Practical Byzantine Fault Tolerance

Ira A. Fulton Schools of Engineering

Arizona State University

Objectives

Objective

Describe several consensus algorithms at a fundamental level

Objective

Identify trade-offs associated with consensus algorithms

From Public to Permissioned Blockchains

- Consensus algorithms for open and public blockchains
  - Defenses against sybil attacks
  - Aligning incentives (due to voluntary mining)
  - Latency issues
- Consensus algorithms for permissioned blockchains
  - Nodes within network identified, trusted, and verified
  - Often direct connection made between all/most nodes
    - Communication latency is minimized

Practical Byzantine Fault Tolerance - Introduction (1/2)

- Introduction in 1999 paper by Miguel Castro and Barbara Liskov
- Classic consensus algorithm designed for asynchronous environments
  - Distributed internet applications
  - Now used for blockchain networks
- Byzantine failure model
  - Additional assumption of independent node failures

Practical Byzantine Fault Tolerance - Introduction (2/2)

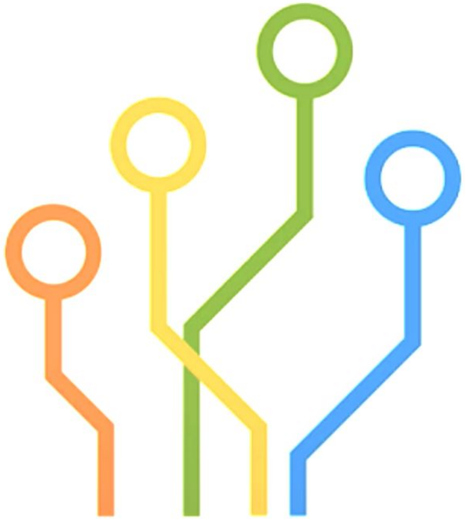
- Requires minimum of 3*f* + 1 nodes
  - Provides safety and liveness for up to *f* faulty nodes
- A form of **state machine replication**
  - The state machine is replicated across different nodes in a distributed system.
  - Each node maintains a copy of the state and implements operations.

PBFT - How it Work

Moves through phases called **views**

- Numbered consecutively
- One node denoted as **primary**, other nodes are **backups**

1. A client sends a request to invoke a service operation to the primary
2. The primary multicasts the request to the backups
3. All nodes execute the request and send a reply to the client
4. The client waiting for f + 1 replies from different replicas with the same result - the result of the operation



PBFT Blockchain Implementation (1/3)

- Nodes Connected in distributed network
  - Nodes directly connected with each other
- Can submit a transaction to any of the nodes
  - That node will propagate the transaction to other peers
- Primary node verifies and orders the propagated transaction
  - Verifies and orders propagated transactions submitted in block time interval
  - Sends the newly created block to all of the other nodes

PBFT Blockchain Implementation (2/3)

Nodes receive the block and proceed as follows:

1. Every node executes the transactions within the block in order and verifies their validity

2. Once all transactions are successfully executed, each node calculates the hash of the block

3. Nodes broadcast the block hash to all the other nodes

4. Once a node receives and confirms two-thirds of its peers sent the same block hash, it commits the new block to its local copy of the blockchain

PBFT Blockchain Implementation (3/3)

- A lot of direct communication between the nodes
  - Works best in environments with small consensus group sizes and high speeds– permissions/private blockchain networks

Unit Recap