UCSB CS 291D: Blockchains and Cryptocurrencies **Fall 2020**

Lecture 4: Nakomoto Consensus and the Longest Chain Protocol

Lecturer: Shumo Chu *Scribes: Daniel Shu, Karl Wang, Chen Zhu*

4.1 Blockchain Protocol From Byzantine Protocol (BPBB)

4.1.1 Desired Properties

4.1.1.1 Consistency

All nodes must have the same view of transactions log. However, it is very hard to achieve consensus among distributed system due to some network problems such as Network Partition and Network Latency.

- LOG_k^s denotes the log at node k after round s
- For any honest node i, j and any round t, r where $(t \leq r)$, $LOG_i^t \leq LOG_j^r$
- $LOG_i^t \preceq LOG_j^r$

4.1.1.2 Liveness

Suppose that a honest node receives transaction txs in round r, then trx will appear in every honest node's log within a bounded round.

- T_{conf} Liveness: If an honest node receives a transaction t_x in some round r, then by the end of $t_{conf} + r$ round, all honest nodes have t_x in their logs

4.1.1.3 Additional Notes

- Log is append only
- If the protocol is deterministic, then guarantees are deterministic
- If the protocol is randomized (for performance purpose), a high probability guarantee is wanted
  - $Pr[fail] \leq negl(\cdot)$ over the randomness of the protocol

4.1.2 Blockchain Protocol

4.1.2.1 Protocol Steps

Byzantine Broadcast (BB): R rounds to reach consensus among n nodes

- In every kR round $(k = 0, 1, 2, ...)$

- Spawn a new BB instance
- Leader Election: $L_k := (k \bmod n)$
- L_k collects all transactions txn ($txn \notin LOG$) and proposes m_k in this round of BB
- At any time, if you have a node that finished $BB_0, BB_1, ..., BB_{k'}$ with messages $m_0, m_1, ..., m_{k'}$, then log would be updated as $m_0||m_1||...||m_{k'}$

Theorem 4.1 *If BB can tolerate up to f corrupted nodes and reach consensus within r rounds, BPBB is consistent and has O(Rn)-liveness for some n, R*

Proof: Consistency and Liveness are maintained for this protocol.
Given that BB is consistent, if the first instance of BB is consistent, then by the end of round r, all nodes have the same view because each round is consistent. This can be proved by induction.
Prove $O(Rn)$-liveness. For a transaction tx, if it is proposed in round r, then it must appear in everyone's log in $O(n)$ BB instances.
For a single tx, it takes at most $n + 1$ BB instances before it is added to the log. This is because the nodes take turns in a Round Robin to be leader, so if tx arrived at node i right after the BB already started, then it must wait n + 1 instances before tx can be added to the log. ■

4.1.2.2 Problems with BPBB

- Rn could be very huge
- If we want the blockchain to be permissionless, then it is vulnerable to a Sybil Attack because a malicious party can create many accounts

4.2 Block format and notations

When discussing about blockchain, we will refer to blocks and prefixes using a syntax similar to Python.

- We use chain to denote the blockchain maintained by the node.
- We use $\mathsf{chain}[0]$ to denote the first block of the blockchain, also called the *genesis block.* [1]
- We use $\mathsf{chain}[i], i > 0$ to denote the ith block of the blockchain, where each one is a 4-tuple $(h_{-1}, \eta, \mathsf{txs}, h)$
  - h_{-1} is a a hash pointer to the previous block.
  - η is a solution to the mining puzzle.
  - txs is the transactions.
  - h is hash of the current block.
- We use $\mathsf{chain}[-1]$ to denote the last block
- We use $\mathsf{chain}[-i], i > 1$ to denote the ith to last block
- We use $\mathsf{chain}[: l]$ to denote the prefix $\mathsf{chain}[0, ..., l]$
- We use $\mathsf{chain}[: -l]$ to denote the prefix of chain except for the last l blocks

[1] Fun fact: the genesis block for Bitcoin contains the following text: "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks". This is possibly a comment on the instability of traditional banking systems.

4.3 Mining (Proof of Work)

Mining is a process that that helps determine who can propose the block for each round. In BPBB, we used round robin. However, that cannot be used in a distributed environment. Furthermore, we have to solve Sybil attack.

The intuition for mining is that we do not want corrupted nodes to become the sender. However, it is impossible for a decentralised protocol to distinguish the corrupted nodes from the honest nodes. The best we can get is if we assume that the majority of node is honest. Then, by selecting a fair random sample, the chance of selecting an honest node is proportional to number of honest nodes.

To avoid Sybil Attack, if the cost of creating an account or participating in the consensus is really cheap, then the attacker can create a lot of accounts. Therefore, there should be some form of entry barrier to the consensus protocol.

