# Transactions Discussion Questions

So far in this course, we have studied systems that are *read-only* and do not perform in-place updates to the underlying state. Systems that seek to perform such updates should implement some transactional protocol. The FaRM paper describes one such system. It is also unique in that it takes advantage of upcoming new technologies: cheap nonvolatile memory by combining battery backup with conventional memory, and fast “direct” access to remote memory (RDMA) without interrupting the remote CPU. These two technologies are exciting and I hope you will incorporate them and other such emerging technologies in your future system designs.

The STO paper describes a way to optimize transactions for shared-memory systems. But, it opens up the exciting possibility of using library interfaces to optimize distributed transactions. We have seen this theme of using higher-level programming abstractions to optimize systems many times in this course.

Now for the questions.

1. The transactional protocol in the FaRM paper validates a read of an object (Step 2, Section 4) even if another machine currently holds a (write) lock on the object but has not (yet) changed the version of the object. Compare this with the transactional protocol in STO (Section 3.3). Is the FaRM protocol correct? If so, why? If not, give a counter example. (You need to understand the standard [two-phase locking protocol](https://en.wikipedia.org/wiki/Two-phase_locking) to answer this question.)
2. An example of STO’s ability to optimize transactions is described in Section 3.5. A transaction X observes an initial counter state of 5, subsequently performs five decrement operations, an unsuccessful test() operation (i.e. observes a counter value of zero), four increment operations, a successful test() operation.
	1. What is the resulting predicate?
	2. Another transaction Y starts after X but commits well before X starts its commit phase. Y performs 10 decrements followed by 11 increments. Can Y commit now?
3. (Optional) Read about the [Paxos](https://en.wikipedia.org/wiki/Paxos_%28computer_science%29) family of protocols.