Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2017 Prof. R. Wattenhofer

Distributed Systems Part II

Solution to Exercise Sheet 4

1 PBFT basics

a) According to Lemma 4.18, it is impossible that two prepared-certificates for the same sequence number are gathered within the same view (not even at different nodes). Therefore, once a node has a prepared-certificate, it can be sure that no correct node will execute a different request for the same sequence number.

b) The new primary has to send around the new-view-certificate $\mathcal{V}$; that certificate has to be valid and the set of `pre-prepared`-messages $\mathcal{O}$ has to be constructed validly from $\mathcal{V}$ in the way specified by the protocol. Since $\mathcal{V}$ already determines the content of $\mathcal{O}$ and the `view-change`-messages in $\mathcal{V}$ are signed, correct replicas can rely on $\mathcal{O}$ if the above conditions hold.

c) Not necessarily. It is possible that some node u collected a prepared-certificate for a triple (v, s, r), but as soon as u collected the prepared-certificate, a view change happened. In that case, no correct node can have executed that request yet, but u's `view-change`-message could still end up in the set $\mathcal{V}$ of the `new-view`-message for the next view.

d) The proof of Theorem 4.25 shows that if a request was executed by a correct node, then a prepared-certificate will end up in $\mathcal{V}$. If we take the contrapositive of that statement, we find that if there is no prepared-certificate for a request in $\mathcal{V}$, then no correct node has executed that request yet. Omitting prepared-certificates for requests that no correct node executed cannot harm correctness of the system.

2 PBFT: we need the phases of the agreement protocol

a) Backups start their faulty-timer after they receive a request. If backups do not forward requests to the primary, then a faulty client could just send requests to the backups, and the backups' faulty timers would permanently keep expiring, inducing view change after view change.

A byzantine client could make sure to send a request to a backup even without knowing which node is the primary by simply sending distinct requests to all nodes; all but one node will be backups, and all of their faulty-timers would start running for requests that the primary has never seen and for which the primary can therefore not start the agreement protocol.

b) Lemma 4.18 implies that two correct nodes cannot agree to execute different requests within a single view, and the proof does not rely on nodes waiting for `commit`-messages, so this Lemma remains intact even with the alteration made in this exercise.

However, the `commit`-messages are important for the view-change protocol to maintain safety across views, which we can see in the proof of Theorem 4.25. Consider the following sequence of events:

1. Node u collects a prepared-certificate matching (v, s, r), and directly executes r. No other node has seen a prepared-certificate yet, and a view-change occurs at this moment.
2. The new primary p' of view $v' > v$ collects $2f + 1$ `view-change`-messages, and u's message is too slow to be included. p' thus does not add a $\texttt{pre-prepared}(v', s, r, p')_{p'}$-message to $\mathcal{O}$.
3. In the new view v', correct nodes (with the "help" of byzantine nodes) run the agreement protocol for (v', s, r') for some $r' \neq r$. As soon as correct node $w \neq u$ collects a prepared-certificate matching (v', s, r'), node w will execute r' with sequence number s.

With this, u will execute r with sequence number s, and w will execute $r' \neq r$ with sequence number s.

If $s < s^{\mathcal{V}}_{max}$ (cf. Algorithm 4.23), then r' will be `null`. However, if $s > s^{\mathcal{V}}_{max}$, then r' can be a distinct non-`null` request.

3 Authenticated Agreement

a) We can do roughly the same as we did in Algorithm 4.2, but for multiple values in parallel. Every backup will be collecting messages for every value they hear about. If a correct node gathered agreement for multiple values (or for no values) after $f + 1$ rounds, then it knows that the primary must be faulty. The new algorithm looks like this:

Algorithm 1 Byzantine Agreement with Authentication

Code for primary p:

```
x ← input value of p
broadcast value(x)_p
decide x and terminate
```

Code for backup b:

```
A ← ∅
for all rounds i ∈ {1, ..., f + 1} do
  for all messages value(x)_u that b received this round do
    V_x ← {all messages value(x)_v that b received since round 1}
    if |V_x| ≥ i and value(x)_p ∈ V_x then
      A ← A ∪ {x}
      broadcast V_x ∪ value(x)_b
    end if
  end for
end for
if |A| = 1 then
  decide on the single element in A and terminate
else
  decide "sender faulty" and terminate
end if
```

b) The proof is very similar to the one in the script, so we will only give a rough sketch of how to adapt it here:

- If the primary is correct, then he only sends one message `value`$(x)_p$ in the first round, and all correct backups decide on x after round $f+1$.
- If the primary is byzantine, then there are these cases:
  1. No correct node ever adds a value to A, then all correct nodes output "sender faulty".
  2. (The proof of this case is analogous to correct nodes deciding on 1 in the proof in the script. Check the proof in the script if some detail here is unclear.)
     At least one correct node adds at least one value x to A. For any value x that gets added to A by some correct node, the first time a correct node adds x to A necessarily happens in a round $i < f+1$, and all correct nodes will have $x \in A$ in round $i+1 \leq f+1$. Since this holds for all x, all correct nodes have the same A after round $f+1$.
     If A contains exactly one value after round $f+1$, then all correct nodes decide on that value, otherwise all of them decide on "sender faulty".