Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2015 Prof. R. Wattenhofer

Distributed Systems Part II

Solution to Exercise Sheet 4

1 Zyzzyva

a) If we remove a correct node, the node which replaces it does not know anything and can not assist in the view change process whatsoever. Therefore, commands may be lost. Replacing a byzantine node by any other node is safe since the byzantine node can emulate any behavior which the new node may have.

b) The inconsistent replica histories are reported to any correct client executing a command after the inconsistency. Hence, a correct client can form a proof of misbehavior and trigger a view change. The command may not complete in the current view and can be requested again in the next one.

c) 5 rounds are necessary: 1. client request, 2. primary order request, 3. replica to client answer, 4. client distributes commit certificate to replicas, 5. replicas acknowledge local commit.

2 Zyzzyva . . . again

a) let there be f faulty replicas, one of them the primary. the primary causes f correct replicas to commit to a view change and stop acting in the current view. In this situation, a correct client may only receive $f+1$ responses from the remaining correct replicas. Not enough for the request to complete in either of the two ways. Because there are fewer than $f+1$ replicas that demand a view change, a view change does not occur. Hence the system is not live anymore.

b) Yes this may happen. Assume a client requests a command which is acknowledged by $2f$ correct replicas. It is not complete and the view change is initiated before the client obtains enough answers from the replicas. However, the command is stored in $2f$ local histories of which all may be included in the `NewView`$(C)_p$ message which means that the command is carried over into the new view.

3 Authenticated Agreement

a) The new algorithm looks like this:

if I am P **then**
  $values \leftarrow \{input\}$
  broadcast "P has input"

else
  $values \leftarrow \{\}$
end if
for $r = 0$ to $f + 1$ **do**
  for all received values x **do**
    if $|values| < 2$ and accepted r messages "P has x" with $x \notin values$ **then**
      $values \leftarrow values \cup \{x\}$
      broadcast "P has x"
    end if
  end for
end for
if $|values| = 1$ **then**
  decide item in $values$
else
  decide "sender faulty"
end if

b) If P is correct: there is only one message in the system, which is accepted in the first round. There are no other messages, hence for all processes $|values| = 1$.

If P is Byzantine:

- Assume that a correct process `p` adds x to its value set in a round $r < f + 1$: Process `p` has accepted r messages including the message from P. Therefore all other correct processes accept the same r messages plus `p`'s message and add x to their value set as well in round $r + 1$.
- Assume a correct process `p` adds x to its value set in round $f + 1$: In this case, `p` accepted $f + 1$ messages. At least one of those is sent by a correct process, which must have added x to its set in an earlier round. We are again in the previous case, i.e., all correct processes added x to its value set.

4 Even Faster Zyzzyva

We assume that the primary is correct and that there are $5f + 1$ replicas in total.

We change Zyzzyva in the following ways:

a)
- A client assumes a command to be complete after $|S| \geq 4f + 1$ instead of $|S| = 3f + 1$ $\texttt{Response}(a,\texttt{OR})_r$ messages.
- After $4f + 1 > |S| \geq 3f + 1$ replica responses, clients form the commit certificate (which clients distribute to at least $3f + 1$ replicas).
- If there are less replica responses, we go into the byzantine primary algorithm (unchanged).
- After $f + 1$ $\texttt{IHatePrimary}_r$ messages we initiate a view change.
- We collect $3f + 1$ `ViewChange` messages instead of $2f + 1$.
- Commands that are consistently reported in $f + 1$ histories are accepted into the new history.

b) Even if f $\texttt{Response}(a,\texttt{OR})_r$ messages are missing, no commit certificate is formed! So the new algorithm can handle more requests in 3 instead of 5 rounds of communication. The proofs remain the same as in the script.

- Lemma 4.14 (different sequence numbers) still holds because if we take two subsets of $3f + 1$ replicas from a set of $5f + 1$, they will overlap in at least one correct node.

- Lemma 4.15 (prefix) still holds (same argument as for Lemma 4.14).
- Lemma 4.20 (commit certificate is in C) still holds because if we take two subsets of $3f+1$ replicas from a set of $5f+1$, they will overlap in at least one correct node.
- Lemma 4.21 ($f+1$ reports of commands) still holds because C contains $3f+1$ messages and $4f+1$ replicas sent a $\texttt{Response}(a,\texttt{OR})_r$ message. Therefore, $2f+1$ replicas contributed to C and sent a $\texttt{Response}(a,\texttt{OR})_r$ message. Hence, at least $f+1$ correct replicas have to report the complete command in C.
- Lemma 4.23 and 4.24 follow directly.