Problem 1  The snapshot protocol discussed in class assumes that communication channels are FIFO. Derive a snapshot protocol for an asynchronous system that does not depend on the FIFO assumption, and prove it correct (i.e. prove that the protocol produces a consistent global state). Your protocol should not introduce unnecessary blocking. You may assume that at most one snapshot is being computed at any point during a run. Note: A solution that is tantamount to re-implementing FIFO channels is not an acceptable solution.

Problem 2  The snapshot protocol for asynchronous systems that we have discussed in class has a single initiator. We further assumed that protocol completes before a new snapshot protocol is invoked.

Suppose now that a snapshot is initiated concurrently in an asynchronous system by a set of processes $I$. Write an algorithm that returns a consistent global state to each of the processes in $I$, and prove it correct. You should try to minimize the number of local checkpoints taken by your solution. The global state collected by your solution does not need to contain the state of the channels (the state of the processes suffices), and you may assume that channels are FIFO and that the network is completely connected.

Note: Two snapshots $s_1$ and $s_2$ are initiated concurrently if $s_1 \not\rightarrow s_2$ and $s_2 \not\rightarrow s_1$.

Problem 3  Consider a set of processes that are communicating in an asynchronous system by broadcasting messages to each other—think of the broadcast as a sequence of unicasts. Assume that the broadcast is reliable—we will spend quite a bit of time on what that means, but for now, to make things easy assume that there are no failures, and that, therefore, if a process broadcasts a message $m$, then all processes eventually deliver $m$.

It is often valuable to add additional ordering properties to such a broadcast primitive. For instance, consider the following definition of FIFO Order:

If a process broadcasts a message $m$ before broadcasting a message $m'$, then no (correct) process delivers $m'$ if it has not already delivered $m$.

Another useful ordering property is Causal Order:

If the broadcast of a message $m$ causally precedes the broadcast of a message $m'$, then every (correct) process delivers $m$ before delivering $m'$.

As we discussed in class, causal order implies FIFO order. Indeed, Causal Order is FIFO order, plus some ”mystery property” X. So, here is the problem:

• Specify property X.
• Prove that Causal Order is equivalent to FIFO Order plus X.

Problem 4  Prove that $\Sigma_{\text{min}}(\sigma_k^i) = (\sigma_1^{c_1}, \sigma_2^{c_2}, \ldots, \sigma_n^{c_n})$ such that $\forall j : c_j = VC(\sigma_k^i)[j]$. In other words, prove that $\Sigma_{\text{min}}(\sigma_k^i)$ and $\sigma_k^i$ have the same vector clock (see class notes and reading).