Combining these two ideas together, we get the intuition for mining for the bitcoin protocol.

4.3.1 Detailed Algorithm

1. Let the last block be $(\_, \_, \_, h^*)$. For a proposed next block to be valid, the miner must solve the puzzle to find a random bit sequence η, such that $H(h^*, \eta, txs) \leq D_p$.
2. D_p is a congestion control parameter that is adjusted as part of the protocol. If there are a lot of congestion, then the time between block will be much longer. In that case, the protocol will increase D_p, which makes this puzzle much easier to solve. H is a collision resistant hash function - in Bitcoin it's SHA256.
3. The time to solve this puzzle corresponds with the computational power of the miner. It's hard to reverse engineer the solution because
   (a) H is a random oracle.
   (b) Since it requires hash of the most recent block, attacks based on pre-computation are not possible. Unless the hash function is broken, the best the attacker can do is brute force.

4.4 Longest Chain Protocol

Even with D_p , there is still a chance that multiple miners solve the problem in the same time. Thus, in bitcoin, there is an additional rule that miners should always mine from the longest chain it has seen. When there are two different head of the chain, the miner will always be incentivised to mine the longest chain. All but the last K blocks are final. K is defined by users and not the protocol. The larger the K, the more security the user will have. In bitcoin, the common standard is 6.

4.4.1 Attacks

Let's suppose an attacker wants to get an item for free. The attacker would fist purchase the item, then wait for K blocks to be appended to the chain so that the transaction is confirmed. After receiving the item, the attacker would then try to fork the chain with blocks that does not contain the transaction. If the new chain becomes the longest, then the attacker would have essentially reversed the transaction and got the item for free. However, with the mining puzzle, this is really hard. Attacker would have to mine K more blocks than

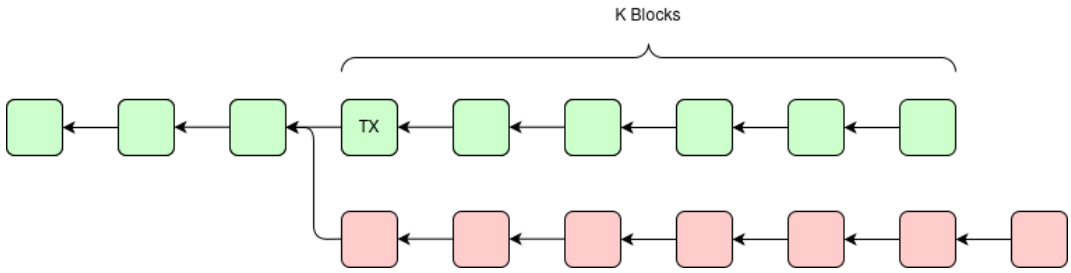

Figure 4.1: a successful attack needs to create a longer chain that does not contain the transaction TX

the honest nodes in a short time window. Since the probability of solving the puzzle is proportional to the computational power, the attacker would need to have more computational power than the computational power of all honest nodes combined.

4.5 Nakamoto Consensus Algorithm

Regarding Nakamoto's Blcokchain Implementation, the following a few rules must be observed

- Each node always tries to mine transaction blocks off the **Longest Valid Chain**.
- When a new block is mined, the miner node must append it to others.
- When a host node receives a fresh message or transaction in round r, it must echo the message or transaction and all honest node will have observed it by round $r + \Delta$.

4.5.1 Detailed Algorithm

1. A new node decide to participate and it starts from the initial chain's genesis block
   - $chain := (0, 0, \perp, H(0, 0, \perp))$
2. Whenever a node receives a message $chain'$ from the network, the node will validate the chain and replace the known chain if needed.
   - if $valid(chain') \wedge (|chain'| > |chain|) \longmapsto chain := chain'$
3. In each round, all nodes try to mine a block from the longest chain.
   - Previous chain is denoted as $chain[-1] := (\_, \_, \_, h_{-1})$
   - A new txs is picked from the transaction pool
   - A Difficulty Dynamic Parameter, D_p, is set for this round
   - A random seed $y \in \{0, 1\}^\lambda$ is picked to compute the hash $h = H(h_{-1}, y, txs)$
     - if $h < D_p$, append the newly minded block (h_{-1}, y, txs, h) to the chain and broadcast $chain||(h_{-1}, y, txs, h)$
     - else, go back to $step1$ and try again
4. $chain[: -K]$ is considered as final
   - For example, the longest chain that a node observed so far removing the last K blocks, for which K is set as sufficiently large to maintain consistency

References

[1] "The computational power in Bitcoin" *https://bitinfocharts.com/comparison/bitcoin-hashrate.html*