30/30

Advanced Operating Systems (621)
Homework Assignment – 3
Sandipan Dey

7.10. Consider a personal mailbox for a mobile user, implemented as part of a wide-area distributed database. What kind of client-centric consistency would be most appropriate?

Answer:

Let's represent the mailbox as a data item x.

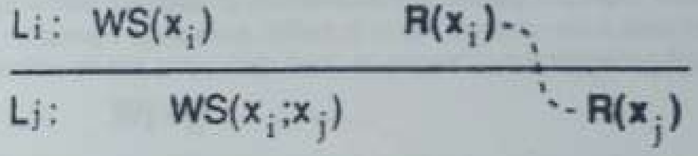
Let the mobile user first access (read / write) the mailbox version $x_i[t_i]$ at local copy L_i (at time t_i) and then access (read / write) another version of his mailbox $x_j[t_j]$ (from another geographic location) at local copy L_j (at a later point of time t_j), with $i < j,\ t_i < t_j,\ i, j \in Z^+$.

The read operation will typically be opening / reading the mails received and stored in local copy. The write operation can be writing a new mail / deleting a mail / moving a mail to different subdirectory / marking a mail as read or unread.

The user will expect each of the following:

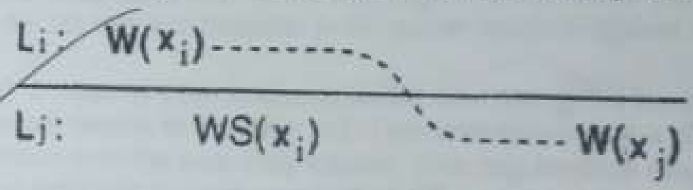
- The mailbox he reads at local copy L_j must be at least the same version or more recent than (must not be an older version) he does at local copy L_i. Assume the user has not done any write operation after reading from L_i, so if reading from L_j at a later point of time he finds that they are older (some of the mails he saw last time are missing! he will be surprised and wonder who deleted those mails), this will be an utter inconsistent behavior.

  This is exactly what **monotonic read consistency** ensures not to happen.



- The user will write / modify his mailbox at local copy L_j only after that copy has brought up to date by any pervious write operation at local copy L_i at an earlier point of time, not on an old version.

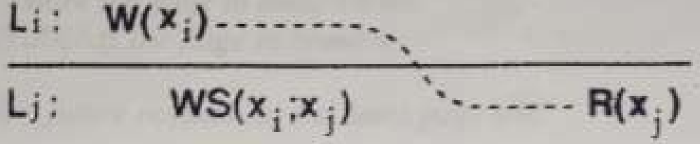
  This is exactly what **monotonic write consistency** ensures.



- Modifications / writes the user does to the mailbox at local copy L_i must be available to him when he reads mailbox at local copy L_j at a later point of time.
  Consider the user reads his mailbox at local copy L_i and starts composing a long important mail.

But due to lack of time and since he needs to move to another geographic location, he can't complete it and saves it as draft. At later point of time he moves to another location and again reads his mailbox at local copy L_j (assume no intermediate write) and plans to finish the unfinished mail. To his astonishment he finds no such draft in his mailbox! (when the earlier writes were not propagated to this local copy).

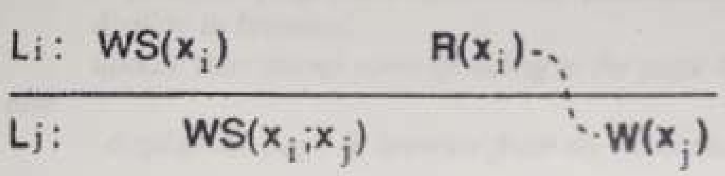
This is exactly what **read-your-writes consistency** ensures not to happen.



- The user will write / modify his mailbox at local copy L_j that is up to date with the value most recently read, not on an old version.

This is exactly what **write-follows-read** ensures.

