ECE 526 (Spring 2013) Distributed Algorithms
Homework 1 Due: 9:30 a.m., February 5, 2013 (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 a synchronous system. Suppose that we want Byzantine consensus to satisfy the following condition: every non-faulty node's decision must equal the input of some non-faulty processor. As usual, agreement and termination conditions should be satisfied as well.

   (a) Show that to be able to satisfy these conditions while tolerating up to f Byzantine faults, the number of nodes n must be at least $\max(3, m)f + 1$, if each node's input is an integer value between 0 and $m - 1$ (both inclusive), where $m \geq 2$.

   (b) Show that $n \geq \max(3, m)f + 1$ is also a sufficient condition for complete graphs. Hint: Try implementing the algorithm using a Byzantine broadcast algorithm.

2. 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 (also considered in the FLP paper) can be viewed as k-set consensus with $k = 1$.

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

- 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.

(c) For the synchronous case, suggest an algorithm that allows the nodes to decide on an output earlier, if there are fewer than f crash failures. Do all nodes decide in the same round when using your modified algorithm?

- 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*.

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

  (b) Is fault-tolerant consensus possible in an asynchronous network that satisfies the above broadcast property?

- Consider the *Phase King* algorithm. The proof of correctness requires that $n \geq 4f + 1$. Convince yourself that the algorithm may behave incorrectly if $n = 3f + 1$ (identify a behavior of the faulty nodes that will lead to incorrect outcome).