**PSEUDO CODE**

1. **Validity checks for cryptographic values**

Using nacl.signing for digital signature and verification

private\_key 🡨 generate\_private\_key()

  public\_key 🡨 generate\_public\_key()

share\_public\_to\_validators()

Function broadcast\_msg()

signature 🡨 sign\_message(private\_key)

send(msg, signature, to=destination)

Function receive\_msg()

public\_key.verify(signature)

1. **“sync up” replicas that got behind**

In Block\_tree:

Function process\_vote:

If node is leader & vote\_count == 2\*f+1:

// By broadcasting leader’s state list to every validator once we know the leader is a loyal one by getting 2f+1 votes for its proposal, validators can sync up

broadcast\_leader\_state\_list()

Function receive\_state\_list(states):

If not compare(states, local.states):

Update state

Update high\_commit\_qc

1. **Client requests: de-duplication**

In Mempool, maintain state of transactions and compare when inserting a new transaction into the mempool

Function insert(transaction t):

If not find(t , transaction\_map):

transaction\_map.add(t, initial\_state)

1. **Verification of submitted command committed to the ledger**

When leader commits a transaction into ledger, it broadcasts committed message to client of the transaction .

Function process\_qc():

If node is leader:

Client\_id = get\_client\_id(qc.vote\_info.tid)

broadcast(“Commit”, {tid, safety.signature}, to=client\_id)

In Client:

Function receive\_commit():

verify\_commit\_message()

update\_commit\_status(tid)