Answers to Quiz II

Question 1
(a) $T \geq TT.now() + \text{trans}$

(b) There are several correct answers to this question. The simplest solution is to always keep record of all messages sent for the past $\text{trans}$ time units. Since the maximum transmission time is $\text{trans}$, this keeps record of all messages that could be currently in channels. Some of those messages might have arrived already because they took less than $\text{trans}$ time units. It is fine to include them as if they are currently in the channels, this won't affect consistency. If you want to exclude them, just keep record of all messages received during the past $\text{trans}$ time units, and exclude those form the messages that we record as currently in channels.

(c) There are several correct answers to this question. It is enough to refer to the fact that all processes take the snapshot at the same point in time.

Question 2
(a) If $TT_i[j, k] \geq \text{time}(e_k)$

(b) If, for all $j$, $TT_i[j, \text{node}(e)] \geq \text{time}(e)$

Question 3
- The correct order of messages is: c, f, e, d, g, b, a.

- The chunk server that holds the primary copy of the chunk determines the order of appends, and informs other secondary chunk servers about this order (the message g in the diagram), so that secondary chunk servers use the same order of appends.

Question 4
(a) Incorrect. Consider two messages, A and B, multicast concurrency by two different processes, with no causal dependency among each other. In a causal multicast protocol, other processes may deliver these two messages in any order since there is no causal dependency. One process may deliver A before B, while another may deliver B before A, thus violating the total order property.

(b) Incorrect. Consider two messages, A and B, multicast by the same process. The process multicasts A then B, therefore there is a causal dependency between the two messages. However, in a total order multicast protocol, the only restriction is that all processes deliver messages in the same order. Thus it is possible that all processes deliver B before A, and it is valid as a total order multicast, but it violates causal dependency.

(c) Correct. Note that the two events of multicasting A and multicasting B, by the same process, are causally dependent, because they are two events happening at the same process. Therefore any causal multicast protocol maintains the order between these two events, and consequently ensures the FIFO property.