Lightweight Consensus Algorithm

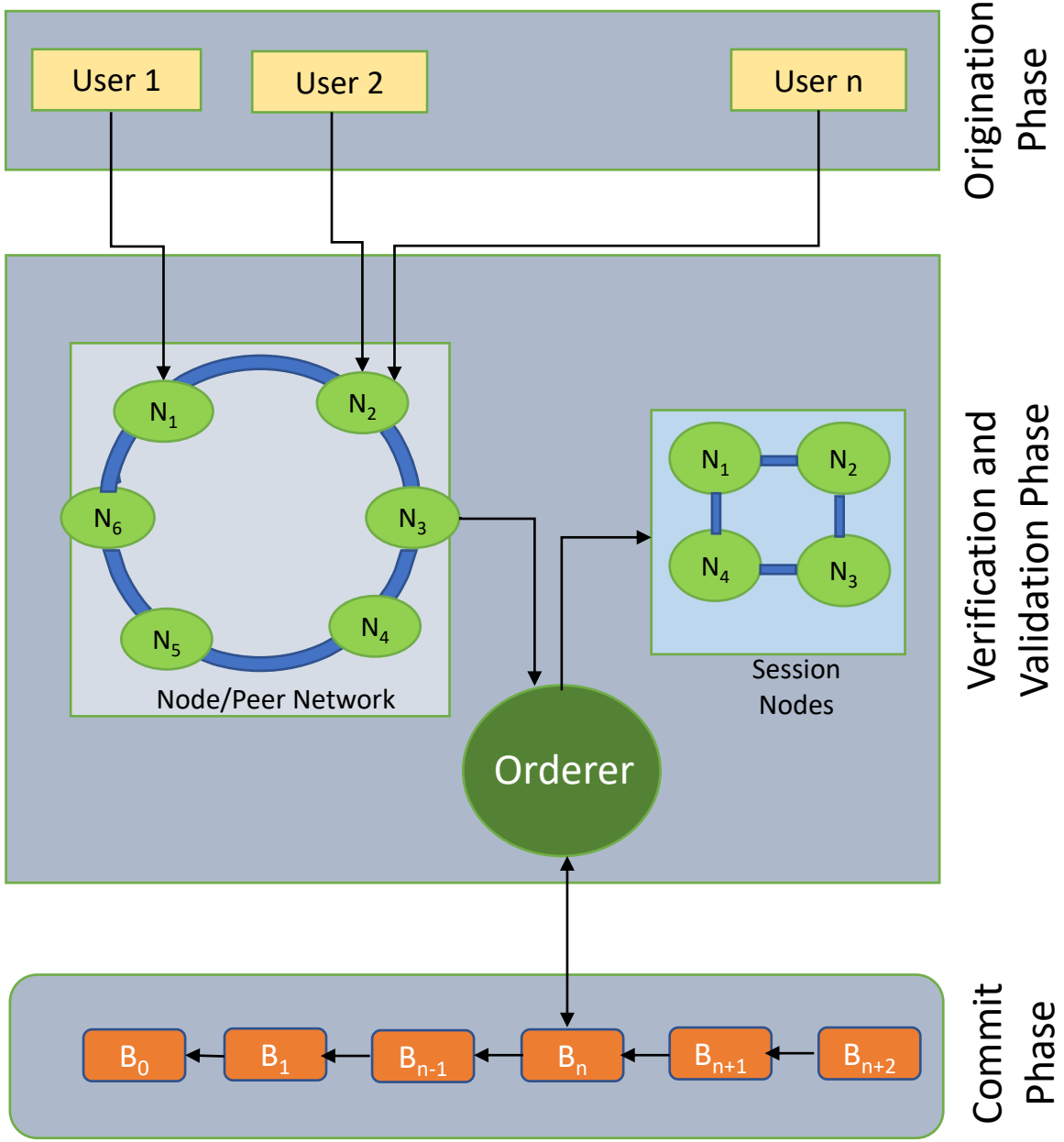
Department of IT, NITK, Surathkal

Guided by,
Dr. Kiran Manjappa

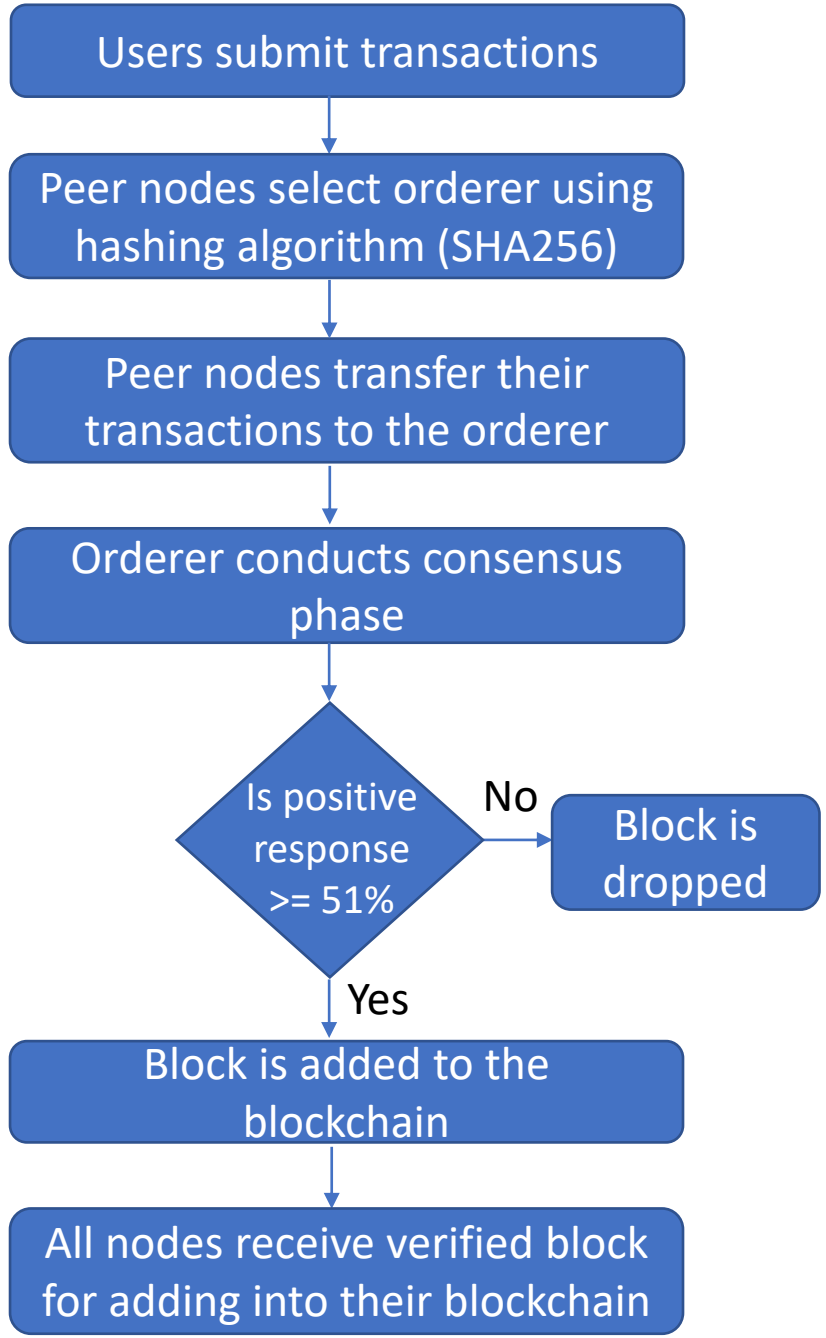
Presented by,
222IT018 - Pankaj Kumar Magar
222IT034 - Vedant Parwal

Overview

- The entire framework consists of Peer nodes, session nodes, orderer, Admin and Key distribution center (KDC).
- Each peer node is linked with key distribution center (KDC) which is responsible for providing keys, signature to all the entities in the network.
- Admin is a trusted authority whose credentials like signature and keys are generated when the network is instantiated.
- Peers are specialized nodes that have enough resources to execute consensus algorithm and maintain the distributed ledger.
- Orderer is one of the peer node, which is responsible for grouping all approved transactions into a newly generated block.



