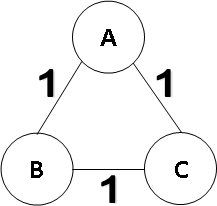
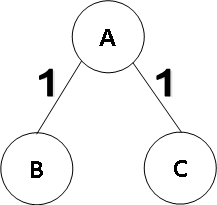
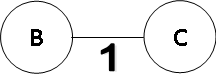
1. This will be disproven by counterexample. Imagine a graph like the one below. The weight between all the vertices is 1. Suppose we want to create a Steiner tree containing vertices B and C.



The claim is that the smallest sub-tree of an MST of this graph that contains vertices B and C is a Steiner tree for those vertices. To disprove this, create an MST rooted at vertex A like below. As the weight between all three vertices is the same, it is trivial to confirm this is an MST of the graph.



The smallest sub-tree of the above tree containing both vertex B and C is the tree itself, which has a total weight of 2. With access to all the edges in the original graph, however, we can create a Steiner tree for those vertices with a total weight of 1. This can be seen below.



The arbitrary sub-tree found before had a weight greater than the above Steiner tree. This means the MST sub-tree was not a Steiner tree and the claim is not true for an arbitrary MST of an arbitrary graph.

1. To achieve totally-ordered multicasting with Lamport timestamps, it is necessary that every message is acknowledged. By the definition of “Totally Ordered Multicast”, all messages need to be received and processed in the same order by all receivers. Requiring acknowledgements from all the processes means that a process will not get to process a message until it knows for certain that all the processes have the same message ready as well. This can be important for cases where messages to certain processes are either lost or delayed.

Consider a message X sent to multiple processes, which is delayed for one of the processes P. If the other processes do not wait for the acknowledgement from P, they will process the message. Suppose there is another message Y which is sent after X, but reaches P before X. In this case without acknowledgements, P will process Y before X since it has not received the latter yet, while the other processes are process X before Y. This difference violates totally-ordered multicasting.