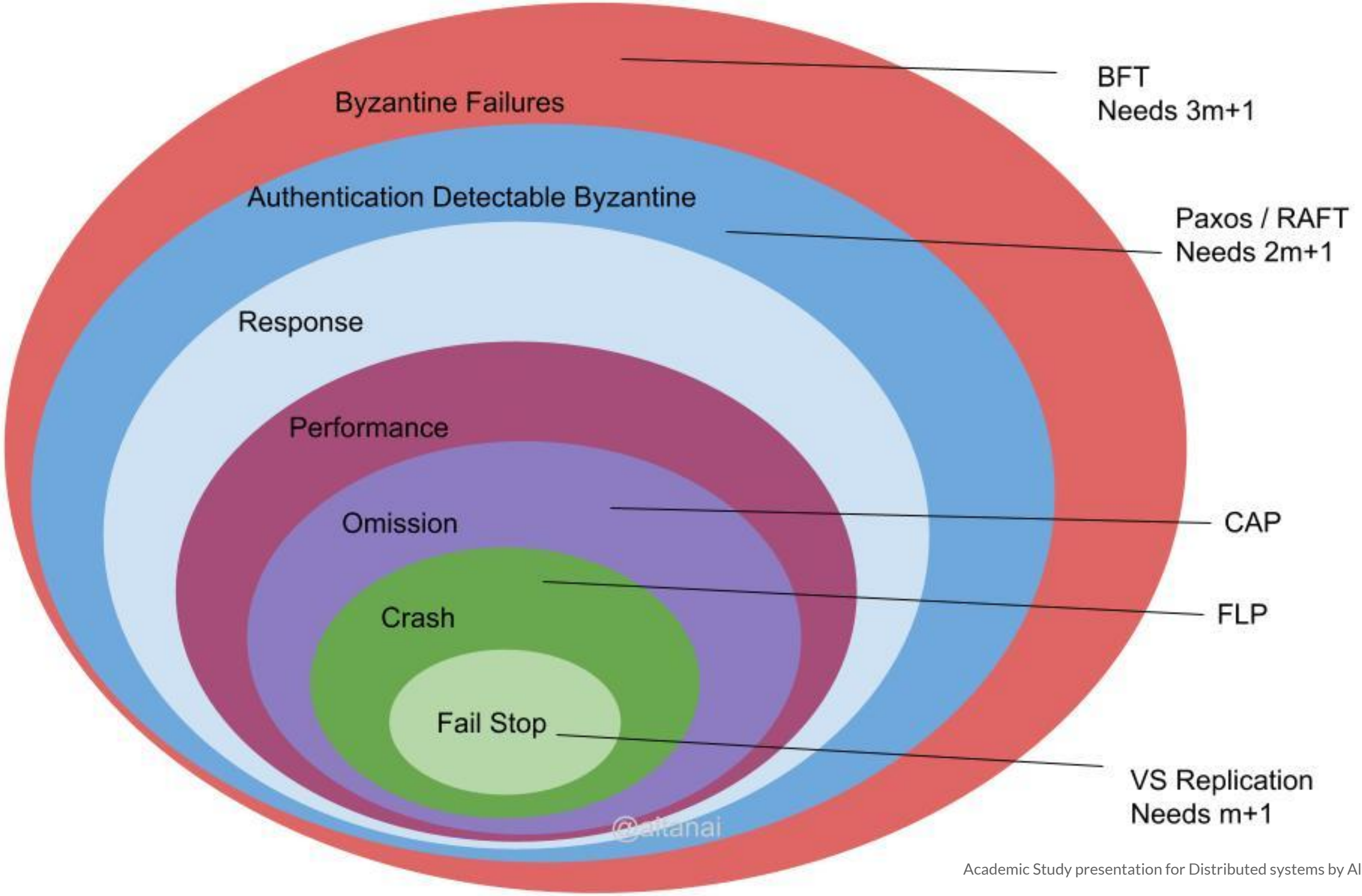
- 2f+1 to act with confidence, f may never respond

Accountable Systems

- Actions are undeniable
- State is tamper evident
- Correctness can be certified

[Yumerefendi05] Example: Building trust in federated systems

## Faults


Byzantine Failures
Authentication Detectable Byzantine
Response
Performance
Omission
Crash
Fail Stop
BFT
Needs 3m+1
Paxos / RAFT
Needs 2m+1
CAP
FLP
VS Replication
Needs m+1
@altanai

## Prior approaches

Prior approaches to replication tolerate benign faults

- Alsberg and Day [1976], Gifford [1979], Viewstamped Replication Oki and Liskov [1988], Paxos Lamport [1989], and Liskov et al. [1991]

Earlier techniques to tolerate Byzantine faults were inefficient and could misclassify replica as faulty.

