Alan Rosenthal (alan_rosenthal@student.uml.edu)
91.515 Operating Systems I
Project #3
17 December 2012

Degree of Success
I successfully was about to complete the majority of the project. I was able to get a fully connected network of four nodes plus a buffer mananger. I was also able to fully implement Lamport's algorithm. I also completed the majority of the code for the buffer manager. However, I was not able to complete the project due to time constraints. I did not implement the code for the consumers and producers.

I would say I completed the most challenging parts of the assignment. This is how I would grade myself:

Fully connected network	45% / 45%
Buffer manager	15% / 20%
Lamport's Algorithm	25 / 25%
Consumer & Producer Code	0% / 10%
Total	85% / 100%

Description of Approach
<u>Completely Connected Network</u>
I set up a completely connected network by having different nodes do one of the following a) listen for all connections, b) connect to the listening nodes, accept some connections, c) connect to all listening nodes.
I took advantage of the fact that all nodes on the 91308-0X machines mounted the same folder. I created a shell script that would write the address of the nodes to a file hostfile.txt and use the information in the file to connect to the other nodes in the network. I would run the following commands in order.
```
./connect_to_nodes.sh BM
./connect_to_nodes.sh 0
./connect_to_nodes.sh 1
./connect_to_nodes.sh 2
./connect_to_nodes.sh 3
```
A prompt would then come up saying "All nodes present, press enter to connect!"