Framework and processes of the proposed system



Working of Consensus algorithm

Announce:

- Peer nodes that are interested in functioning as orderers generate a orderer interest transaction (t_0) that is structured as

  <timestamp, pk, sign>

  where,

  pk is the public key (should be verified by admin) to protect sybil attack, where a single node pretends to be multiple nodes by generating multiple t_0.

  sign is the signature corresponding to public key.

  timestamp the time at which transaction is created.

- t_0 is broadcasted to the network. Any t_0 that is generated after some $ð_1$ is discarded by the network.

Calculate:

- Peer nodes verify public key with KDC and matches sign with pk. The potential validators then calculate Key weight metric (KWM) which is the sum of the numerical weight corresponding to each value of hash function (SHA256) output applied on the t_0 transaction.
- The weight corresponding to each character in the hash function output is extracted from a key weight metric (KWM) dictionary.
- All participating nodes apply the same KWM dictionary to ensure that all have the same view of KWM. Admin populates the KWM dictionary.
- The one with highest KWM value is selected as orderer to create the candidate block.
- At the end of each session, peer nodes need to use a new public key. This make sure that the process of selecting orderer remains randomized.

Agree:

- Each participating node agree on the selection of the orderer.

Transaction verification

Users send transaction to the peer nodes:

- The transaction and its private key are inputted in a sign function in order to create a personal signature.
- After that, the sender sends to the peer node a transaction and his signature created by the algorithm.

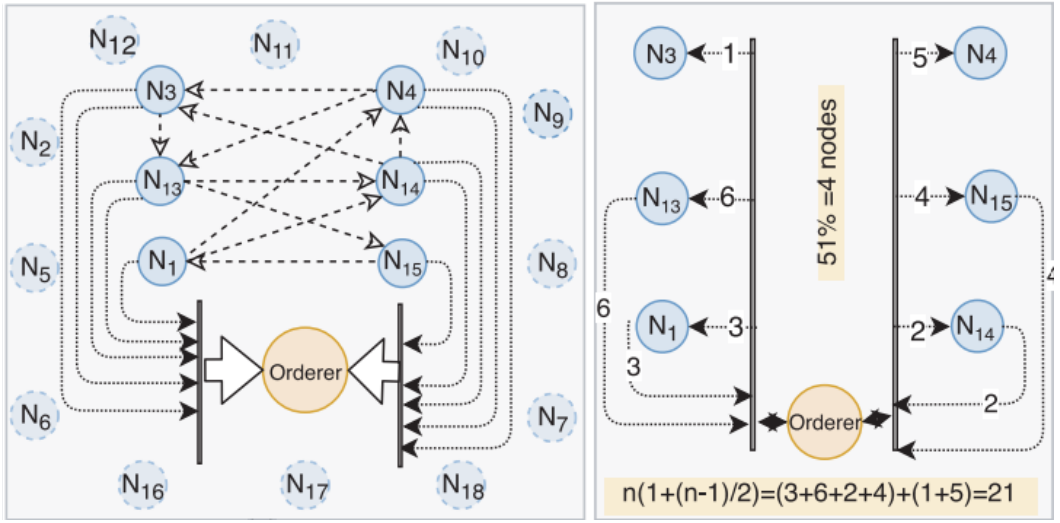
Transaction verification by peer nodes:

- To verify, if this request belongs to the right sender, the verifier uses the sender's public key, which is known by everybody.
- The verifier has a mathematical function that allows him to verify, with the signature and the public key, if the request belongs to the right person.
- It's impossible to fake a signature because it is composed of 256 bit.

