Universal Consensus

Karan Agrawal

February 2024

1 Algorithm Explanation

1. **Classes**

   - **Consensus Template**: A simple template class with only one function declaration, *decide()*.
   - **Consensus Protocol**: A simple template class that inherits the **Consensus** class and adds a new function, *propose()*.
   - **CASConsensus**: Class for implementing compare_and_exchange.
   - **Response**: Class to store responses of a function.
   - **Invoc**: Class to represent a function with corresponding arguments.
   - **Seqobject**: Class to implement a stack and queue with a function *apply()* to apply the corresponding invoke on an object.
   - **Node**: Class to represent a node of a linked list in Universal Consensus.
   - **Universal**: Class to implement Universal wait-free Consensus.
   - **LFUniversal**: Class to implement Universal lock-free Consensus.

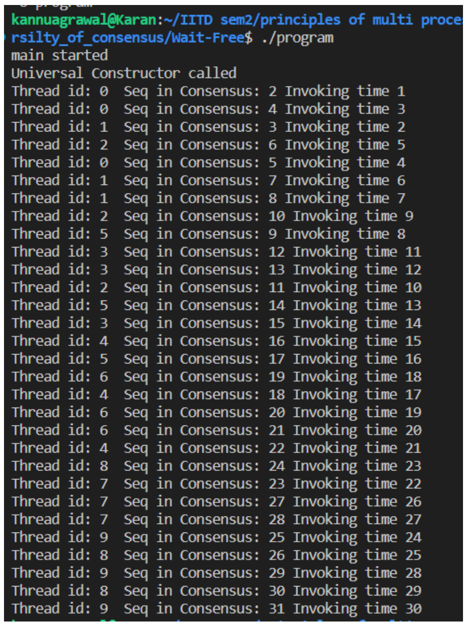
2. **Common Main Function to Implement Both Variants Lock-Free and Wait-Free of Consensus**

   - Firstly, create an object of Universal Consensus for N threads in the corresponding variants.
   - Then create N threads.
   - Each thread applies an invoc three times on the Universal/LFUniversal object so that we can differentiate between the lock-free and wait-free Consensus behavior.
   - Each call will return:
     - The sequence number at which the invoc of this call is applied to SeqObject.
     - The sequence number of the **last node of the universal list** when the thread enters the Consensus system.

2 Results

1. **Wait Free**

    - In the result of the wait-free approach, it is evident that each thread completes its operation of invoking in a finite number of steps, independent of the behavior of other threads.
    - because the difference between the sequence number when the operation completes and when it enters the Consensus system is less than or equal to N (where N = 10).

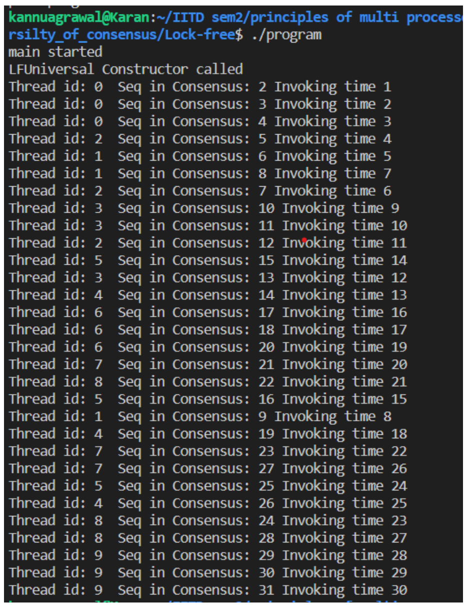


```
kannuagrawal@Karan:~/IITD sem2/principles of multi proce
rsilty_of_consensus/Wait-Free$ ./program
main started
Universal Constructor called
Thread id: 0  Seq in Consensus: 2 Invoking time 1
Thread id: 0  Seq in Consensus: 4 Invoking time 3
Thread id: 1  Seq in Consensus: 3 Invoking time 2
Thread id: 2  Seq in Consensus: 6 Invoking time 5
Thread id: 0  Seq in Consensus: 5 Invoking time 4
Thread id: 1  Seq in Consensus: 7 Invoking time 6
Thread id: 1  Seq in Consensus: 8 Invoking time 7
Thread id: 2  Seq in Consensus: 10 Invoking time 9
Thread id: 5  Seq in Consensus: 9 Invoking time 8
Thread id: 3  Seq in Consensus: 12 Invoking time 11
Thread id: 3  Seq in Consensus: 13 Invoking time 12
Thread id: 2  Seq in Consensus: 11 Invoking time 10
Thread id: 5  Seq in Consensus: 14 Invoking time 13
Thread id: 3  Seq in Consensus: 15 Invoking time 14
Thread id: 4  Seq in Consensus: 16 Invoking time 15
Thread id: 5  Seq in Consensus: 17 Invoking time 16
Thread id: 6  Seq in Consensus: 19 Invoking time 18
Thread id: 4  Seq in Consensus: 18 Invoking time 17
Thread id: 6  Seq in Consensus: 20 Invoking time 19
Thread id: 6  Seq in Consensus: 21 Invoking time 20
Thread id: 4  Seq in Consensus: 22 Invoking time 21
Thread id: 8  Seq in Consensus: 24 Invoking time 23
Thread id: 7  Seq in Consensus: 23 Invoking time 22
Thread id: 7  Seq in Consensus: 27 Invoking time 26
Thread id: 7  Seq in Consensus: 28 Invoking time 27
Thread id: 9  Seq in Consensus: 25 Invoking time 24
Thread id: 8  Seq in Consensus: 26 Invoking time 25
Thread id: 9  Seq in Consensus: 29 Invoking time 28
Thread id: 8  Seq in Consensus: 30 Invoking time 29
Thread id: 9  Seq in Consensus: 31 Invoking time 30
```