I would press enter on each terminal in the same order. The script would launch the node controllers and buffer managers programs, which would connect to each other creating a completely connected network.

<u>Buffer Manager</u>
The buffer manager node was mostly complete. I had set up full data structures for the donut ring buffers. I was also able to communicate fully with the node controllers. Since I was still

in the development phase I had set up a slightly different spec then was requested in the assignment. The buffer manager had a lifeline connection to the node controllers instead of accepting and disconnecting connections from node controller. This was just to eliminate possible networking issues during development.

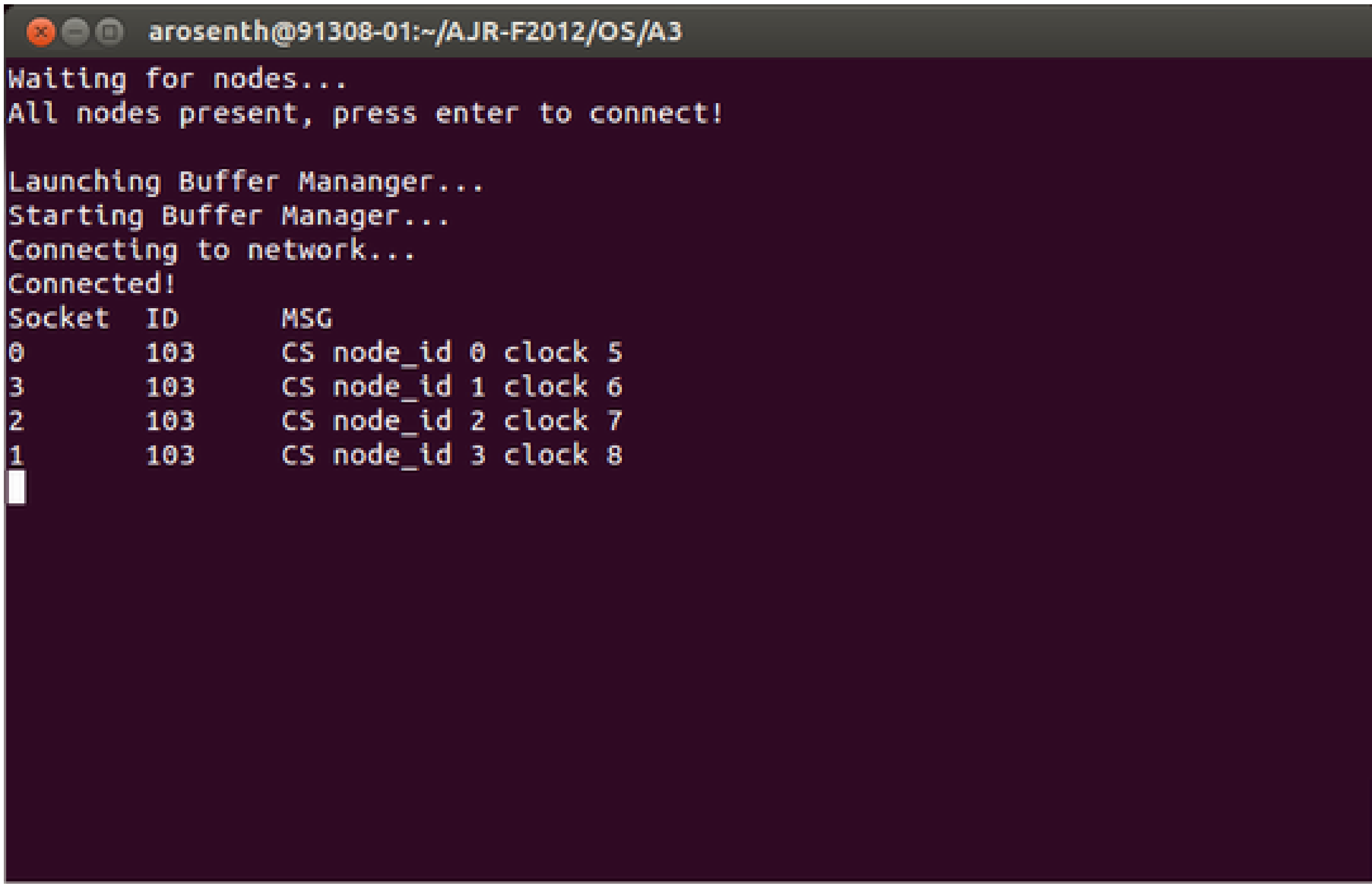
Lamport's Algorithm
I created a data structure for lamport's algorithm in the node controller program. Essentially it kept track of the logical clock as well as a pseudo linked list of requests and replies.
Every time the node controller received a REQUEST it would put it at the end the the queue and shift it towards the front until it was in the correct space (sorted by clock then by node-id). This would ensure that queues on different nodes would remain the same.
Every time the node controller received a REPLY it would file it under the proper REQUEST. Once a REQUEST had the sufficient amount of REPLY messages, it would start the CS. As I was still in the development phase of the project, the CS was just a message to the buffer manager node. The buffer manager would send back a test message, which would trigger a RELEASE message.
Every time the node controller received a RELEASE message it would remove the proper REQUEST from the head of the queue.

