Problem Set 2, Part b

Due: Thursday, October 8, 2015
Problem sets will be collected in class. Please hand in each problem on a separate page.

Readings:

Section 5.1, Chapter 6 of *Distributed Algorithms*
Aguilera, Toueg paper, listed in Handout 3
Keidar, Rajsbaum paper (skim)

For next week: Chapter 7 (skim 7.2); Chapter 8.

Problems:

5. Exercise 5.2.

   Consider the following variant of the (deterministic) coordinated attack problem. Assume that the network is a complete graph of $n > 2$ participants. The termination and validity requirements are the same as those in Section 5.1. However, the agreement requirement is weakened to say: "If any process decides 1, then there are at least two that decide 1." (That is, we want to rule out the case where one general attacks alone, but allow two or more generals to attack together.) Is this problem solvable or unsolvable? Prove.

6. Modify the *FloodSet* algorithm of Section 6.2.1 by adding a local *stopping condition*, in order to obtain the following additional *early stopping* time bound property:
   If the execution has only $f' \leq f$ failures, then all nonfaulty processes decide (but don't halt) by the end of round $f' + 2$.

   Prove that your algorithm works, that is, that it solves the stopping agreement problem for stopping failures, and that it has the additional time bound property.

   Note: Recall that we require the *uniformity* property, which says that all processes that decide (even if they later fail), decide on the same value.

7. Exercise 6.21.

   Consider the *EIGByz* algorithm. Construct explicit executions to show that the algorithm can give wrong results if it is run with

   (a) Seven nodes, two faults, and two rounds.

   (b) Six nodes, two faults, and three rounds.

8. Exercise 6.33.

   This exercise is designed to explore the construction in the proof of Lemma 6.26, which pastes together two triangle systems to yield a hexagon system.

   (a) Carefully describe an algorithm A for a three-process complete graph that solves the *no-fault agreement problem*, that is, the Byzantine agreement problem in the special case where no processes are faulty.

(b) Now construct system S by pasting together two copies of your algorithm A, as in the proof of Lemma 6.26. Describe carefully the execution of S in which processes 1, 2, and 3 start with input 0, and 1', 2', and 3' start with input 1.

(c) Does S solve the no-fault agreement problem (for the hexagon network)? Either prove that it does or give an execution that shows that it does not.

(d) Does there exist a three-process algorithm A such that *arbitrarily many* copies of A can be pasted together into a ring, and the resulting ring will always solve the no-fault agreement problem?

9. Exercise 6.45.

In Section 6.7, it is shown that stopping agreement tolerating f faults cannot be solved in f rounds. The construction involves the construction of a long chain connecting the two runs in which all the processes are non-faulty and have the same inputs. The chain, however, is only constructed implicitly.

(a) How long is the chain of runs?

(b) By how much can you shorten this chain using Byzantine faults rather than stopping faults?