CS 9668/4438 Internet Algorithmics
Solution for Assignment 4

1. *(i) (25 marks) Prove that the given algorithm correctly solves the half consensus problem in the presence of at most f crash failures, for any $f < n$.*

   Since there are at most f failures and the algorithm performs $1 + \frac{f}{2}$ rounds, then there must be a round j where the number of failures is at most 1 (as otherwise, if there are at least 2 failures per round the total number of failures would be $f + 2$).

   If no processor fails in round j then all messages are correctly delivered and all processors will select the same value v by the end of that round. Notice that in further rounds the processors will only transmit the value v; hence, even if additional processors fail, all the non-failing processors will at the end select value v, and thus half consensus would be achieved.

   If one processor p fails during round j, then some of the messages that p was intending to send during the round might be lost. Let P_1 be the set of processors that received a message from p in this round and let P_2 be the set of processors that did not receive a message from p. Note that all processors in P_1 received the same set of messages and thus they will select the same value v_1 by the end of the round. Similarly, all processors in P_2 received the same set of messages, and by the end of the round they all will select the same value v_2. (Note that it could be that $v_1 = v_2$.)

   Therefore, in the following rounds (if any) only two values v_1 and v_2 will be transmitted by the processors. Regardless of how many processors fail in these rounds at the end there can be at most two values chosen by all the processors: v_1 and v_2. Since at least half of the non-failing processors will select v_1 or v_2, then the processors achieve half consensus.

   (*ii*) *(5 marks) Compute the time complexity and communication complexity of this algorithm.*

   The algorithm performs $1 + \frac{f}{2}$ rounds, so the time complexity is $O(f)$. In each round each processor sends messages to all other processors, so in each round at most $n(n-1)$ messages are sent. Since the algorithm performs $1 + \frac{f}{2}$ rounds, the total number of messages sent is $(1 + \frac{f}{2})n(n-1)$, which is $O(fn^2)$.

2. *(5 marks) Consider a synchronous system in which processors can fail by* clean *crashes. What is the minimum number of rounds that the algorithm described in the previous question needs to perform to ensure that at the end all the processors select the same value?*

   Since in a clean crash either all the messages that a processor needs to sent in a round are sent or none of them is sent, then the algorithm only needs to perform one round for the non-failing processors to achieve consensus. This is because during this round all processors receive the exact same set of messages, so at the end of the round all of them will have selected the same value.

3. *(i) (30 marks) In class we studied the use of consistent hashing in Chord for finding keys in a peer-to-peer network. Assume that each processor p_i has a finger table that allows it to store its own address plus the addresses of two other processors. Which addresses must be stored in the table so as to, both, minimize the number of messages needed to find a key (or to decide that a key is not stored in the system), and to ensure that every key can be found?*

   One of the addresses must be that of the successor as this will ensure that each processor can be reached, and hence there will be the possibility of finding every key. Hence the first finger will be the address of the successor. The second finger we will set as the successor of $h_p(ID) + x$, where ID is the address of the current processor and the value of x will be determined so as to minimize the maximum number of messages that need to be sent to find a key.

Note that to find a key it might be necessary to send messages using both fingers. The second finger will be used to move quickly along the ring and to get as close as possible to the position of the key. Then, when the distance between the current processor and the position of the key in the ring of identifiers is smaller than x, the first finger will be used to reach the processor that must store the key being sought. Since the size of the ring is 2^m then the maximum number of messages that need to be sent is

$2^m/x$ plus the number of messages sent to the successors.

If we assume a hash function that maps processors uniformly across the entire ring, then in a segment of the ring of size x there will be $(x/2^m)n$ processors, where n is the total number of processors in the system. Therefore, the maximum number of messages that need to be sent to find a key is

$$f(x) = \frac{2^m}{x} + \frac{x}{2^m}n$$

The derivative of $f(x)$ with respect to x is

$$f'(x) = -\frac{2^m}{x^2} + \frac{n}{2^m}$$

Setting $f'(x) = 0$, we get that $x = 2^m/\sqrt{n}$. Therefore, the second finger must be chosen as the successor of $h_p(ID) + 2^m/\sqrt{n}$.

(ii) (5 marks) Prove that with your choice of fingers any key stored in the system can be found.

Regardless of where the key is located in the system, the use of the second finger will locate a processor p that is at a distance smaller than $2^m/\sqrt{n}$ from the location of the key in the ring of identifiers. Then, by using the first finger we are guaranteed to find the key as all processors at distance at most $2^m/\sqrt{n}$ from the ring identifier for the key will be queried and, if the key is stored in the system, it must be stored in one of these processors.

(iii) (5 marks) Compute the maximum number of messages that need to be sent to either find a key or to determine that the key is not stored in the system.

As explained above, at most

$$\frac{2^m}{x} + \frac{x}{2^m}n = \frac{2^m}{2^m/\sqrt{n}} + \frac{2^m/\sqrt{n}}{2^m}n = 2\sqrt{n}$$

messages are sent. So the communication complexity is $O(\sqrt{n})$.

4. *(i) (25 marks) Write a synchronous algorithm in Java that a processor in a peer-to-peer system can use to find a given set of keys, assuming that the consistent hashing scheme described in class is used to determine where to store the keys. Your algorithm* **must** *use finger tables to find the keys.*

   You can download the algorithm from the course's website.