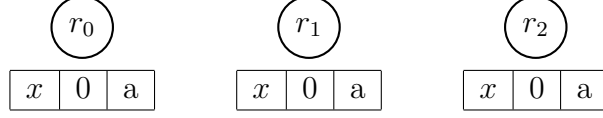
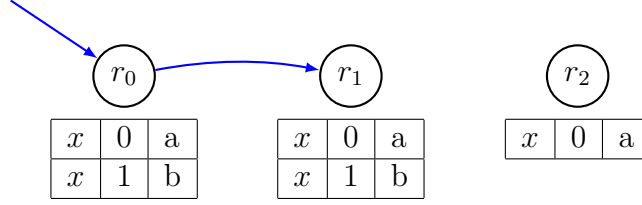


## CRAQ Garbage Collection Bug

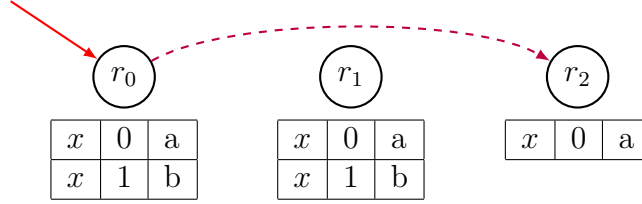
We describe a very minor bug in CRAQ’s garbage collection. Consider the CRAQ deployment with three replicas shown below, where  $r_0$  is the head and  $r_2$  is the tail. Initially, all three replicas store a single version (i.e. version 0) of object  $x$ , which has value  $a$ .



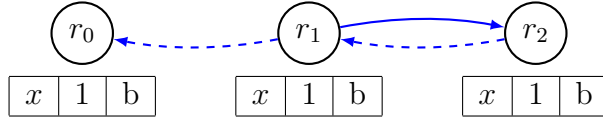
A client sends a write request with value  $b$  to the head,  $r_0$ .  $r_0$  creates a new version of  $x$  and forwards the write request to  $r_1$ , which also creates a new version of  $x$ .



Then, a client sends a read request of value  $x$  to the head.  $r_0$  has more than one version of  $x$ , so it cannot perform the read locally. Instead, it sends a version request to the tail  $r_2$ . When  $r_2$  receives the version request, it queries to find that its version of  $x$  is 0. It sends 0 back to  $r_0$ , but this message is delayed in network and is not yet received by  $r_0$ .



Next,  $r_1$  forwards the write request to the tail, and the tail successfully propagates write acknowledgements backwards through the chain. When  $r_0$  and  $r_1$  receive the acknowledgement, they garbage collect version 0 of  $x$ , as described in the CRAQ paper: **“When an acknowledgment message for an object version arrives at a node, the node marks the object version as clean. The node can then delete all prior versions of the object.”**



Finally,  $r_0$  receives the delayed message, which includes version 0, from  $r_2$ .  $r_0$  then attempts to read version 0 of  $x$ . The CRAQ paper says **“by construction, the node is guaranteed to be storing this version of the object”**, but this is not true.  $r_0$  does not have this version (actually, no node has this version), and is stuck.

