Distributed Computing: Spring 2017
Programming Assignment 3: Implementing EIG & Phase-King Algorithms for solving Byzantine Agreement
Submission Date: 30th March 2017, 11:00 pm

Goal: The goal of this assignment is to implement two algorithms for solving **Binary Byzantine Agreement Problem**: (1) EIG and (2) Phase-King algorithms. Implement both these algorithms in C++. Then, you have to compare the message-complexity of both the algorithms.

Description. As discussed in the class, assume a synchronous network with n processes, f faulty processes with $n > f/4$ (not $f/3$). As per the Byzantine Agreement (BA) problem, one of the processes denoted as the source process will broadcast a binary value v to all the other processes. Among the remaining processes, all the correct processes must decide on the same value. If the source is non-faulty, the correct processes must decide on v.

To achieve this, implement the algorithms EIG and Phase-King. Measure the message-complexity. Theoretically, EIG algorithm has message complexity of $O(n^f)$. On the other hand, Phase-King has a complexity of $O(n^2 f)$.

Also note that technically Phase-King algorithm is for solving the consensus problem. But as discussed in the class from Prof.Neeraj Mittal's notes, a solution to consensus problem can be easily adapted to solve the BA problem.

Practical Considerations - Implementation of Synchronous Systems. In synchronous systems as discussed in the class, the computation proceeds in rounds. In a round, a process send messages, receives messages and performs local computation.

To implement this, you can assume that a round consists of a few milli-seconds. To get a more reliable estimate, measure the average round-trip time in the network that you are running the program. Then to obtain a safe estimate of the round-trip time, multiply this average by 5 or 10.

If in the course of execution, your program observes that a message m by a non-faulty process P_i is delivered to P_j late, i.e., not in its round, then you can stop the execution and re-calibrate the length of a round.

It can be seen that it is easier to implement the synchronous system on a network constructed from a single machine rather than on distributed system such as MS azure.

Also you have to assume that the underlying graph of the system is completely connected. Any process will be able to message another process. Assume that the system provides the connectivity of oral messages, OM.

Practical Considerations - Implementation of Malicious Processes. You can simulate the behaviour of a malicious process using randomness. Each time a malicious process P_m wishes to send/broadcast a message, it decides the value to send by flipping a coin. For simplicity, you can assume that the value it sends in a round has no connection with the value it receives in the previous round.

Input: The input to the program will be a file, named inp-params.txt, consisting of all the parameters described above. The first line of the input file will contain the parameters: n, f.

Assuming that processes are numbered $0, 1, 2....n$, you can denote P_0 as the source process for the BA problem. The next line indicates the set of processes that are faulty. A sample input file is as follows:
20 4
3 6 8 23

Output: Your program must produce an output for each algorithm. The output is the combined log of all the events of the algorithm. For instance the output for EIG algorithm is as follows:

	OM(0)	OM(1)	...	...	OM(m)
P_0	1	0	...	...	0
P_1	1	1	...	...	0
	...	...	...	...	
P_n	1	0	...	...	0

Here each row represents the value that process chooses as majority from the commander. Similarly, the output of Phase-King algorithm is as follows:

Phases	1	2	...	...	f+1
P_0	1	0	...	...	0
P_1	1	1	...	...	0
	...	...	...	...	
P_n	1	0	...	...	0

Each row shows the value of v as decided by a process in in a given phase. Note that each phase consists of two rounds.

Report: The report must explain the low-level design of your algorithms. It must explain the specifics of your implementation.

The report should contain a comparison of message-complexity of both the above mentioned algorithms: (1) EIG and (2) Phase-King algorithms. You must run both these algorithms multiple times to compare the performances and display the result in form of a graph.

The graph in the report will be as follows: the x-axis will vary the number of processes n while the y-axis will show the average message complexity. Vary the number of processes n from 20 to 50 in the increments of 10. Have the total number of faulty processes f to be 4 in all these simulations. Finally, you must also give an analysis of the results while explaining any anomalies observed.

Deliverables: You have to submit the following:

- The source file containing the actual program to execute. Name it as: ProgAssn3-<rollno>.cpp
- A readme.txt that explains how to execute the program
- The report as explained above

Zip all the three files and name it as ProgAssn3-<rollno>.zip. Then upload it on the google classroom page of this course. Submit it by **30th April 2017, 11:00 pm**.

Evaluation: The break-up of evaluation of your program is same as Assignment 1 and as follows:

1. Program Design as explained in Report - 30%
2. The Graphs obtained and the corresponding analysis shown in the report - 30%
3. Program Execution - 30%
4. Code Documentation & Indentation - 10%.

Please make sure that you your report is well written since it accounts for 60% of the marks. Also note that no late submission will be accepted this time as it there will not be sufficient time to grade the assignments.