**IS 651, Distributed Systems**

**Homework #3**

**Due Mar 20, 2020 Answers to the written questions will be posted online after the deadline**

**140pts**

**Q1. (30pts)** Google file system.

Q1-1. (10pts) What are the benefits of having multiple data copies in google file system?

Q1-2. (20pts) Why does the client only need to talk to one chunkserver to read data? Is it possible that the client gets outdated data by connecting to one chunkserver?

**Q2.** **(20pts)** Explain the problems with the #4 Majority Votes approach for mutual exclusion. Draw a figure or write a scenario to explain your answers.

**Q3. (50pt)** Consistency

Are the following statements true or false? Briefly explain your answers.

Q3.A. Sequential consistency guarantees causal ordering.

Q3.B. Sequential consistency does not admit a scalable implementation because it requires a total ordering of all operations in the system.

Q3.C. Obeying all causal orderings is sufficient to guarantee convergence of state maintained by multiple replicas.

Q3.D. Causal consistency is sufficient to guarantee that an airline never overbooks its seats when multiple clients reserve seats on the same airplane.

Q3.E. Sequential consistency is sufficient to guarantee that an airline never overbooks its seats when multiple clients reserve seats on the same airplane.

**(Optional) Q4. (40pt)** Vector clock implementation

Based on either rpc or socket (from HW1), implement a service with vector clocks. (Feel free to use any programming language you are familiar with)

The system should have the following modules/properties. Use one script to represent all nodes. There is no need to have multiple scripts for different nodes.

1. Every node has a unique id, such as 0,1,2… (You can pre-specify the id of the nodes)
2. A vector clock updating function. (You can assume all nodes know the number of nodes in the system)
3. A ‘send’ module where if node i sends a message to j, j will receive a message with vector clock in it. (You can assume all nodes know the port numbers of each other for sending and receiving messages. The messages do not have to have any specific content other than the vector clock.)
4. During your testing, you can generate random workloads where each node sends a message to another random node after waiting for a random period of time. Remember to print the timestamp of each event (physical clock) and the logical clocks of each event so you can double check your answers.

**Please submit 1) your source code. Include screenshots in your answers. 2) a figure showing the happen-before diagram of all the events according to your screenshot.**

Note: You can use any method for serialization. Optional ones: pickle (python only), json, or manual serialization such as struct.pack

On the class website, a template (python3 only, called *VC-template*) is provided which can be your starting point. The run.sh bash script allows you to run multiple nodes on the same screen so you get the real-time sequence of all the events in a distributed system. There is a screenshot showing an example you can refer to.