CS 9868 Internet Algorithmics
Assignment 4: Graduate and Undergraduate Students
Due November 28

Zaid Albirawi
250626065

1. **HalfConsensus:**

Proof of algorithm correctness: For each process the algorithm initializes the selected value to be the value of that process. For each round, each process will send the selected value to all the processes in the network. Furthermore, each process will receive k values form the processes in the network, where k is less than or equal to the number of processes in the network and set the selected value to be the lowest value received. Once a round limit is reached each process outputs the lowest value it had encountered.

Assumption: the processes that fail contain the lowest values in the network. Otherwise, those processes' failures will not affect the network.

Firstly, if there exists a round with zero failures, then each process would have received a value from each of the processes in the network and each process will have selected the smallest value in the network by the end of the round.

Furthermore, if there exists a round with at most one failure, then that will cause a split of the selected values for all the processes in the network. The selected values will either be skewed towards the smallest value, the value of the process that failed, the second smallest value, a process that is functioning and had sent a message to all processes, or a create a 50 50 split in the network. In each of the three cases half consensus is guaranteed.

Lastly, since one round of at most one failure will achieve half consensus then at least two failures per round may prevent half consensus. Therefore, if

$$rounds = \frac{f}{2} + 1$$

then,

$$2 \times rounds = 2 \times \left(\frac{f}{2} + 1\right) = f + 2$$

however, since

$$f + 2 > f$$

then, the algorithm with will always achieve half consensus.

Time Complexity: the time complexity of the algorithm is $O(f)$ as the algorithm has to at least run $\frac{f}{2} + 1$ times.

Communication Complexity: the time complexity of the algorithm is $O(f \times n^2)$ as the algorithm has to at least run $\frac{f}{2} + 1$ times and with zero failures it will have to broadcast $n \times (n-1)$ messages a round. Therefore, $\left(\frac{f}{2} + 1\right) \times \left(n \times (n-1)\right)$.

2. Only one round is needed. In f processors fail then those processes either sent out all their messages or none of them. Therefore, all the processors in the system would have received the same set of values and will end up selecting the same value.

3. **Assumption**: Processors are organized in a ring where the right neighbour of each processor has a higher key.

   The first finger will be the address of the right neighbour processor.

   Assumption: If the ring is split into two equal segments, then those two segments will approximately have the same number of processes mapped to them and each process will approximately have the same number of keys mapped to it.

   To compute the second finger that will be stored by each process the ring is split into two segments s_1 and s_2. Each segment will approximately have $\frac{n}{2}$ processes mapped to them. The first process of s_1 will store the address of the first process of s_2. While the first process of s_2 will store the address of the first process of s_1.

   Furthermore, the second process of each of these segments, s_1 and s_2, will split each of these segments into smaller segments, $s_{1,1}$, $s_{1,2}$, $s_{2,1}$ and $s_{2,2}$ each of size $\frac{\left(\frac{n}{2}-1\right)}{2}$. The first process of each of the new segments $s_{1,1}$ and $s_{2,1}$ will store the address of the first process of $s_{1,2}$ and $s_{2,2}$ respectively. While the first process of each the $s_{1,2}$ and $s_{2,2}$ segments will store the address of the first process of s_1.

   Assumption: The smallest segments will always be of size two for simplicity.

   The segments are split until the size of each segment is equal to two. The first processors in each of those first segments will store the addresses of the first processors of each of the second segments. While each of the first processors in the second segments will store the address of the first process of s_1.

   In the worst-case scenario these addresses will allow the search operation to be conducted in $\log n + 1$ operations. The one is for reaching a process that has the address of s_1 as one of its fingers and the $\log n$ is for traversing down the binary tree created by splitting the ring segments into smaller segments.

   Proof that any key can be found: Since every process has the address of the its right neighbour as one of its fingers then every process in the network is accessible. Therefore, since every key has to be assigned to a process in the network and since every process in the network is accessible then any key can be found in the network if It exists.

   Communication Complexity: In the worst-case scenario $\log n + 1$ messages will be transmitted to find a key or find out if a key exists or not. This is because the fingers for every process allow it to divide the search space by half after every message. Therefore, the communication complexity is $O(\log n)$.