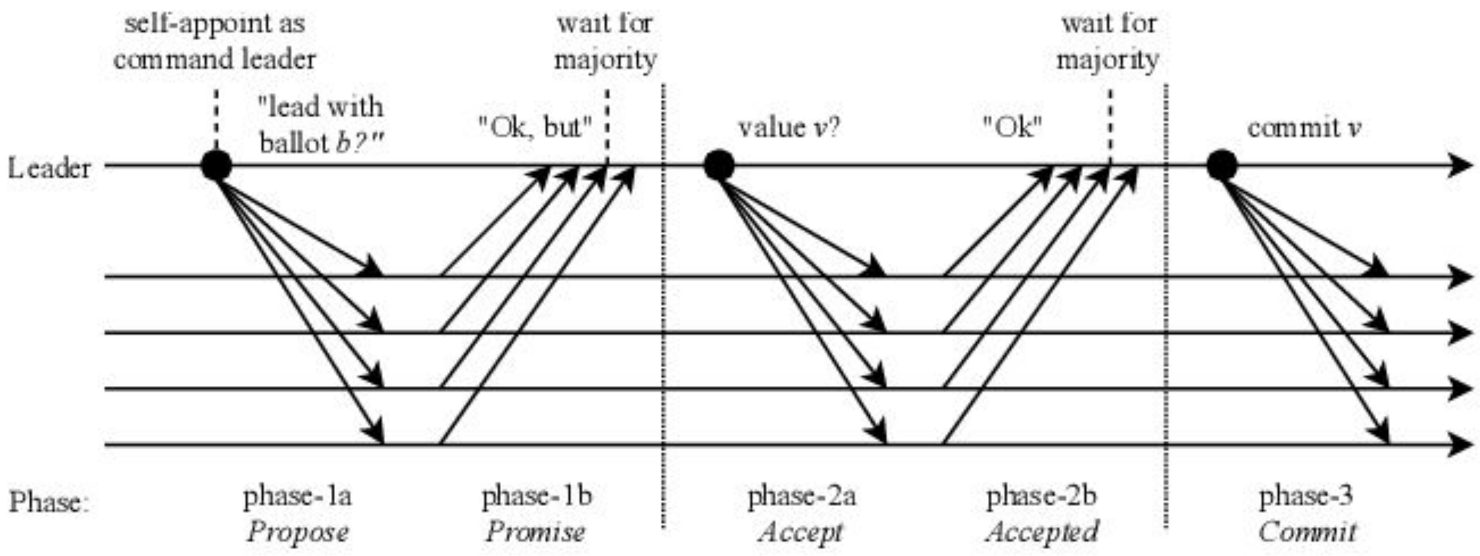
- synchrony [Lamport 1984] rely on bounds on message delays and process speeds

Earlier proactive security algorithms assume that program is in read only memory and non compromisable with authenticated channel persisting between replicas even after a compromise

- [Ostrovsky and Yung 1991; Herzberg et al. 1995, 1997; Canetti et al. 1997; Garay et al. 2000]

## Paxos

Guarantee safety, but not liveness



Source IEEE 2018 [5]

## Byzantine Fault Tolerance

BFT is a state machine replication algorithm that is safe in asynchronous systems such as the Internet

Used to build highly available systems and incorporates mechanisms to defend against Byzantine-faulty clients

- Safety
  - Never returns bad replies even in the presence of denial-of-service attacks.
- Liveness
  - provided message delays are bounded eventually
- Recovers replicas proactively
  - provided fewer than 1/3 of the replicas become faulty within a small window of vulnerability

## System Model

- Deterministic Asynchronous distributed system
- No Impersonation : Public-key signature using cryptographic hash function to compute message digests
- Non Tampering : uses message authentication codes (MACs) to authenticate all messages

Bound on Faults

f = |_(n−1)/3 _| is the bound on number of faulty replicas

Strong cryptography

assume the attacker cannot forge MAC

assume the cryptographic hash function is collision resistant

Weak Synchrony (Only for Liveness)

