Agreement protocols

Question 1: prove the impossibility proof for the byzantine agreement problem with 3 generals/ prcessors where one general is a traitor. Show the execution of lamport-shostak-peace algorithm using 4 processors with 2 faulty processors, and infer that byzantine agreement cant be reached.

Question 2: Show that interactive consistency, consensus and byzantine agreement problem are reducible to each other. And hence prove that a solution to any of the above problem implies a solution to the other two. An interesting observation here is that all three problem are hence impossible to solve(owing to the impossibility result and the reducibility).

Question3: Suppose that the Byzantine generals only need to reach an approximate agreement, That is, the following two conditions must hold:
a) All loyal generals attack within 10 minutes of each other.
b) If the commander is loyal, all loyal generals attack within 10 minutes of the time given by the general's order. Show that approximate agreement cannot be guaranteed unless more that two thirds of the generals are loyal.

Leader Election

Question 1: For a synchronous system, design an election algorithm using a mutual exclusion algorithm, without knowing how the mutual exclusion algorithm is implemented.

Questoioin 2: prove that the Hirschberg Hirschberg-Sinclair algorithm has a Message Complexity of O(n lg lgn). If we modify the algorithm so that token is just sent in one direction, show that we end up with an algorithm which doesnt have an NlogN communication complexity. Modify this algorithm appropriately to get the Onlogn complexity.

Question 3: Give a randomized algorithm to elect a leader, ie each node has equal probability of getting elected as the leader. Also analyze its messsage and time complexity.

Question4: How could the echo algorithm be used to get an election algorithm for any undirected graph?

Question5: Consider the problem of determining the size of a synchronous bidirectional ring with UIDs using a comparison-based algorithm. Each process should output the correct number n of processes. (a) Design an algorithm that solves this problem. Prove the worst-case number of messages required to solve the problem with this algorithm. Try to get the smallest value you can for this measure.

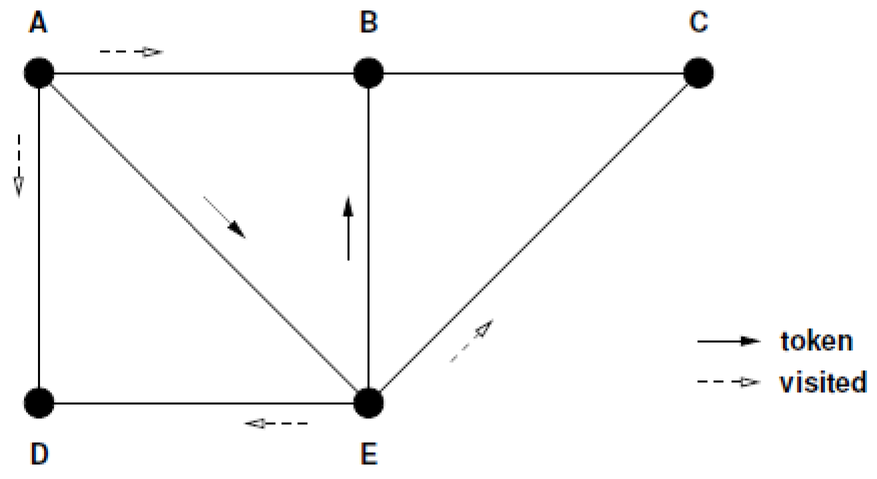
Question 6: Is leader election possible in a synchronous ring in which all but one processor have the same identifier? Either give an algorithm or prove an impossibility result.

Question 7: Consider the following algorithm for leader election in an asynchronous ring: Each processor sends its identifier to its right neighbor; every processor forwards a message (to its right neighbor) only if it includes an identifier larger than its own.
Prove that the average number of messages sent by this algorithm is *O(n log n)*, assuming that identifiers are uniformly distributed integers.

Wave/traversal algorithm

Question1: Suppose you want to use the wave algorithm in a network where duplication of the messages may occur. How would you modify the Echo algorithm ensuring that it works correctly.

Question2:



Given the arbitrary network topology shown above, describe the different ways how the Cidon's algorithm may proceed if the token arrives at node B after the visited message sent by A? and also when the token arrives at node B before the visited message send by A?

Question 3: What are the advantages of the Awerbuch's improvement over the classical depth-first-search algorithm? How did Cidon further improved it.

Question4: Let each process initially carry a random integer value.. Adapt the echo algorithm to compute the sum of these integer values.

Question 5: Design an algorithm to find the depth first search tree starting at a root assuming that you know the neighbour list and using the hint that a node when it receives a token, notifies all the neighbors but pass the token to only one of the neighbor. The algorithm should have a complexity of O(n) rather than O(m).

Minimal Spanning tree protocol

Question1: in the GHS algorithm, if L > level v , why does v postpone processing the incoming test message from u? which problem could occur if at this time v was reporting uv as lowest-weight outgoing edge to its core nodes? Why can we be sure that this is never the case? Why does this postponement not lead to a deadlock?

if level v = levelu, why does v postpone processing the incoming connect message from u? Why does this postponement not give rise to a deadlock?

Question2: How can the Gallager-Humblet-Spira algorithm be turned into an election algorithm for any undirected graph?

Question3: Consider a ring of size 3, in which all edges have weight 1. Show that in this case, the Gallager-Humblet-Spira algorithm could get into a deadlock. How can we avoid situations like this so as to ensure no deadlock?

Some General Question

Question : State whether the following statements are True or False and provide a 1 sentence justification for your answers in each case.

a) (3 Points) Causal ordering implies total ordering and FIFO ordering in an ordered multicast protocol (you don't need to come up with an example, just one sentence justification to argue why the statement is true or false).

False. Causal does not imply total ordering.

b) (3 Points) The Chandy-Lamport global snapshot algorithm works correctly for non-FIFO channels.

False. Non-FIFO channels could lead to application messages overtaking the marker, thus leading to the recording of an inconsistent state of the channels.

c) (3 Points) In a system with N processes, the Chandy-Lamport snapshot algorithm will always show that at least (N-1) channels are empty.

True. The channels through which each process receives the marker for the first time are recorded to be empty. All such channels define a spanning tree, thus, there are N-1 empty channels.

d) (3 Points) It is possible to solve Byzantine Generals problem in an asynchronous system where all the generals (processes) are loyal (correct) but the messages may be dropped.

False. Failed processes are indistinguishable from correct processes with arbitrary message delays. Since asynchronous BG is impossible, so is the above.

e) (3 Points) Consider two clocks that drift 1 second in every 106 seconds with respect to each other. A resynchronization interval of 2 x 104 milliseconds is sufficient to limit their skew to 20 milliseconds.

True. Since the clocks resynchronize every 208 ms to skew 0, we have to worry only about an interval of 128 ms at the end of the 106 seconds where a skew could be accumulated 106000/ 208 = 509 with reminder 128). If the clocks have 1000 ms (1 second) skew after 106 seconds (the worst case as specified), they will have 1.96ms skew after 208 ms. So even in the worst case the skew after 128 ms will be always below 20 ms.