Project4 Report

**Group Member:**

Tao Yu (UFID: 66115481) & Yue Yu (UFID: 47084095)

**Overview**

In this project, we have revised our Push-Sum algorithm simulator developed in project 2 and added the logging facility to it. Based on such log files generated during execution, we can trace the execution of the Push-Sum simulator. The logging facility is undoubtedly important for not only debugging, but also understanding the Akka framework better.

**Base Project**

In this project, I reuse the push-sum code in project2. In project2, the push-sum algorithm is to calculate the average of the value of all the nodes in the network. The core part of push-sum algorithm is message transmitting and when the average of one node varies within 10-10 in 3 consecutive rounds, the actor terminates. In this project, we add the actorlog.scala as the logging facility. The logging facility logs the initialization process, the message received, the message sent, the current value of the actor and timestamp of each message. Each worker has its own log file, which is named after its worker+its id+.log+date. The boss has a special log file called boss.log+date.

In the actorlog.scala, I implemented a class called actorlog. This class has a constructor with a parameter filename, which specifies the name of the log file and a variable writer, which creates an output stream to its log file. This class has two functions. The log function does the logging and the exit function exit the output stream to the file. Each actor maintains an object of the actorlog class. Every time it receives a message, it logs corresponding information. When the system is to shut down. Each actor will receive a terminate message and the actor will call the exit function of its instance of actorlog to break the out stream to its log file.

**Bonus Project**

As for the bonus part of this project, we have implemented the logic for adding the logging facility totally within the original Push-Sum simulator code. We have created a new trait named Log, which inherited by both of the PushsumWorker and the Boss actors. There is only a single method within this trait, which is logging(). It can log into different log files with appropriate contents based on the arguments sent from either the PushsumWorker or the Boss actor. In our code, such arguments are just a group of case classes that inherited from a same father trait. Each of such case classes contains all of the information needed for logging. Therefore, based on the hierarchy described above, we just let PushsumWoker and Boss actors invoke the logging method whenever they send or receive messages.

**How to run our simulator**

1. project4.scala (base project)

$ scala project4 [number\_of\_nodes] [topology] [algorithm]

number\_of\_nodes -> an integer

topology -> full, 2D, im2D, line

algorithm -> push-sum

2. project4-bonus.scala (bonus project)

$ scala project4bonus [number\_of\_nodes] [topology]

number\_of\_nodes -> an integer

topology -> full, 2D, im2D, line