Consensus Protocols

Saravanan Vijayakumaran

Associate Professor
Department of Electrical Engineering
Indian Institute of Technology Bombay

March 14, 2024

Consensus

- Informally, consensus protocols enable multiple computers connected by a network to be in sync
  - The computers are called **nodes**
  - Network may be unreliable (packet drops, delays)
  - Some nodes may act maliciously (deviate from protocol)
- Assumptions
  - Semi-reliable point-to-point communication between nodes
  - Secure digital signatures are available
- State Machine Replication (SMR)
  - First studied in the 1980s
  - **Clients** submit transactions to one or more nodes
  - Each node maintains a local append-only data structure representing an ordered sequence of transactions (**history**)
  - **Goal**: All nodes must have identical local histories
  - If all nodes start from same initial state, then applying the transactions in the same order will result in the same final state
- SMR protocol requirements
  - **Consistency**: All nodes agree on the same history
  - **Liveness**: Every valid transaction submitted by a client is eventually added to the history

SMR in Synchronous Setting with Honest Nodes

SMR in Synchronous Setting with Honest Nodes

- **Assumption 1**: Permissioned Network
  - Set of nodes running the protocol is fixed and known
  - Let the nodes be denoted by $\{1, 2, \dots, n\}$
- Blockchain networks represent a permissionless setting. Why study the permissioned setting?
  - Impossibility results in permissioned setting automatically apply to the harder permissioned setting
  - Permissionless consensus protocols use ideas from the permissioned setting
- **Assumption 2**: Public Key Infrastructure
  - Each node has a public key which is known to all other nodes
- **Assumption 3**: Synchronous Network
  - Shared global clock: If time is broken into time steps, then all nodes agree on which time step they are currently in
  - Bounded message delays: A message sent at time step t arrives before the beginning of time step $t + 1$ (msg reordering is possible)
- **Assumption 4**: All nodes are honest
  - All nodes run the protocol without deviations or errors

Two Faulty SMR Protocols

- A faulty SMR protocol
  - Nodes do not communicate with each other
  - As soon as a node receives a client request, it adds the transaction to its local history
  - If clients submit transactions to all nodes at the same time, then local histories will be identical
  - Protocol fails to guarantee consistency if
    - A client submits requests only to a strict subset of the nodes
    - Or if requests arrive in different orders at different nodes
- Another faulty SMR protocol
  - One of the nodes is designated as the leader (say node 1)
  - At the beginning of each time step, the leader sends an ordered list of transactions it knows about to all nodes
  - Each node (including the leader) appends this list of transactions to its history
  - Consistency is achieved as all local histories will be identical
  - Liveness is not achieved if client submits a request to a non-leader node

SMR via Rotating Leaders

- In each time-step, a node is designated as the leader in round-robin fashion
- In time step t, the leader is the node with ID equal $1 + (t \bmod n)$
- At the beginning of each time step, the leader sends an ordered list of **new** transactions it knows about to all nodes
  - new = Transactions not yet added to the history
  - Empty list is also allowed
- Before the beginning of the next time step, each node (including the leader) appends this list of transactions to its history
- Consistency is achieved as all local histories will be identical
- Liveness
  - Suppose a client submitted a request to node i
  - Node i will eventually become the leader in a time step
  - If the client's transaction has not been included so far, it will be sent by node i to all other nodes

The Byzantine Broadcast Problem

Faulty/Byzantine Nodes

- The honest nodes assumption is unrealistic
- Types of faults
  - **Crash faults**: A node stops working at some time t
  - **Omission faults**: A node does not send a subset of the messages it is supposed to send
  - **Byzantine faults**: A node can deviate from the protocol in an arbitrary manner
    - Cannot forge digital signatures
- Byzantine?
  - Istanbul was known as Byzantium in the past
  - In a 1982 paper, Lamport, Shostak, and Pease introduced a consensus problem using a story of Byzantine army generals
- **Relaxed Assumption 4**: Number of Byzantine nodes $\leq f$
  - All honest nodes assumption corresponds to $f = 0$
  - Parameter f is assumed to be known (protocol description can depend on it)
  - Identities of the (at most f) Byzantine nodes are not known; but set of Byzantine nodes is fixed
  - Even with $f = 1$, the SMR via rotating leaders protocol fails

Byzantine Broadcast Problem

- A single-shot consensus problem
- Any solution to BB can be combined with the rotating leaders idea to solve SMR
  - Need to detect that the leader in current time step is Byzantine
- Setting of Byzantine broadcast
  - One of the n nodes is the **sender** (who may be Byzantine)
  - The identity of the sender is known to all nodes in advance
  - The sender has a **private input** v^* which belongs to some set V
- Desired properties of a Byzantine broadcast protocol
  - **Termination**: Every honest node i eventually halts with some output $v_i \in V$
  - **Agreement**: All honest nodes halt with the same output (even if sender is Byzantine)
  - **Validity**: If the sender is an honest node, then the common output of the honest nodes is the private input v^* of the sender
- Agreement is a safety property similar to consistency
- Termination and validity together are similar to liveness
- Termination + validity or termination + agreement are easy to achieve; getting all three properties is challenging