Figure 1: Wait-free

2. **Lock Free**

    - In the result of the lock-free approach, it is not necessary that each thread completes its operation of invoking in a finite number of steps. Other threads can starve it.
    - We can analyze this result that the difference between the sequence number when the operation completes and when it enters the Consensus system may be greater than N (where N = 10).It is not happening because the threads complete their tasks very quickly.



```
kannuagrawal@Karan:~/IITD sem2/principles of multi process
rsilty_of_consensus/Lock-free$ ./program
main started
LFUniversal Constructor called
Thread id: 0  Seq in Consensus: 2 Invoking time 1
Thread id: 0  Seq in Consensus: 3 Invoking time 2
Thread id: 0  Seq in Consensus: 4 Invoking time 3
Thread id: 2  Seq in Consensus: 5 Invoking time 4
Thread id: 1  Seq in Consensus: 6 Invoking time 5
Thread id: 1  Seq in Consensus: 8 Invoking time 7
Thread id: 2  Seq in Consensus: 7 Invoking time 6
Thread id: 3  Seq in Consensus: 10 Invoking time 9
Thread id: 3  Seq in Consensus: 11 Invoking time 10
Thread id: 2  Seq in Consensus: 12 Invoking time 11
Thread id: 5  Seq in Consensus: 15 Invoking time 14
Thread id: 3  Seq in Consensus: 13 Invoking time 12
Thread id: 4  Seq in Consensus: 14 Invoking time 13
Thread id: 6  Seq in Consensus: 17 Invoking time 16
Thread id: 6  Seq in Consensus: 18 Invoking time 17
Thread id: 6  Seq in Consensus: 20 Invoking time 19
Thread id: 7  Seq in Consensus: 21 Invoking time 20
Thread id: 8  Seq in Consensus: 22 Invoking time 21
Thread id: 5  Seq in Consensus: 16 Invoking time 15
Thread id: 1  Seq in Consensus: 9 Invoking time 8
Thread id: 4  Seq in Consensus: 19 Invoking time 18
Thread id: 7  Seq in Consensus: 23 Invoking time 22
Thread id: 7  Seq in Consensus: 27 Invoking time 26
Thread id: 5  Seq in Consensus: 25 Invoking time 24
Thread id: 4  Seq in Consensus: 26 Invoking time 25
Thread id: 8  Seq in Consensus: 24 Invoking time 23
Thread id: 8  Seq in Consensus: 28 Invoking time 27
Thread id: 9  Seq in Consensus: 29 Invoking time 28
Thread id: 9  Seq in Consensus: 30 Invoking time 29
Thread id: 9  Seq in Consensus: 31 Invoking time 30
```

Figure 2: Lock-free