Name (Last, First): ______________________________

This exam consists of 5 questions on 8 pages; be sure you have the entire exam before starting. The point value of each question is indicated at its beginning; the entire exam is worth 100 points. Individual parts of a multi-part question are generally assigned approximately the same point value: exceptions are noted. This exam is open text and notes. However, you may NOT share material with another student during the exam. No computing devices are allowed.

Be concise and clearly indicate your answer. Presentation and simplicity of your answers may affect your grade. **If I cannot read your answer easily you will not get credit**. Answer each question in the space following the question. If you find it necessary to continue an answer elsewhere, clearly indicate the location of its continuation and label its continuation with the question number and subpart if appropriate.

You should read through all the questions first, then pace yourself.

The questions begin on the next page.

Problem	Possible	Score
1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

1. (____________/20 points)

   Short Answer

   (a) What is the difference between synchronous networks and asynchronous networks as used in the leader election papers we read?

   (b) What is an advantage of using Gevent and Greenlets over conventional threads?

   (c) On an interview you are asked "Why is Paxos important?" What do you say?

   (d) What is the main strategy employed by Google MapReduce to enable MapReduce programs to process huge files that are spread across many disks in the Google File System?

2. (______________/20 points)

   Leader Election

   Recall that the Invitation Algorithm for leader election as described in *Elections in a Distributed Computing System* by Garcia-Molina can can handle asynchronous networks and deal with network partitions. However, in the Invitation Algorithm there can be more than one leader if some nodes are partitioned from other nodes. For some applications you do not want more than one leader. Describe how you would change the Invitation Algorithm so that an elected leader can only declare itself the leader and run the leader code (task) if it is in a partition with a majority of the participating servers. Give an English description of your approach, then show how to modify the algorithm as presented in the paper using the same notation as the paper. Note that your modification should still allow merging of partitioned groups of nodes. Hint: keep your solution as simple as possible.

3. (______________/20 points)

   Broadcasting a value with ZeroRPC

   One way to send a value or message from one server, say server A, to all other known servers is for server A to send a message (make a remote call) on each of the other servers. I provide the Python/ZeroRPC code to achieve this below.

   However, another approach is to order the servers into a logical ring and have server A send the message to server B and have server B send the message to server C, and so on, until the message arrives back at server A.

   (a) (15 points) Modify the code below or write new code that broadcasts a message using the ring approach. Be neat and explain your solution in English in addition to providing your code.

   (b) (5 points) Explain how to modify your ring algorithm so that it will proceed even if a node in the ring is down. For example, say you have nodes A, B, C, D, and E. If D is down, we still want the message to be passed to all the nodes that are up. You do not need to provide code, just an English description.

   You can continue you solution on the next page.

```
class Server(object):

    def __init__(self, addr, config_file='config'):
        self.addr = addr

        # Read config here
        ...

        self.n = len(self.servers)
        self.connections = []

        for i, server in enumerate(self.servers):
            if server == self.addr:
                self.i = i
                self.connections.append(self)
            else:
                c = zerorpc.Client(timeout=10)
                c.connect('tcp://' + server)
                self.connections.append(c)

    def broadcast_send(self, msg):
         for i, server in enumerate(self.servers):
             if i == self.i: continue
             try:
                self.connection.[i].broadcast.recv(msg)
            except zerorpc.TimeoutExpired:
                    print 'Timeout!'

    def broadcast_recv(self, msg):
            # Store msg
            self.msg = msg
            return
```

Continue problem 3 here.

4. (______________/20 points)

   GFS

   Recall the paper *The Google File System* by Ghemawat, Gobioff, and Leung. In distributed systems, replication is often used to increase reliability. The more copies of a chunk of data you have the less likely all the machines with a copy will crash simultaneously. However, replication is also used for performance.

   (a) (10 points) How can replication improve the performance of clients in GFS? Also explain how replication can improve MapReduce performance.

   (b) (10 points) What is the disadvantage of too much replication?

5. (______________/20 points)

   ZooKeeper

   (a) (10 points) Explain the difference between persistent and ephemeral z-nodes in ZooKeeper.

   (b) (10 points) Explain why the Zab protocol is not a good choice for reliably storing large amounts of data. That is, why not just use ZooKeeper for storing key/value pairs rather than implemented separate storage servers for Project 2? Provide evidence from the Zab paper.

Continue your answers here if necessary.