1. Client requests: de-duplication; include appropriate requests in proposals

MemPool

```
request_pool; // submitted (but not yet proposed) transactions/requests
pending_requests; // set of requests that were submitted or proposed
cached_results; // cache of recently executed transactions/requests and their results

Function get_transactions()
    return request_pool.pop() // one transaction/request per block

Procedure add_request(req) // called in Main's request handler
    if <req, result> ∈ cached_results
        Reply = <req.id, result>
        send Reply to req.client // take care of client retransmission after execution
        return

    if req ∈ request_pool ∨ req ∈ pending_requests
        return // duplicate request so don't add to request pool

    request_pool.push(req) // add request
    pending_requests.add(req)
```

2. Client verification of committed command

Client

```
replies; // collected replies per request, indexed by hash of request id and result
pending_requests; // requests that client submitted and is awaiting response for
...
Procedure start_event_processing(M)
    if M is a reply msg then process_reply_message(M)
    ...

Procedure process_reply_message(reply)
    req_idx ← hash(reply.request_id || reply.result)
    replies[req_idx] ← replies[req_idx] ∪ reply.sender
    If |replies[req_idx]| == f + 1 then
        pending_requests.remove(reply.request_id) // remove request_id from pending_requests
```

(CONTINUED ON NEXT PAGE)

BlockTree

```
...
Procedure process_qc(qc)
    if qc.ledger_commit_info.commit_state_id ≠ ⊥
        Ledger.commit(qc.vote_info.parent_id)
        committed_block ← pending_block_tree[qc.vote_info.parent_id]
        request ← committed_block.payload
        result ← "Committed {request.id}"
        Reply ← <request.id, result>
        send Reply to request.client
        MemPool.cached_results.add(Reply)
        ...
```

3. Validity checks for cryptographic values

Main

```
public_keys ← Safety.get_public_keys() // map of validators to their public keys
...
Function verify_signature(sender, signature)
    if sender ∉ public_keys
        return False // unrecognized key
    sender_public_key ← public_keys[sender]
    return verify(signature, sender_public_key) // returns True if signature is verified
...
Procedure process_proposal_msg(P)
    if verify_signature(P.sender, P.signature) == False
        return // drop message
    if ( ¬Safety.valid_signatures(P.block.qc, P.last_round_tc)
                                                ∨ ¬Safety.valid_signatures(P.high_commit_qc, ⊥)
                                                )
        return // drop message
    ...
Procedure process_vote_msg(M)
    if verify_signature(M.sender, M.signature) == False
        return // drop message
    if Safety.valid_signatures(M.high_commit_qc, ⊥) == False
        return // drop message
    ...
Procedure process_timeout_msg(M)
    if verify_signature(M.tmo_info.sender, M.tmo_info.signature) == False
        return // drop message
    if Safety.valid_signatures(M.high_commit_qc, M.last_round_tc) == False
        return // drop message
```

...

4. Catching up with latest commits:

PSEUDOCODE for updating other validators

```
PROCEDURE update_node(block_id):
    missing_commits ← list() #empty initially
    if block_id  ∈ commits: #This means block id is in memory
        idx ← index of block_id in commit_queue + 1
        for i from idx to commit_queue.end
            missing  ∪= <commit_queue[i].id, commit_queue[i].txn>    #append all transactions from idx+1 onwards
    else: #This means block_id has been flushed to the commits file
        no_more_records = False
        commit_found = False
        while(!commit_found)
            record ← read a line in file commited_records
            <id, transaction> ← record
            if block_id ∈ record
                commit_found ← True
                break
        if commit_found
            while (!end_of_file)
            record ← read a line in file commited_records
            <id, transaction> ← record
            missing ∪= <id, transaction>
        for commit in commit_queue:
            missing ∪= <commit.id, commit.transaction>
```