Jonathan Miu

Distributed Computing

July 26, 2018

To run:

compile with the command: g++ peer.cpp -o peer -lpthread

and execute using ./peer x (with x being replaced by the peer ID)

A few notes about the execution:

There are some really interesting bugs in this program, most of which are from having to create workarounds to other problems. A major problem is occasionally a peer will be shown as leaving a critical section twice. Obviously this isn't possible. This problem is due to having to package together the "left critical section" message with the follow-up "reply" message. The reason for this is that when these two messages were sent separately, the reply message would never arrive, causing a deadlock. Additionally, the "sectionion" bug is caused by this as well.

As you can see in the screenshot, sometimes digits are added to the localclocks. This has yes to affect message ordering, but it is possible. I have no idea what causes this. If debug messages are enabled, you can see that the character array itself is correct, but for some reason the resulting integer is converted incorrectly.

Overview:

This program begins be sending each peer a "request" asking for the critical section, and waits until all peers "reply" before entering. If a peer is waiting to enter themselves, then the lower local clock gets to enter first. If the local clocks are the same, then the lower peer ID gets to enter first. (it should be noted that it is impossible to have the same peer ID)

While a simple concept, this proved rather difficult to implement as contradicting messages and occasional bugs and errors got in the way. The most difficult of which was the "peer replies to itself" wherein a peer would send the critical "reply" message to itself instead of the peer that requested it. This inevitably causes a deadlock as a single peer is stuck waiting for a message that will never come.

Note:

gcc (Debian 7.3.0-1) 7.3.0