Massachusetts Institute of Technology — Handout 5
6.852: Distributed Algorithms
Prof. Nancy Lynch — September 15, 2005

Problem Set 1, Part b (modified)

Due: Thursday, September 22, 2005

Reading:

Sections 3.5-3.6 and 4.1-4.4 of *Distributed Algorithms*
(Optional) Attiya-Welch, Chapter 2

Reading for next week: Section 5.1 and Chapter 6 of *Distributed Algorithms*
Aguilera, Toueg paper, listed in Handout 3
(Optional) Keidar, Rajsbaum paper, Sections 4.1-4.4 (skim)
(Optional) Attiya-Welch, Sections 5.1 and 5.2

Problems:

0. Email Tina to join the course mailing list and to select a problem set to grade, if you have not already done so (details on course website).
1. Exercise 4.4.
   The main point of this exercise is to get some practice defining and using a simple simulation relation.
2. Consider a variation of the Shortest Paths problem described in Section 4.3 of the textbook where:
   A. Halting is not required: we require only that *eventually* a shortest paths tree over the processes exists and does not change,
   B. Each process's state contains a Boolean constant, *source*, which is set to *true* for i_0 and $false$ for every process $i \neq i_0$, and
   C. (modified:) Except for *source* and edge *weight* constants, the initial state of each process is arbitrary: each non-*source*, non-*weight* state variable v of a process, (including any *rounds* or *status* variables), is initially set to an arbitrary value from $\mathbf{type}(v)$.

   Note that these conditions mean that different processes might think they are in different rounds, and processes can't tell if they are just starting an algorithm execution or are in the middle of an execution.

   (a) Explain informally why the $BellmanFord$ algorithm descibed on p. 62 of the textbook does not solve this new version of the Shortest Paths problem.
   (b) Describe informally a modified version of $BellmanFord$ that does solve the Shortest Paths problem under these conditions.
   (c) Give formal code for your new algorithm.
   (d) (Don't turn in:) Think about how you would prove that your modified algorithm is correct.
3. Exercise 4.17
4. Exercise 4.22.
5. Design an algorithm to produce a depth-first spanning tree starting at the root. It should be similar to the one discussed in class, but should have time complexity $O(n)$ rather than $O(|E|)$.
   Hint: When a node receives the token for the first time, it notifies all its neighbors, but passes the token to only one of them. Give pseudocode, and give a convincing argument for why your algorithm is correct and has the claimed time bound.