assume that delay(t) has an asymptotic upper bound

## BFT algorithm

without proactive recovery

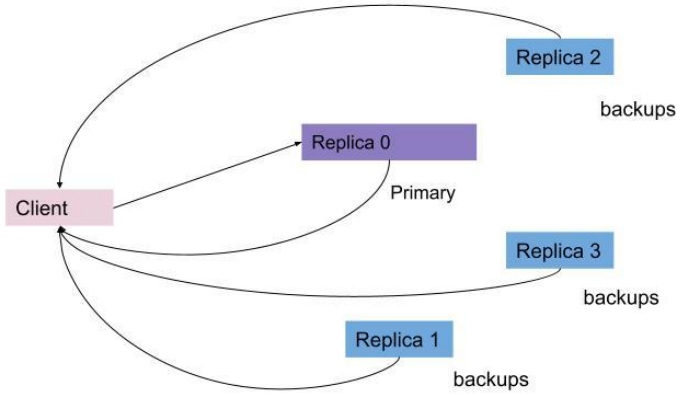
## Overview : BFT

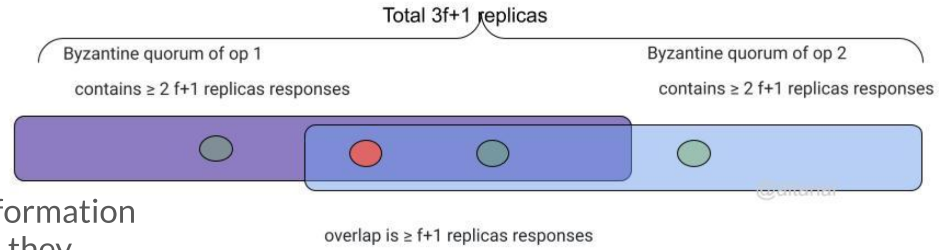
Primary-backup

- Primary picks ordering and sends assignment to backups
- Backups check sequence
    - request sequence numbers are dense, no skipping

Quorum replication

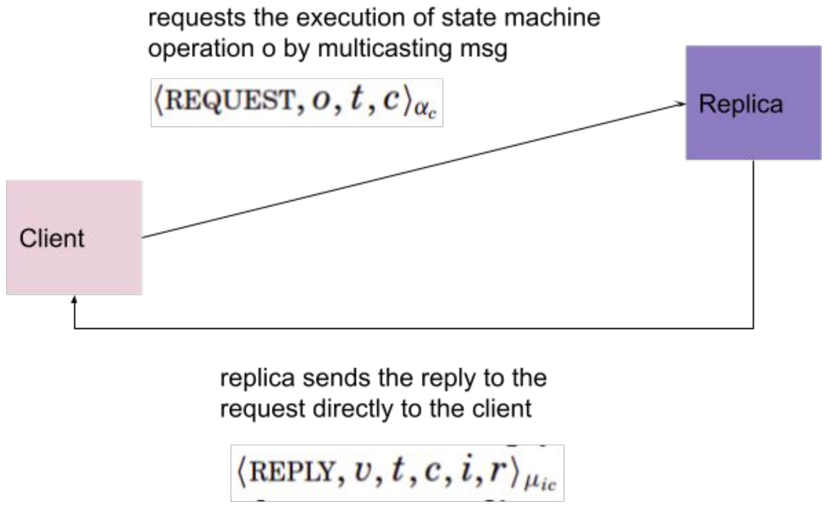
- Order requests correctly despite failures
    - Quorums : Reliable memory for protocol information
    - Replicas write information to a quorum and they collect quorum certificates
- Intersection
- Availability




Total 3f+1 replicas
Byzantine quorum of op 1
contains ≥ 2 f+1 replicas responses
Byzantine quorum of op 2
contains ≥ 2 f+1 replicas responses
overlap is ≥ f+1 replicas responses