/10

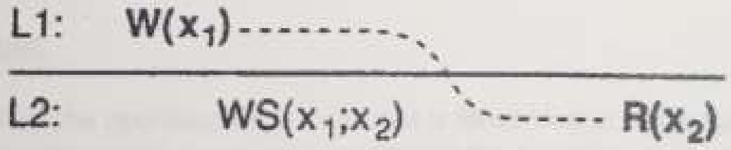


Hence, as in the scenarios explained above, all of the 4 client-centric consistency techniques will be appropriate since the user should always see the same mailbox, irrespective of whether he is reading or updating it. It can be implemented using primary-based local-write protocol. In this case, the primary should always be located on the user's mobile computer.

7.11. Describe a simple implementation of read-your-writes consistency for displaying Web pages that have just been updated.

Answer:

Read-your-writes consistency guarantees that always the most recent data item is being read, i.e., data read is always up-to-date. Effect of write operation on a data item x must be seen by a successive read operation on the same data item, as shown in the following figure:



The browser stores a web page in cache (local copy). But the web page can be updated in the server. If the same out-dated old version of the web page is still shown to the user from the cache at a later point of time even after the updation happens in the server, it will be against read-your write consistency.

In order to implement the read-your-writes consistency the browser must ensure whether it's displaying the most recent version of a page or not. This requires sending a request to the web server every time a page is to be shown from the local copy (cache). This may be done simply by associating a timestamp (last updation time) with the webpage both at the server and the local cache. Maintain the following (hash) table at the local machine:

Page URL	Time Stamp (Last

	Updated)

1. *When the web page is requested for the 1st time*

   a) fetch the page from the web server along with its last updation time.
   b) insert the page URL / updation time pair in the (hash) table.
   c) store the page in local cache.
   d) display the page in browser.

2. *Any future request for the same page will*

```
a) find the last update time stamp ($t_L$) from the table.
b) request the server for the last update information corresponding to the page ($t_U$).
If ($t_L < t_U$)
        re-fetch the page from server to the local cache.
        display in browser.
        update time stamp corresponding to the page in the (hash) table, $t_L \leftarrow t_U$.
else
        display the page in browser from the local cache.
```

7.19 To implement totally-ordered multicasting by means of a sequencer, one approach is to first forward an operation to the sequencer, which then assigns it a unique number and subsequently multicasts the operation. Mention two alternative approaches, and compare the three solutions.

Answer:

If all multicast messages/operations are funnelled through a single member of a process group, that member can assign message identifiers from a sequence. The funnelling process is called the sequencer. The originator of a multicast message sends it to the sequencer which adds a sequence number and then relays it to the other members using a single broadcast message. The sequence numbers are used to ensure that multicast messages are delivered in the same order to all members.
Depending on when / how the sequencer assigns the sequence number to the operation to be multicast and when the actual multicast happens there can be 3 approaches:

Approaches:

1) First the operation to be multicast is forwarded to the sequencer, which then assigns it a unique number and subsequently multicasts the operation: here the sequencer does everything, responsible for both sequence number generation and multicasting.

2) First multicast the operation, but defer delivery (to the application layer) until the sequencer has subsequently multicast a sequence number for it. The latter happens once the sequencer receives the operation: here multicast itself does not wait for the sequencer, but delivery of the multicast operation certainly waits, sequencer is responsible only for sequence number generation.

3) First obtain a sequence number from the sequencer, and then multicast the operation: here the multicast operation must wait till the sequence number generation and the sequencer is only responsible for sequence number generation.

Comparison:

The 1st approach requires sending one point-to-point message containing the operation (send to the sequencer), and a multicast message.

The 2nd approach needs two multicast messages: one containing the operation and one containing a sequence number (to be multicast by the sequencer).
The 3rd approach needs one point-to-point message with the sequence number (to be sent by the sequencer), followed by a multicast message containing the operation / message to be multicast.

Hence the 2nd approach is more expensive than the other two, since it costs 2 multicast messages, as opposed to 1 point to point + 1 multicast message, as required by each of the other two approaches.

✓

10/10