Consumer & Producer
Due to time constraints I was unable to complete the consumer and producers programs. I was planning on using the pthread library. I did not foresee this being an issue since I had my code from A1 and A2.

Problems unable to be resolved
I was unable to do the required testing because I was unable to finish the project due to time constraints.

I had a major issue with the read() function. It was blocking since there was no information coming from the sockets. I set the file descriptor using the fcntl() function to be nonblocking. However this caused read() to return -1 and errno was set. I was unaware that this was the design of the read() and I was supposed to catch errno == EAGAIN. This caused a significant delay.

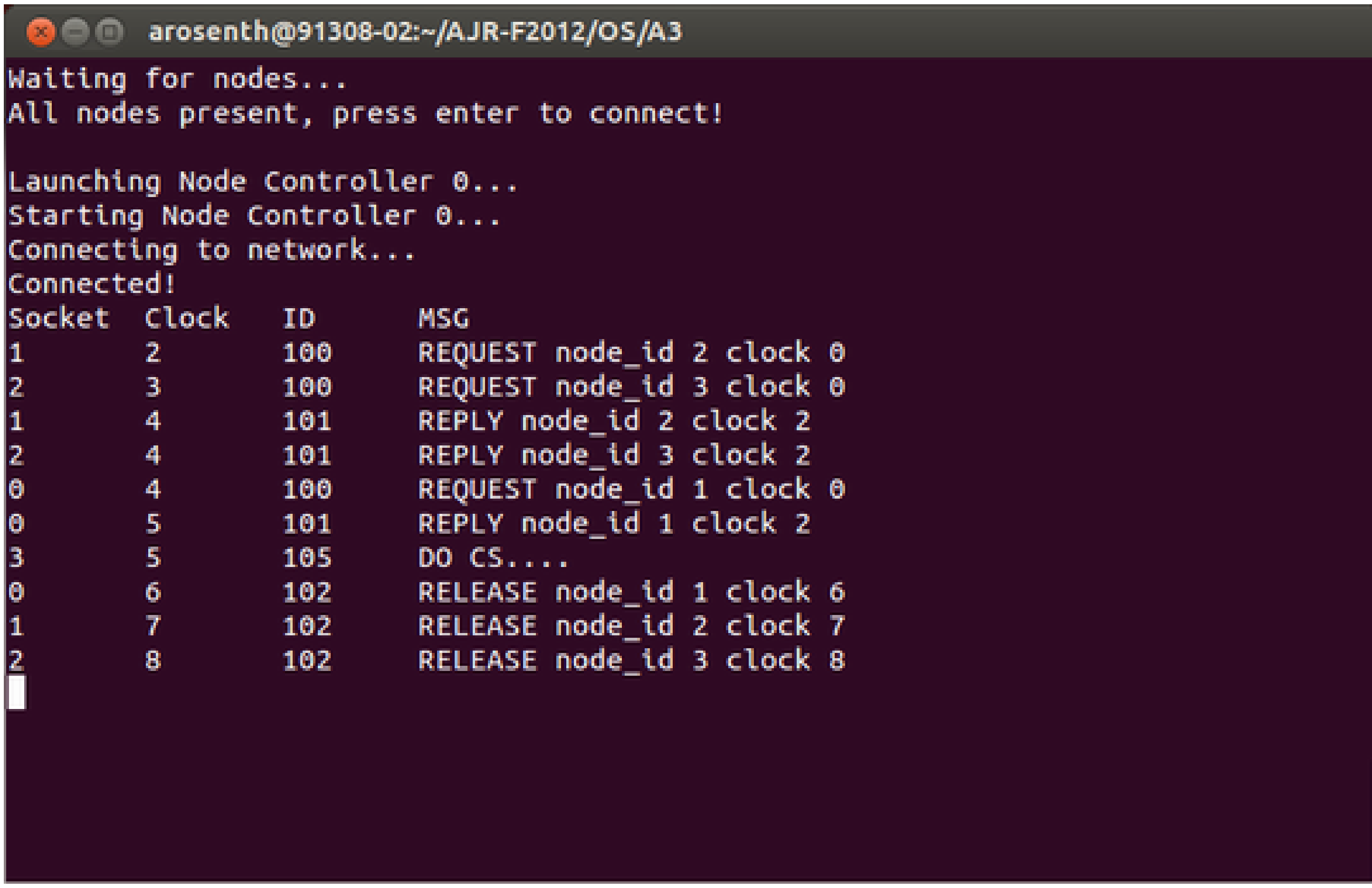
This was my first experience using sockets and I did not fully realize how complicated they were. I had a few issues with TCP connections being in TIME_WAIT and not closing. I did not have sufficient administrative privileges to forcibly close the connection. I would have to wait a few minutes for the OS to decide the connection was in fact dead.
Screenshots
Buffer Manager


arosenth@91308-01:~/AJR-F2012/OS/A3

```
Waiting for nodes...
All nodes present, press enter to connect!

Launching Buffer Mananger...
Starting Buffer Manager...
Connecting to network...
Connected!
Socket  ID      MSG
0       103     CS node_id 0 clock 5
3       103     CS node_id 1 clock 6
2       103     CS node_id 2 clock 7
1       103     CS node_id 3 clock 8
```

