15-440 Distributed Systems
Homework 1 (6 problems)

Due: November 30, 11:59 PM via electronic handin
Hand in to Autolab in PDF format

November 29, 2011

1. You have set up a fault-tolerant banking service. Based upon an examination of other systems, you've decided that the best way to do so is to use Paxos to replicate log entries across three servers, and let one of your employees handle the issue of recovering from a failure using the log.

   The state on the replicas consists of a list of all bank account mutation operations that have been made, each with a unique request ID to prevent retransmitted requests, listed in the order they were committed.

   Assume that the replicas execute Paxos for every operation.[1] Each value that the servers agree on looks like "account action 555 transfers $1,000,000 from Mark Stehlik to David Andersen". When a server receives a request, it looks at its state to find the next unused action number, and uses Paxos to propose that value for the number to use.

   The three servers are S1, S2, and S3.

   At the same time:

   - S1 receives a request to withdraw $500 from A. Carnegie.
     - S1 picks proposal number 101 (the n in Paxos).
   - S2 receives a request to transfer $500 from A. Carnegie to A. Mellon.
     - S2 picks proposal number 102.

   Both servers lok at their lists of applied account actions and decide that the next action number is 15. So both start executing Paxos as a leader for action 15.

   (a) Each sequence below shows one initial sequence of messages of a possible execution of Paxos when both S1 and S2 are acting as the leader. Each message shown is received successfully. The messages are sent one by one in the indicated order. No other messages are sent until after the last message shown is received.

   Answer three questions about the final outcomes that could result from each of these sequences:

   - Is it possible for the servers to agree on the withdrawal as entry 15?
   - Is it possible for the servers to agree on the transfer as entry 15?
   - For each of these outcomes, explain how it either could occur or how Paxos prevents it.

   To be clear on the message terminology: The PREPARE message is the leader election message. RESPONSE is the response to the prepare message. ACCEPT says "you've agreed that I'm the leader, now take a value."

[1] In practice, most systems use Paxos to elect a primary and let it have a lease on the operations for a while, but that adds complexity to the homework problem.

Sequence 1:

```
S1 -> S1 PREPARE(101)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(101)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(101)
S3 -> S1 RESPONSE(nil, nil)

S2 -> S1 PREPARE(102)
S2 -> S2 PREPARE(102)
S2 -> S3 PREPARE(102)
... the rest of the Paxos messages.
```

Sequence 2:

```
S1 -> S1 PREPARE(101)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(101)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(101)
S3 -> S1 RESPONSE(nil, nil)

S1 -> S3 ACCEPT(101, ‘‘withdraw...’’)

S2 -> S1 PREPARE(102)
S2 -> S2 PREPARE(102)
S2 -> S3 PREPARE(102)
... the rest of the Paxos messages.
```

Sequence 3:

```
S1 -> S1 PREPARE(101)
S1 -> S1 RESPONSE(nil, nil)

S1 -> S2 PREPARE(101)
S2 -> S1 RESPONSE(nil, nil)

S1 -> S3 PREPARE(101)
S3 -> S1 RESPONSE(nil, nil)

S1 -> S3 ACCEPT(101, ‘‘withdraw...’’)
S1 -> S1 ACCEPT(101, ‘‘withdraw..’’)
S2 -> S1 PREPARE(102)
S2 -> S2 PREPARE(102)
S2 -> S3 PREPARE(102)
... the rest of the Paxos messages.
```

(b) Suppose one of the servers received an ACCEPT for a particular instance of Paxos (remember, each “instance” agrees on a value for a particular account event), but it never heard back about what the final outcome was. What steps should the server take to figure out whether agreement was reached and what the agreed-upon value was? Explain why your procedure is correct even if there are still active leaders executing this instance of Paxos.

2. In order to reduce load on the root and top level domain servers, local DNS servers cache queries. The length of time that a query is kept is referred to as the time to live (ttl). Caching also allows a DNS server to respond more quickly to queries for the same domain or host. In this problem you will compute downtimes for a configuration of servers.

   (a) Suppose a particular ISP has a set of servers with the following records:

```
-----------------------------------------
Server 1
Authoritative for CAR
Knows the following Authoritative Servers:
- HIT (Server 4)
- BLUE (Server 5)
- ODD (Server 6)
-----------------------------------------
Server 2
Replica of Server 1
-----------------------------------------
Server 3
Replica of Server 1
-----------------------------------------
Server 4
Authoritative for HIT
Knows the IP address of
- WWW.HIT.CAR
-----------------------------------------
Server 5
Authoritative for BLUE
Knows the IP address of
- WWW.BLUE.CAR
-----------------------------------------
Server 6
Authoritative for ODD
Knows the IP address of
- WWW.ODD.CAR
-----------------------------------------
```

