Massachusetts Institute of Technology — Handout 11
6.852: Distributed Algorithms
Prof. Nancy Lynch — October 22, 2015

Problem Set 4, Part a

Due: Thursday, November 5, 2015

Readings:

Sections 15.3-15.5. Chapter 16.
Next week:
Chapter 18. Lamport's "Time, Clocks,..." paper. Mattern paper. Chapter 19.

Problems:

1. Design a version of the *AsynchBFS* algorithm in Section 15.4 that allows node i_0 to discover when the tree construction has been completed. You should follow a strategy like the one described in the paragraph on "Termination" on p. 503-504.
   (a) Explain your algorithm in words.
   (b) Give pseudocode.
   (c) Analyze its message and time complexity.

2. Exercise 15.33.
   Consider the *GHS* minimum spanning tree algorithm.
   (a) State and prove carefully an upper bound on the time from when the first process awakens until the last process announces its results. You may assume that a preliminary protocol is used to awaken all the nodes as quickly as possible.
   (b) How tight is the upper bound proven in (a)? That is describe a particular execution of the algorithm that takes time that is as close as you can get to your upper bound.

3. Design an algorithm for an undirected graph network in which every process determines the *diameter* of the network, which is the maximum distance between any pair of nodes in the graph. Assume that the processes start out not knowing anything about the graph.
   Try to minimize the time until every node outputs the diameter, perhaps using a synchronizer strategy.
   (a) Describe your algorithm in words.
   (b) Provide pseudocode for the part of the algorithm that uses the synchronizer. You don't need to write code for the synchronizer itself or its implementation.
   (c) Give an upper bound for the time until every node outputs the diameter.

4. For k even, consider the *k-pseudo-session problem.* This is a modification of the k-session problem that allows a limited amount of leeway in the way events at different processes are ordered. Namely, define a *pseudo-session* to be any sequence of *flash* events containing at least 2 $flash_i$ events for every i. The *k-pseudo-session problem* requires that the algorithm should perform at least $\frac{k}{2}$ disjoint pseudo-sessions, in any fair execution.
   (a) What is the best upper bound you can give for the worst-case time of the last *flash* event in any execution? Describe an algorithm that demonstrates this bound.

(b) What is the best lower bound you can give for the worst-case time of the last *flash* event in any execution?

(Note: Formally, the measure we consider is $T(A)$, as defined on p. 557 of the textbook.)

5. Work on your project proposal.