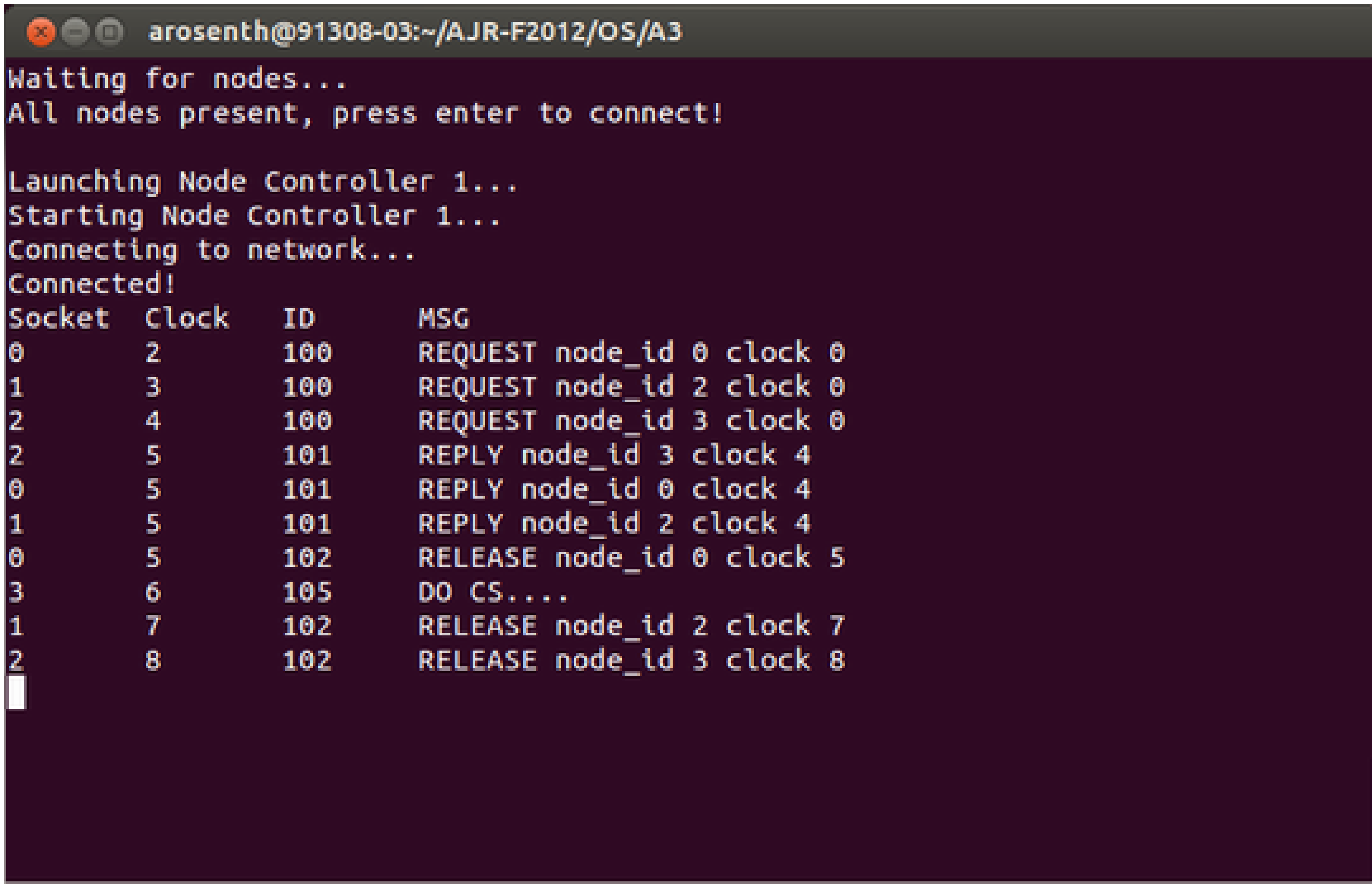
Node Controller 0



```
Waiting for nodes...
All nodes present, press enter to connect!

Launching Node Controller 0...
Starting Node Controller 0...
Connecting to network...
Connected!
Socket  Clock   ID      MSG
1       2       100     REQUEST node_id 2 clock 0
2       3       100     REQUEST node_id 3 clock 0
1       4       101     REPLY node_id 2 clock 2
2       4       101     REPLY node_id 3 clock 2
0       4       100     REQUEST node_id 1 clock 0
0       5       101     REPLY node_id 1 clock 2
3       5       105     DO CS....
0       6       102     RELEASE node_id 1 clock 6
1       7       102     RELEASE node_id 2 clock 7
2       8       102     RELEASE node_id 3 clock 8
```