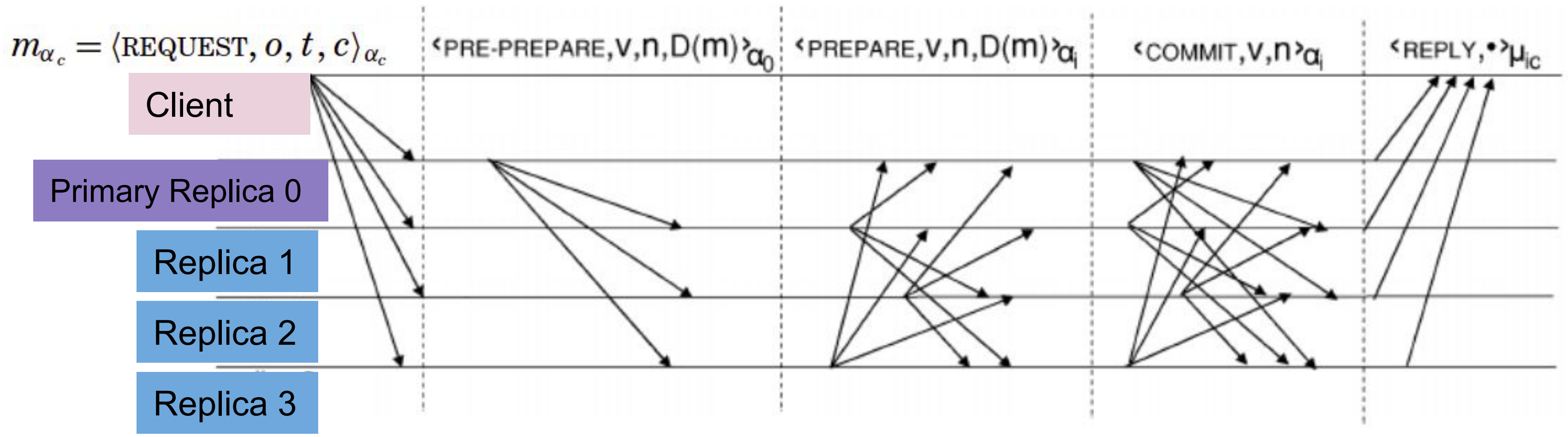
## BFT

Client waits for a weak certificate with **f +1 replies** with valid MACs from different replicas, and with the same Timestamp t and result r, before accepting the result.


requests the execution of state machine operation o by multicasting msg
$\langle \text{REQUEST}, o, t, c \rangle_{\alpha_c}$
Replica
Client
replica sends the reply to the request directly to the client
$\langle \text{REPLY}, v, t, c, i, r \rangle_{\mu_{ic}}$

## BFT Normal Case Op

Atomically multicast requests to the replicas using three-phase protocol (pre-prepare, prepare, and commit)



$m_{\alpha_c} = \langle \text{REQUEST}, o, t, c \rangle_{\alpha_c}$
$\langle$PRE-PREPARE,v,n,D(m)$\rangle_{\alpha_0}$
$\langle$PREPARE,v,n,D(m)$\rangle_{\alpha_i}$
$\langle$COMMIT,v,n$\rangle_{\alpha_i}$
$\langle$REPLY,•$\rangle_{\mu_{ic}}$
Client
Primary Replica 0
Replica 1
Replica 2
Replica 3

When primary fails, backup replicas can trigger view changes to select a new primary

Low h and high H watermark for sequence number help in Garbage collection to prevents a faulty primary from exhausting the space of sequence numbers

## Proactive recovery mechanism for BFT ( BFT-PR)

Recovers replicas periodically even if there is no reason to suspect that they are faulty.

## Overview : BFT PR

Assumptions

- Secure Cryptography
- Read-Only Memory : memory storing public keys survives failures, without being corrupted
- Watchdog timer , hands control to a recovery monitor

Quorum certificate received by a non faulty replica must be backed by a quorum

changing MAC keys during recoveries

- replicas and clients reject messages that are authenticated with old keys or not part of complete certificate

Refreshing session keys

Recovery and reboot

- Estimation Protocol for HM
- multicasts a recovery request
- Check and fetch state

## Limitations

1. Does not address the problem of fault-tolerant privacy: a faulty replica may leak information to an attacker.
2. assumes static membership
3. assumes that a replica server's memory acts as a stable, persistent storage.
4. Strict upper bound on faulty processes ( only fewer than 1/3 of the replicas can fail ie strictly > ⅔ of the total number of processors should be honest.)
5. High Communication overhead - increases exponentially with every new node added
6. Susceptible to Sybill attack in p2p system [4]

## Advantages

1. Does not rely on client to order or synchronize
2. Better in energy efficiency than PoW ( proof of Work) in Bitcoin where every node individually verifies all the transactions
3. Detection of denial-of-service attacks
4. Garbage collection

## Applications :

1. Byzantine-fault-tolerant NFS file system using symmetric cryptography to authenticate messages( implemented in paper itself)
2. pBFT in distributed computing and blockchain
   a. pBFT + DPoS(Delegated Proof-of-Stake)
   b. pBFT in combination with PoW consensus

## References

[1] Paxos - Leslie Lamport [1998] [Lampson, 1996; Prisco et al., 1997;Lamport, 2001; van Renesse and Altinbuken, 2015]

[2] CAP theorem - Gilbert and Lynch [2002]

[3] state machine replication [Lamport 1978; Schneider 1990]

[4] Sybil Attack - John R. Douceur at the Microsoft Research.

[5] Charapko, Aleksey, Ailidani Ailijiang and Murat Demirbas. “Bridging Paxos and Blockchain Consensus.” 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (2018): 1545-1552.

[6] Core Concepts, Challenges, and Future Directions in Blockchain: A Centralized Tutorial , JOHN KOLB, MOUSTAFA ABDELBAKY, RANDY H. KATZ, and DAVID E. CULLER, University of California – Berkeley, USA, ACM Comput. Surv., Vol. 53, No. 1, Article 9, Publication date: February 2020.