SMR Reduces to Byzantine Broadcast

- **Idea**: Use rotating leaders and in each time step invoke BB with current leader as sender
  - Many blockchain consensus protocols work by reducing multi-shot consensus to single-shot consensus
- Suppose π is a protocol for BB that terminates in at most T time steps while satisfying agreement and validity
- At each time step in $0, T, 2T, \ldots$
  - Define the current leader using round-robin assignment
  - The leader constructs an ordered list L^* of all not-yet-included transactions that it has heard about
  - Invoke π with leader as sender and L^* as private input
  - When π terminates, every node i appends its output L_i in the BB to its local history
- If π requires at most f nodes to be Byzantine, then the SMR protocol above satisfies consistency and liveness for same bound f
  - Small modification of liveness definition: Only client requests submitted to honest nodes need to be eventually added to the history

Protocol for BB when $f = 1$

- **Canonical strategy by Byzantine nodes:** Send conflicting messages to different nodes
- Honest nodes need to perform cross-checking to detect conflicting messages
- Simple Cross-Checking Protocol for Byzantine Broadcast
  - In the first time step, sender sends it private value v^* to all non-senders along with its digital signature
    - Let m_i be the message (including signature) which the sender sent to non-sender i
  - In the second time step, every non-sender signs the message m_i and sends the message-signature pair to all other non-senders
    - If an honest non-sender does not receive a message from the sender in the first time step, it continues as if it received a null value $\perp$
  - In the third time step, each non-sender uses the majority rule to choose its output from the messages it received from the sender and other non-senders
    - Ties are broken in a consistent manner, such as lexicographic ordering
    - Sender outputs its private input v^*
- **Proposition:** When $f = 1$, the above protocol satisfies termination, agreement, and validity

Protocol for BB when $f = 1$

- **Proposition:** When $f = 1$, the protocol satisfies termination, agreement, and validity
- Termination: Every honest node halts after 3 steps with an output
- Agreement
  - Honest sender
    - All honest non-senders receive v^* from the sender
    - The Byzantine non-sender cannot forge sender's signature on v^* (it can only induce an omission fault)
    - All honest non-senders receive at least $n - 2$ votes for v^*
  - Byzantine sender
    - If the sender is Byzantine, all non-senders are honest
    - At the start of the third time step, all non-senders have exactly the same information (the set of messages sent by the sender to different nodes)
    - The result of majority voting will be the same at all non-senders (it could be $\perp$)
- Validity
  - Only need to care about honest sender case
  - All honest non-senders receive at least $n - 2$ votes for v^*

Protocol Fails for $f = 2$

- Suppose the sender and a non-sender are Byzantine
- Suppose the number of nodes n is even and $n \geq 4$
- Let the set of possible values V contain 0 and 1
- In the first time step
  - Byzantine sender sends 0 to $\frac{n}{2} - 1$ honest non-senders and 1 to the other $\frac{n}{2} - 1$ honest non-senders
  - Sender also shares both the 0 and 1 messages (including its signatures) with Byzantine non-sender
- In second time-step
  - Byzantine non-sender echoes the 0 message to the first group of honest non-senders
  - It echoes the 1 message to the second group of honest non-senders
  - Honest non-senders echo the message they received from sender
- In the third time-step
  - Half the honest non-senders will have received $\frac{n}{2}$ votes for 0 and $\frac{n}{2} - 1$ votes for 1
  - The other half of honest non-senders will have received $\frac{n}{2} - 1$ votes for 1 and $\frac{n}{2}$ votes for 0
  - The two groups will output different values, violating agreement

The Dolev-Strong Protocol

- Proposed in 1983 as a solution to Byzantine broadcast problem
- Works for any f in the permissioned, PKI, synchronous setting
- **Definition of Convincing Messages**: A node i is **convinced of value v at time step t** if it receives a message prior to that time step that:
  - references the value v,
  - is signed first by the sender,
  - is signed by at least $t - 1$ other distinct nodes
- Protocol description
  - **Time step 0**: Sender sends its private input v^* along with its signature to all non-senders and outputs v^*
  - **Time steps $t = 1, 2, \ldots, f + 1$**: If a non-sender is convinced of a value v by a message m prior to this time step and had not previously been convinced of v, it signs m to get a signature s and sends (m, s) to all other non-senders
  - **Final output**: If a non-sender is convinced of exactly one value v, it outputs v. Otherwise, it outputs $\perp$
- **Theorem**: The Dolev-Strong protocol satisfies termination, validity, and agreement for any number of Byzantine nodes f

FLM Impossibility Result

- Established first by Pease, Shostak, and Lamport (1980)
- Named after Fischer, Lynch, Merrit (1986) who gave a nice proof
- Shows that the PKI Assumption is necessary in the Dolev-Strong protocol
- Consider the synchronous, permissioned, non-PKI setting
- If $f \geq \frac{n}{3}$, there is no Byzantine broadcast protocol that satisfies termination, agreement, and validity.
- Proof: Lecture 3 of Tim Roughgarden's Foundations of Blockchains course

Asynchronous Network Model

Asynchronous Network Model

- The synchronous network model
  - Shared global clock
  - Every message sent at time step t arrives by time step $t + 1$
  - Unrealistic for modeling the Internet (outages, DoS attacks)
- Assuming known bounds on message delay and clock drifts again leads to the synchronous model
- The asynchronous network model
  - No shared global clock
  - No bound on the maximum message delay
  - Every message sent arrives eventually
- Can we have consensus protocols in the asynchronous network model?