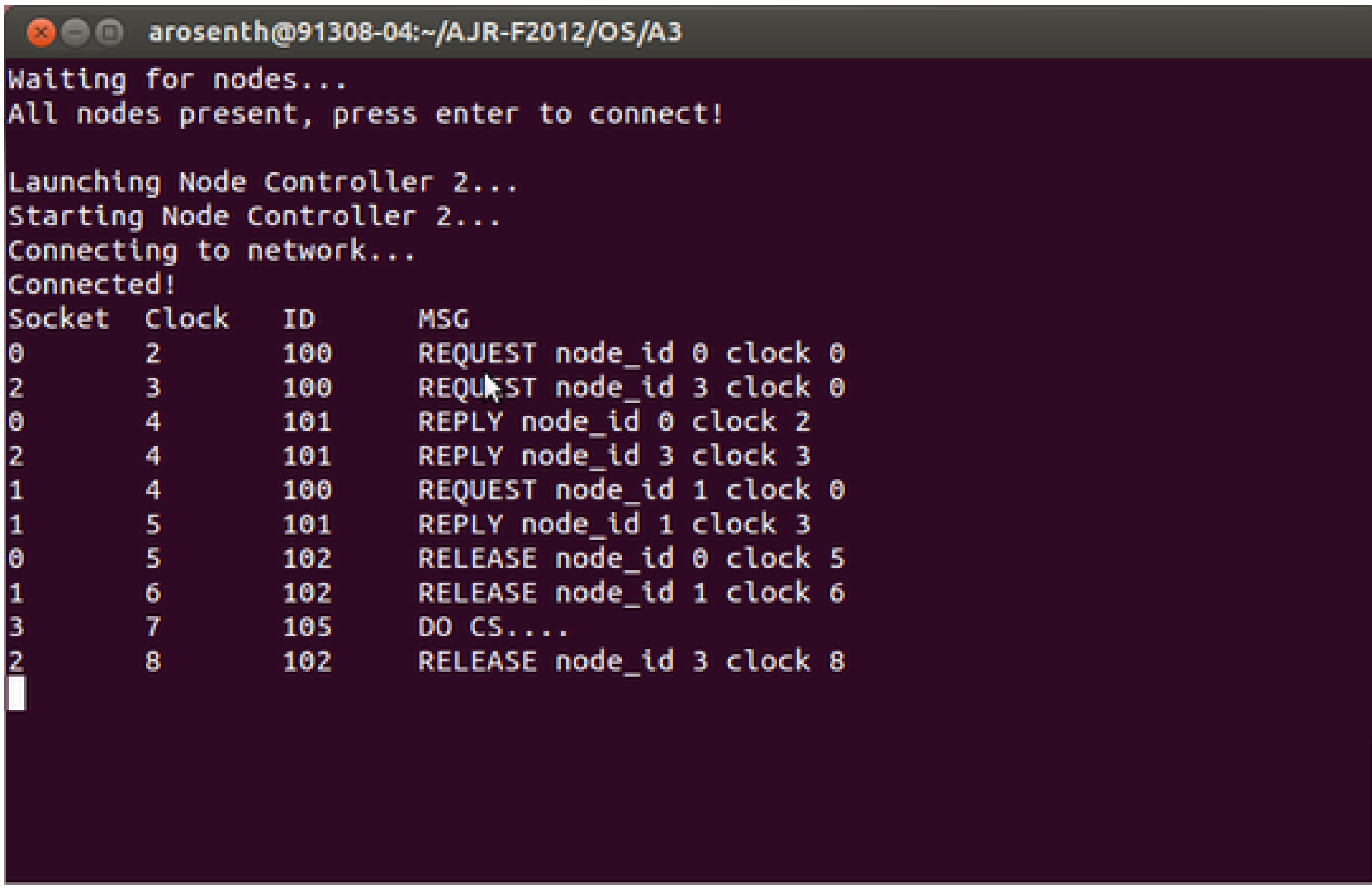
Node Controller 1



```
arosenth@91308-03:~/AJR-F2012/OS/A3
Waiting for nodes...
All nodes present, press enter to connect!

Launching Node Controller 1...
Starting Node Controller 1...
Connecting to network...
Connected!
Socket  Clock   ID      MSG
0       2       100     REQUEST node_id 0 clock 0
1       3       100     REQUEST node_id 2 clock 0
2       4       100     REQUEST node_id 3 clock 0
2       5       101     REPLY node_id 3 clock 4
0       5       101     REPLY node_id 0 clock 4
1       5       101     REPLY node_id 2 clock 4
0       5       102     RELEASE node_id 0 clock 5
3       6       105     DO CS....
1       7       102     RELEASE node_id 2 clock 7
2       8       102     RELEASE node_id 3 clock 8
```