Once the transaction is verified it is transferred to the orderer for block creation.

Orderer starts a consensus phase

- The orderer performs the consensus on a candidate block which contains several verified transactions it collects in a given session time.
- Peer nodes which have transactions to be reported to the orderer, they form the session network and will participate in the consensus formation.
- Each session node is assigned a unique random number in range [1, |Ns|].
- Finally, each Tr'_j and corresponding R_j is sent for verification to a randomly selected but unique N^i_s such that N^i_s != N^j_s. In other words, each node in N_s receives one and only one Tr'_j, and it is not the original forwarder for that transaction to the orderer.



(a) **(b)**

(a) Session node selection. **(b) Consensus process.**

- Finally, candidate block and a random number from above range is sent to each of the session nodes.
- Consensus phase ensure that either 51% (ceil([$|N_s|/2$]+1)) of the nodes respond in positive verification or a timeout happens which ensures that block formation does not continue indefinitely.
- The verification response messages must contain the random number to ensure that the selected node is responding.

- The sum of all random numbers for this session is computed as

  $total = |N_s| (1 + ((|N_s| - 1)/2))$

- While processing the verified responses, algorithm computes the sum of random numbers received.

  $sum = sum + faulty\ nodes(R_x) + non\ responsive(R_y)$

- The algorithm returns true for block creation if sum == total and number of nodes responded correctly is at least (ceil([$|N_s|$/2]+1).
- The orderer approves the new block only if the algorithm returns true, and then distributes it to all connected nodes in the network along with the signature of the orderer.
- Every peer node verifies the signature of orderer and adds the block to its ledger.

Security analysis

What if Orderer node is compromised?

- A transaction is structured as <timestamp, pk, sign>. adversary may only change timestamp and pk to conduct brute force. The transactions must be generated within a particular time range of the block generation time which limits the possible range of values for timestamp.

- Creating new pk and thus a new sign incurs significant computational overhead on adversary. Which makes it nearly impossible to predict who is the orderer for a session.

Validation of Block (at Block Formation):

- In order to ensure that no illegal trade is committed all proposed transactions are grouped and validated by session nodes (N_s).

- Here, N_s is different for each block depending on the transaction involved. Thus, for a rogue node to be included in N_s, it must also have a validated transaction sent to the orderer.

- However, the candidate block must have validation from at least 51% N_s, where it is impossible to predetermine the nodes involved in any given session.

Illegal Formation of Block by N_s Nodes:

- This is difficult, as the candidate block and random numbers are only provided to N_s. Hence, they do not know session creation transactions or participants.

- Even if this information is somehow obtained, $N_{ns} \geq (N_s/2) + 1$ must hold. Here, N_{ns} must be 51% of N_s, and all must validate the candidate block with a correct random number.

- The counter check to this is the summation and crosschecking of a random number provided to the respective nodes in N_s by the orderer. Hence, N_s cannot form a block.

Computation and time required

- Time required is based on the orderer selection and the validation by N_s during consensus formation.

$$T_b = T_o + \sum_{i=1}^{\left(\frac{|N_s|}{2}\right)+1} N^i_s(t)$$

where, T_o represents time taken to select orderer

$N^i_s(t)$ is the time for verification by N_s during consensus.

- As we have considered subset of nodes, the propose algorithm will consume less time in block validation than the existing state-of-the art solutions.

Advantages

- Hashing algorithm used makes prediction of orderer nearly impossible.
- The verification/consensus is not skipped.
- Communication and computation overhead are reduced.

Disadvantages

- If for any reason (e.g., bandwidth, cyber attack, error, etc.) the orderer failed to form a consensus within due time, a timeout error occurs. All transactions are rejected and sent back to the sources.
- The application should be designed to detect such an anomaly and resubmit the transactions without the user's knowledge.
- Memory requirement is not improved.

Thank You