Pseudocode :

A) **Verify signature :**

Function *validate_signatures*(signatures_list):
Input : signatures_list
Output: boolean value to indicate if signatures are validated by any of the public keys of replicas
for each signature in signatures_list :
is_signature_validated ← False
for each key in replica_pub_keys :
is_signature_validated ← is_valid or *verify_signature*(signature, key)
// *verify_signature function* verifies signature with key.
if is_signature_validated not validated :
return False
return True

B) Syncing up replicas that got behind

Class syncMsg :
- last_round_tc

Procedure start event processing(M) :
// existing conditions
If M is a sync_ message then process_sync_message(M)

Function process_sync_message(M) :
If M.last_round_tc is none :
send_msg_replica(replica_id, 'sync_message', object syncMsg(last_round_tc)
)
Else :
results_last_round_tc.append(last_round_tc) //results_last_round_tc contains all values for last_round
If len(results) == num_replicas - 1 :
last_round = getMajority(results)
If last_round - self.last_round_tc > 1 :
send_request_transactions()

Function check_replica_slow :
broadcast_msg('sync_ message', object syncMsg(None))

C) Client requests: de-duplication; include appropriate requests in proposals

Function Mempool_push_transaction(transaction): pushes transactions in mempool

Input: transaction - transaction with ':' separated information

transaction_id ← get transaction_id from the transaction

if req_cache has transaction_id: // req_cache is a cache storing information of the states of each transaction.

Log : "Received duplicate transaction for transaction ID "

If req_cache does not have the transaction or is in the processing stage:

Add transaction ID to pending_transactions

Append transaction_queue with transaction ID

Update req_cache with the stage "queue"

If the transaction is in the "queue" or "processed" stage :

Send Response with information about the stage of the transaction

D) **client pseudocode: verify that a submitted command was committed to the ledger**

Function : is_committed(request):

send_request(request) //send request to all replicas

Wait till the 'results' dictionary has f+1 replies for the transaction_ID :

If 'results' dictionary has f+1 values :

//results is a dictionary of transaction ID and set of corresponding responses

return True

Else if timeout has occurred :

return False