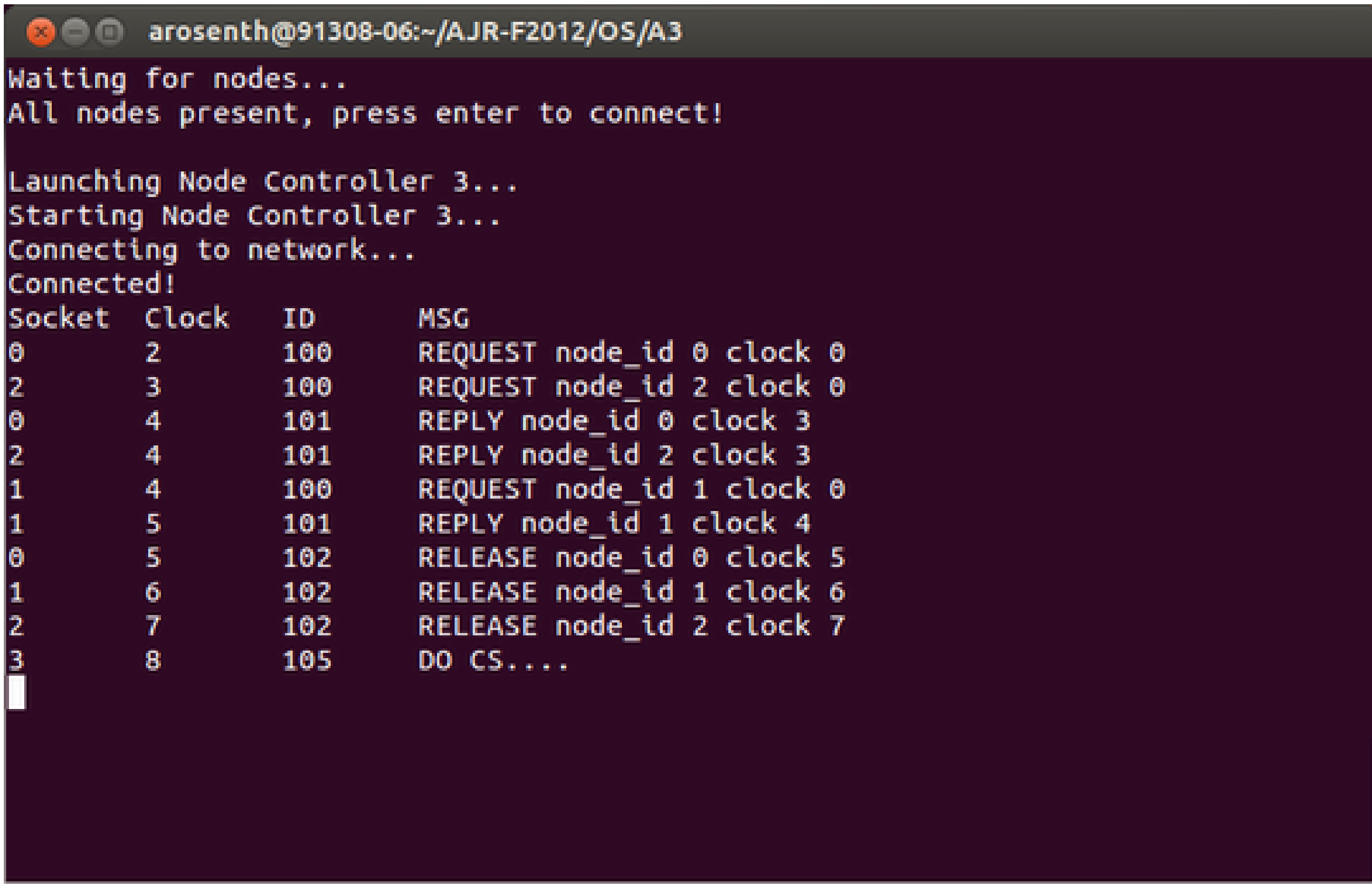
Node Controller 2



```
arosenth@91308-04:~/AJR-F2012/OS/A3
Waiting for nodes...
All nodes present, press enter to connect!

Launching Node Controller 2...
Starting Node Controller 2...
Connecting to network...
Connected!
Socket  Clock   ID      MSG
0       2       100     REQUEST node_id 0 clock 0
2       3       100     REQUEST node_id 3 clock 0
0       4       101     REPLY node_id 0 clock 2
2       4       101     REPLY node_id 3 clock 3
1       4       100     REQUEST node_id 1 clock 0
1       5       101     REPLY node_id 1 clock 3
0       5       102     RELEASE node_id 0 clock 5
1       6       102     RELEASE node_id 1 clock 6
3       7       105     DO CS....
2       8       102     RELEASE node_id 3 clock 8
```

Node Controller 3



arosenth@91308-06:~/AJR-F2012/OS/A3

```
Waiting for nodes...
All nodes present, press enter to connect!

Launching Node Controller 3...
Starting Node Controller 3...
Connecting to network...
Connected!
Socket  Clock   ID      MSG
0       2       100     REQUEST node_id 0 clock 0
2       3       100     REQUEST node_id 2 clock 0
0       4       101     REPLY node_id 0 clock 3
2       4       101     REPLY node_id 2 clock 3
1       4       100     REQUEST node_id 1 clock 0
1       5       101     REPLY node_id 1 clock 4
0       5       102     RELEASE node_id 0 clock 5
1       6       102     RELEASE node_id 1 clock 6
2       7       102     RELEASE node_id 2 clock 7
3       8       105     DO CS....
```