3.3 Lecture Summary

3.3 Sieve of Eratosthenes

Lecture Summary: In this lecture, we studied how to use actors to implement a pipelined variant of the *Sieve of Eratosthenes* algorithm for generating prime numbers. This example illustrates the power of the Actor Model, including dynamic creation of new actors during a computation.

To implement the Sieve of Eratosthenes, we first create an actor, *Non-Mul-2*, that receives (positive) natural numbers as input (up to some limit), and then filters out the numbers that are multiples of 2. After receiving a number that is not a multiple of 2 (in our case, the first would be 3), the *Non-Mul-2* actor creates the next actor in the pipeline, *Non-Mul-3*, with the goal of discarding all the numbers that are multiples of 3. The *Non-Mul-2* actor then forwards all non-multiples of 2 to the *Non-Mul-3* actor. Similarly, this new actor will create the next actor in the pipeline, *Non-Mul-5*, with the goal of discarding all the numbers that are multiples of 5. The power of the Actor Model is reflected in the dynamic nature of this problem, where pieces of the computation (new actors) are created dynamically as needed.

A Java code sketch for the *process()* method for an actor responsible for filtering out multiples of the actor's "local prime" in the Sieve of Eratosthenes is as follows:

```
1  public void process(final Object msg) {
2    int candidate = (Integer) msg;
3    // Check if the candidate is a non-multiple of the "local prime".
4    // For example, localPrime = 2 in the Non-Mul-2 actor
5    boolean nonMul = ((candidate % localPrime) != 0);
6    // nothing needs to be done if nonMul = false
7    if (nonMul) {
8      if (nextActor == null) {
9        . . . // create & start new actor with candidate as its local prime
10     }
11     else nextActor.send(msg); // forward message to next actor
12   }
13 } // process
```

Optional Reading:

1. Wikipedia article on the Sieve of Eratosthenes problem