CS744 (Autumn 2017)

Endsem Review: Practice Problems

1. Consider the pseudocode to acquire and release a lock, based on the test-and-set atomic instruction.

```
acquire():
  while(1) {
    while(lock == 1);
    if(test_and_set(&lock) == 0) return;
    }

release():
  lock = 0;
```

   Consider the scenario where multiple threads of an application contend to use this lock across multiple CPU cores.

   (a) Is this lock a spinlock? That is, do contending threads consume CPU cycles while waiting to acquire the lock?

   (b) Explain what cache coherence traffic is generated when a new thread calls the acquire function when a thread on another core holds the lock.

   (c) Explain what cache coherence traffic is generated when a thread holding the lock releases the lock, when there are threads on other cores waiting for the lock.

   (d) What is the cache coherence traffic on the system bus when multiple threads are contending for a lock that is held on another core, in period when there are no new acquires or releases?

2. Consider a distributed system where two clients generate the following requests to read and write variables from a distributed shared memory. Each item below lists the request issued by the client, the time at which the request was issued by the client to the system (as per a global wall clock), and the time at which a positive acknoeledgement response was received at the client.

   - Client 0 issues two writes $x = 0$ and $y = 0$ at time $t = 0$, gets a response at $t = 0.5$.
   - Client 1 issues a write $x = 1$ at $t = 1$, and gets a response at $t = 1.5$.
   - Client 2 issues a write $y = 2$ at $t = 2$, and gets a response at $t = 2.5$.
   - Client 3 issues reads for x and y at time $t = 3$.
   - Client 4 issues writes $x = 4$ and $y = 4$ at time $t = 2.99$ and gets a response back sometime after $t = 3$.

State all possible values of x and y that can be obtained during the read at client 3, when the distributed system guarantees the following consistency models.

(a) Strict consistency.

(b) Atomic consistency / linearizability.

(c) Sequential consistency.

(d) Eventual consistency.

3. Consider a lockfree stack discussed in class, with the following code for push and pop.

```
push(node *n) {
  do
    n->next = top;
  while(!CAS(&top, n->next, n);
}

node *pop {
  node *result;
  while( (result=top) != NULL) //line X
    if(CAS(&top, result, result->next)) //line Y
      break;
  return result;
}
```

Consider the following series of operations made by 3 threads of an application on the stack.

- Thread 1 performs `push (aaa); push (aa); push(a); push (b); push (c);`
- Thread 2 performs `x = pop();`
- Thread 3 performs `y = pop(); z = pop(); push(y);`

Show the state of the stack (the nodes and all the pointers) after the following scenarios occur.

(a) The three threads run sequentially one after the other.

(b) Thread 1 runs first, and after it completes, threads 2 and 3 execute concurrently. More specifically, all operations of thread 3 run after thread 2 has executed line X of its pop function, has computed the arguments to the CAS function in line Y, but before the call to CAS has happened in thread 2.

(c) Thread 1 runs first, and after it completes, threads 2 and 3 execute concurrently like part (b). However, now, thread 2 runs in between the execution the first pop function invocation of thread 3.

4. Consider Figure 7 in the Raft paper. For each follower shown, explain a scenario where the follower would have ended up the situation shown in the figure. Also, based on the restriction of leader election, which of the followers (a) through (f) would have voted for the new leader?