The FLP Impossibility Theorem

- Named after Fischer, Lynch, Paterson
- Applies in the permissioned, PKI, asynchronous setting
- Byzantine Agreement
  - Single-shot consensus problem
  - No sender node
  - Node has a private input $v_i \in V$
- Desired properties of a Byzantine agreement protocol
  - **Termination**: Every honest node i eventually halts with some output $w_i \in V$
  - **Agreement**: All honest nodes halt with the same output (no matter what the private inputs are)
  - **Validity**: If $v_i = v^*$ for every honest node i, then $w_i = v^*$ for every honest node i
- **FLP Impossibility Theorem**: For every $n \geq 2$, even with $f = 1$, no deterministic protocol for the Byzantine agreement problem satisfies termination, agreement, and validity in the asynchronous model
- Randomized protocols for BA in the asynchronous model exist (for e.g. HoneyBadgerBFT)
- Practical blockchain protocols escape the FLP impossibility theorem by relaxing the asynchronous assumption

Partially Synchronous Network Model

Partially Synchronous Network Model

- Lies between the synchronous and asynchronous models
- All nodes share a global clock
- Message delays are arbitrary up to some **unknown** time instant called the global stabilization time (GST)
- After GST, message delays are bounded by a known value Δ
- Message delivery model
  - If message is sent at $t \leq GST$, then it is received at or before $GST + \Delta$
  - If message is sent at $t \geq GST$, then it is received at or before $t + \Delta$
- Goals of consensus protocols in the partially synchronous model
  - Safety properties must always hold (even pre-GST)
  - Liveness properties must eventually hold (possibly only after GST)
- **Theorem**: There exists a deterministic protocol for the Byzantine agreement problem that satisfies agreement, validity, and eventual (post-GST) termination in the partially synchronous model if and only if $f < \frac{n}{3}$
  - Tendermint and PBFT are protocols that achieve the "if" direction

The 33% Threshold

- Reasoning for 33% threshold
  - Honest nodes can only wait for messages from $n - f$ nodes before taking action
  - But there may by Byzantine nodes in the $n - f$ nodes which responded (honest nodes messages may be delayed)
  - A strict majority of the $n - f$ nodes must be honest

$$f < \frac{1}{2}(n - f) \implies f < \frac{n}{3} \text{ or } n \geq 3f + 1$$

- Proof of "only if" direction in the special case of $n = 3$ and $f = 1$
  - Suppose there exists a protocol π for BA that satisfies agreement (always), validity (always), and termination (eventually)
  - Suppose the three nodes are Alice, Bob, and Carol
  - Bob is Byzantine
  - Alice has a private input 1 and Carol has private input 0
  - All messages between Alice and Carol are delayed
  - Bob sends 1 to Alice and none of Carol's messages
  - Bob sends 0 to Carol and none of Alice's messages
  - Alice will output 1 and Carol will output 0 to satisfy termination and validity
  - Agreement is violated

The CAP Principle

- A well-known result about distributed systems
- **Consistency**: All nodes agree on the same history
    - A consistent distributed system looks like a centralized system to a client
- **Availability**: A client request to the distributed system should be eventually executed
    - Example: A client may query a distributed database. Response should be eventually delivered
- **Partition Tolerance**: Consistency and availability should hold in the presence of a network partition
- **CAP Principle**: No distributed system can have all three properties
    - Network partitions are unavoidable in practice
    - Protocols have to choose between availability (liveness) and consistency (safety)
- Tendermint/PBFT give up on liveness (shared history is frozen during a network partition)
- Bitcoin's heaviest chain rule gives up on consistency (long reorgs of block history are possible)

References

- Foundations of Blockchains: Video lectures by Tim Roughgarden
- Notes of lectures 1 to 6 from 2021 FoB course
  - https://timroughgarden.github.io/fob21/l/l1.pdf
  - https://timroughgarden.github.io/fob21/l/l2.pdf
  - https://timroughgarden.github.io/fob21/l/l3.pdf
  - https://timroughgarden.github.io/fob21/l/l4-5.pdf
  - https://timroughgarden.github.io/fob21/l/l6.pdf
- M. Pease, R. Shostak, and L. Lamport. *Reaching agreement in the presence of faults*. Journal of the ACM, 1980
- M. J. Fischer, N. A. Lynch, and M. Merritt. *Easy impossibility proofs for distributed consensus problems*. Distributed Computing, 1986.
- M. J. Fischer, N. A. Lynch, and M. S. Paterson. *Impossibility of distributed consensus with one faulty process*. Journal of the ACM, 1985
- Blue eyes puzzle https://www.explainxkcd.com/wiki/index.php/Blue_Eyes
- HoneyBadgerBFT https://eprint.iacr.org/2016/199