RELIABILITY OF DISTRIBUTED SYSTEMS

Mike Greenbaum and David Ponarovsky

October 2021

Exercises

1. Show that this protocol is live. Every command sent by a non-faulty client will receive a response in at most 6Δ time. The interesting case is when there is a view change.
2. Show that this protocol is safe. Every client history that is created when running this protocol is a client history that can be generated in the ideal model. The interesting case is when the Primary crashes and there is a view change. There are several cases to consider in terms of when the Primary crashes, make sure you cover all of them.

Solutions

1. Firstly, notice that if the primary server operates without crushing then the response time is at most 3Δ:

   (a) At most 1Δ for the message to reach the Primary server
   (b) At most 1Δ for the Primary server to decide to send a response
   (c) At most 1Δ for the response to reach the client

   The second case is whether the client sends the message after receiving the "view change" from the Backup. In this case, the Primary server already crushed so the Backup server is not faulty and the client sends directly to the Backup server. The response time is at most 3Δ:

   (a) At most 1Δ for the message to reach the Backup server
   (b) At most 1Δ for the Backup server to decide to send a response
   (c) At most 1Δ for the response to reach the client

   The third case to consider is whether the client sends the message to the Primary server, the server adds it to $log[counter]$, sends it to the Backup but crushes before sending the response to the client. If this is the case the response time is at most 6Δ:

   (a) At most 1Δ for the message to reach the Primary server.
   (b) At most 1Δ for the Primary server to send $log[counter]$ to the Backup and crush before sending the response to the client.
   (c) At most 1Δ for the Backup to receive its last $log[counter]$
   (d) At most 2Δ for the Backup to notice that the Primary crushed and send its last $log[counter]$ to the users
   (e) At most 1Δ for the response (which is in the last $log[counter]$) to reach the client

   Observation: the Backup server knows the Primary crushes after at most 3Δ:

   (a) At most 1Δ for the Backup to receive its last $log[counter]$
   (b) At most 2Δ for the Backup to notice that the Primary crushed and send its last $log[counter]$ to the users

The last case to consider is whether the client sends a message to the Primary server and the server crushes before $log[counter]$ with the message is sent to the Backup. In this case the response time is at most 6Δ:

(a) After 3Δ for client not to hear a response from the Primary server, and the Backup becomes the leader.

(b) At most 1Δ for the message to reach the Backup server

(c) At most 1Δ for the Backup server to decide to send a response

(d) At most 1Δ for the response to reach the client

Notice that we covered all cases and therefore the client will always get a response time is at most 6Δ. We showed that the response time is at most 6Δ in every case, therefore we conclude that the response time is always at most 6Δ.

2. We will prove that the algorithm has the safety property using linearizability.

For the proof, we will assume that there are 2 messages $m1, m2$ that arrive at linearizability points $p1, p2$ correspondingly. If they sent another message, the linearizability points will be denoted as $q1, q2$ correspondingly. We will assume that $p1 < p2$, in other words, that there exists a adversary in which $m1$ will be processed before $m2$ in a perfect world.

The first case to consider is whether the Primary server didn't crush and sent the responses correctly. If that is the case, then it will process the points in their arrival sequence and $m1$ will be processed before $m2$ because we assumed that $p1 < p2$. Therefore, the processing sequence exists also in the real world and therefore, in this case the algorithm is safe.

Notice that the proof is exactly the same for the case that both $m1, m2$ are sent directly to the Backup server.

The only cases that are left are that both $m1, m2$ were sent to the Primary, it crushed and then at least one of the messages was sent to the Backup.

From our assumption, $m1$ arrives before $m2$. If the server sends the response to $m1$ before crushing, then only $m2$ is sent to the Backup. In this case, the first message that was processed was $m1$ and then $m2$ by the Backup server. Therefore, the processing sequence exists also in the real world and therefore, in this case the algorithm is safe.

Now we will assume that the Primary server added $m1$ to $log[counter]$, sent it to the Backup server and then crushed. In this case, the Backup server first processes the messages from the last $log[counter]$ ($m1 \in log[counter]$) before handling messages that it receives (i.e. $m2$). Therefore, the processing sequence exists also in the real world and therefore, in this case the algorithm is safe.

The only case that we didn't cover yet, is whether the Primary server crushed before sending $log[counter]$ with $m1$ to the Backup server. In this case, after 3Δ, both $m1, m2$ will be sent to the Backup server and it will handle them by the order of their arrival:

(a) If $q1 < q2$, then $m1$ will be processed before $m2$ and as we stated before, the algorithm is safe.

(b) If $q1 \geq q2$, if this is the case, then the linearizability points $p1 = q1 - 3\Delta, p2 = q2 - 3\Delta$ are also valid linearizability points sent to the Primary server. Notice that $p1 \geq p2$, which means that there exists a adversary in which $m2$ will be processed before $m1$ in a perfect world. In this case, $m2$ will be processed before $m1$ because it arrived first and in this case, there exists a perfect world in which this is the case, so the algorithm is safe in this case as well.

We showed that the algorithm is safe in every case, therefore we conclude that the algorithm is always safe.