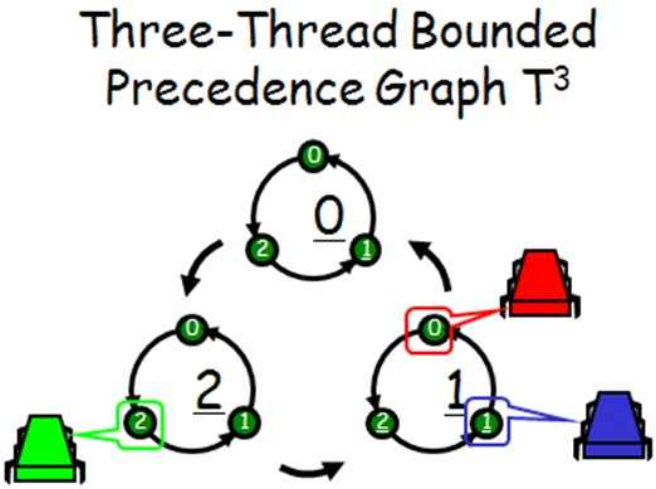
분산시스템 중간고사



1. Shown above, the bounded timestamping system T(3) is consistent when executed sequentially. What would happen if 3 threads run concurrently ? There are two cases of inconsistency, Show the scenario of those cases.

pid(22)=0, pid(11)=1, pid(10)=2.

(10<11<22, in sequential execution, next-TS() = 20)

```
next-TS() {
    i =  pid; j = (i+1) mod 3; k = (i+2) mod 3;
    read TS(i);      read TS(j);        read TS(k);
    calculate-new-TS();      update TS(i);
    }
```

2. This is the abstraction of LL/SC . Please show thar its consensus numbe is the same as CAS.

```
public class LLSC {
 Object value;
 boolean busy;
 public LLSC(Object object) {
  this.value = object;
  this.busy  = false
 }
 public synchronized Object LL() {
  this.busy = true
  return this.value
 }
 public synchronized
  boolean SC(Object newValue) {
  if (this.busy) {
   this.busy  = false
   this.value = newValue
   return true
  } else
   return false
 }
}
```

3. Using SRSW, MV, Regular registers and Timestamp, we can construct a SRSW, MV, Atomic register. However, timestamp always may overflow, Can you use the bounded precedence graph T^2 ?

4. We can do n-thread consensus using CAS. When you have to do it again, can you use the same register again ?

5. How about LL/SC ?

6. We can solve 2-thread consensus using atomic 2-assignment. Would it be possible to extend it to solve 4-thread consensus ? 4 threads can be grouped in two and do rhe 2-thread consensus twice in the 1st phase. Then the winners can do another 2-thread consensus. Is this correct ?

7. How many Safe, SRSW, Boolean Register to construct 2-reader/2-writer, atomic, 4bit Register ? (do not count the registers for timestamp)

8. filter lock can be implemented as follows. Assume we have 3 threads (1,2,3).
Can thread 1 be overtaken by others ? Show the case when 1 is overtaken 3 times..

```
public void lock() {
 for (int L = 1; L < n; L++) {
   level[i]  = L;
   victim[L] = i;
   while ((∃ k != i level[k] >= L) && victim[L] == i );
 } }
public void unlock() {
  level[i] = 0;
```