   - The time to live for each NS record is 3 hours.
   - The timeout for each of the resolvers is 30 seconds. So for example, if a query comes in to Server 1, but Server 1 has crashed, the resolver that issued the query to Server 1 will wait for 30 seconds for a response before querying Server 2.
   - To simplify things a bit, the time for the resolver to respond to the client from the cache is 0. The time to respond to a query not in the cache is 1 second per other server that needs to be contacted.

```
For the following sequence of events at time t (in seconds):

t=0:   QUERY to Server 1 for WWW.BLUE.CAR
t=10:  QUERY to Server 2 for WWW.BLUE.CAR
t=15:  QUERY to Server 1 for WWW.ODD.CAR
t=20:  Server 1 crashes.
t=25:  QUERY to Server 1 for WWW.BLUE.CAR
t=60:  QUERY to Server 3 for WWW.HIT.CAR
t=65:  QUERY to Server 3 for WWW.HIT.CAR
t=100: QUERY to Server 3 for WWW.HIT.CAR
t=105: QUERY to Server 2 for WWW.BLUE.CAR
```

```
t=110: Server 6 crashes.
t=115: QUERY to Server 2 for WWW.ODD.CAR
```

What is the total latency for this set of queries? For this problem, we will define latency as the amount of time that the client has to wait for an answer, whether that answer is the IP address of the hostname or an error. Explain how you got your answer.

(b) Let's start with a clean slate with the same servers as in part a. This time t is in minutes.

```
Consider the following sequence of events:

t=0:   QUERY to Server 1 for WWW.BLUE.CAR
t=15:  The administrator for Server 5 changes the IP address record for WWW.BLUE.CAR
t=20:  QUERY to Server 1 for WWW.BLUE.CAR
t=30:  QUERY to Server 2 for WWW.HIT.CAR
t=40:  QUERY to Server 3 for WWW.BLUE.CAR
t=60:  Server 1 crashes.
t=80:  QUERY to Server 2 for WWW.HIT.CAR
t=85:  QUERY to Server 2 for WWW.BLUE.CAR
t=90:  Server 1 is restored. (same state as when it crashed)
t=100: QUERY to Server 1 for WWW.BLUE.CAR
t=200: QUERY to Server 1 for WWW.BLUE.CAR
```

How many times did the client receive the incorrect IP address for WWW.BLUE.CAR?

(c) As you saw in part b, caching has a drawback. What is this drawback?

3. For this problem you have to implement a barrier using Go channels. To understand how a barrier works, consider N goroutines (where N is a global, known constant), all running the following code concurrently:

```
for i := 0; i < 100; i++ {
    barrier()
    log.Printf("Epoch %d\n", i)
}
```

The barrier ensures that all N goroutines stop at the barrier before any can move past the barrier. As a result, messages from different epochs will not be intermixed.

Provide an implementation for the function *barrier()* using only Go channels for synchronization. You will also need to define and give the implementation of a different goroutine (a different function) dedicated to monitoring the barrier (let's call it *barrier_monitor()*). You don't need to write initialization code: consider that any channel (and any other variable) have been defined globally. Also, N is a global constant that has already been defined.

4. Skype uses a custom protocol to determine whether a user is logged in, where they are located, and what ports they are listening on. Suppose that you have set out to build your own peer-to-peer telephony system.

(a) Very briefly, how could you use the Chord distributed hash table to maintain a directory of users that was stored entirely on the end-users computers, with no centralized infrastructure?

(b) Your telephony system catches on very quickly. Soon you have 100,000 users worldwide. However, users start to complain that it takes a long time to perform directory lookups. Assume that the average latency between users is 100ms. Explain why are lookups taking so long and estimate how long they take.

5. This is the annoying open-ended question. Don't spend more than a half an hour on it for your own sanity. But do discuss it with a friend or three. You will not be able to find an exact answer. The course

staff doesn't know the correct answer. So come up with a decent estimate based upon sources you can find - and have fun with it! (This is good practice for job interviews. :-)

How many *total* entries are there in the domain name system? (including all levels deep - `www.garble.gobble.andrew.cmu.edu` is fair game).

Hint: You'll find some useful statistics on the number of *domain names* at `http://www.dailychanges.com/`. But remember, each domain may have many entries. And there are also country-specific domain names like .uk .de etc. So this page isn't enough on its own, but hopefully it'll get you thinking in the right direction.

6. For this problem, you will devise a method to solve the *all-pairs shortest path problem* via Map/Reduce. Suppose that we have a graph consisting of a set of nodes $\{1, 2, \ldots, n\}$, and a set of edges $\{e_1, e_2, \ldots, e_m\}$, where each edge is of the form (i, j, d), indicating that the edge is from node i to node j and is of length d, where d is a non-negative real number. Our goal is to compute for each pair of nodes i and j, a value $D(i, j)$, equal to either the length of the shortest path from i to j, or to ∞ if no such path exists.

   Devise an algorithm that can perform this computation using a total of $2 \log_2 n$ Map/Reduce steps.

   Hint: Formulate a scheme that computes a sequence of values of the form $D_k(i, j)$ equal to the shortest path from i to j containing at most 2^k edges. You can perform each iteration by adapting the matrix multiplication algorithm shown in the lecture of October 25 for this purpose. The trick is to define the "multiplication" and "addition" operations of matrix multiplication appropriately.