ECE 526 (Fall 2014) Distributed Algorithms
Homework 1 Due: 9:30 a.m., September 16, 2014 (Tuesday)

Please type your answers (e.g., using Latex, or Word).

See the policy of 48-hour extension described in the course handout.

1. Consider n processors $p_0, \cdots p_{n-1}$. Define *k-set consensus* problem as follows. Each processor p_i starts with some arbitrary integer value x_i as its input, and should output a value y_i such that:

   Validity: $y_i \in \{x_0, \cdots, x_{n-1}\}$.

   k-Agreement: the number of different output values at fault-free processors is at most k.

   (a) For a synchronous system, present an algorithm for k-set consensus.

   HINT: Consider the original f crash tolerant consensus algorithm, but instead of running it for $f+1$ rounds, run it for $f/k+1$ rounds.

   (b) Present a proof that your algorithm achieves k-set consensus.

   (c) What is the message complexity of your algorithm.

   INTERESTING FACT: k-set consensus is achievable in *asynchronous* systems only if the number of crash failures to be tolerated (f) is at most $k-1$. Note that our original consensus problem can be viewed as k-set consensus with $k = 1$.

2. Define *clean* crash as the faulty processor that either sends all its messages or none when it crashes in a certain round (in a synchronous system). Consider the consensus problem in a synchronous system with up to f_1 clean crashes and up to f_2 normal crashes (i.e., the crash fault model we have studied in the class). What is the tight bound on the number of rounds? That is, suppose that the bound is x rounds. Then, you have two tasks:

   (a) Show that no consensus algorithm achieves consensus in strictly less than x rounds in all executions, and

   (b) Design an algorithm that solves the consensus problem using at most x rounds. Present a proof that your algorithm is correct.

   You may use any results presented in the class without proving them again.

3. Consider the *Phase King* algorithm. The proof of correctness requires that $n \geq 4f+1$. Does the algorithm still behave correctly if $n = 4f$? Justify your answer.

4. Design a consensus algorithm for $f \geq 2$ crash failures with the following *early stopping* property: If no processors fail in an execution, then the algorithm terminates in less than $f + 1$ rounds; otherwise the algorithm may use more rounds (more than $f + 1$ rounds is acceptable).

   Present an algorithm and prove its correctness.

   HINT: Processors need not decide in the same round.

SUGGESTED EXERCISES
(not for credit – you do not need to submit solutions for these)

- In all of the problems above, we assumed a point-to-point network over which nodes communicate via message passing. Suppose now that the nodes communicate with each other over a reliable broadcast channel. Thus, each transmission of each node is received by all the nodes reliably and *identically*.

  How would the threshold on n be affected when tolerating f Byzantine faults?

- Consider the f-crash tolerant consensus algorithm discussed in the class. The algorithm requires $f + 1$ rounds.

  (a) Modify the original algorithm to achieve consensus in f rounds when $f = n - 1$.

  (b) We know that consensus is impossible in asynchronous systems even with just one crash failure. Consider the crash tolerant algorithm in the book, and identify how it may fail to achieve consensus correctly when $n = 3$ and $f = 1$ (n = number of processors). You may assume that all the nodes begin execution of the algorithm simultaneously. Each processor determines the time duration of each round using its own local clock. The system is asynchronous.