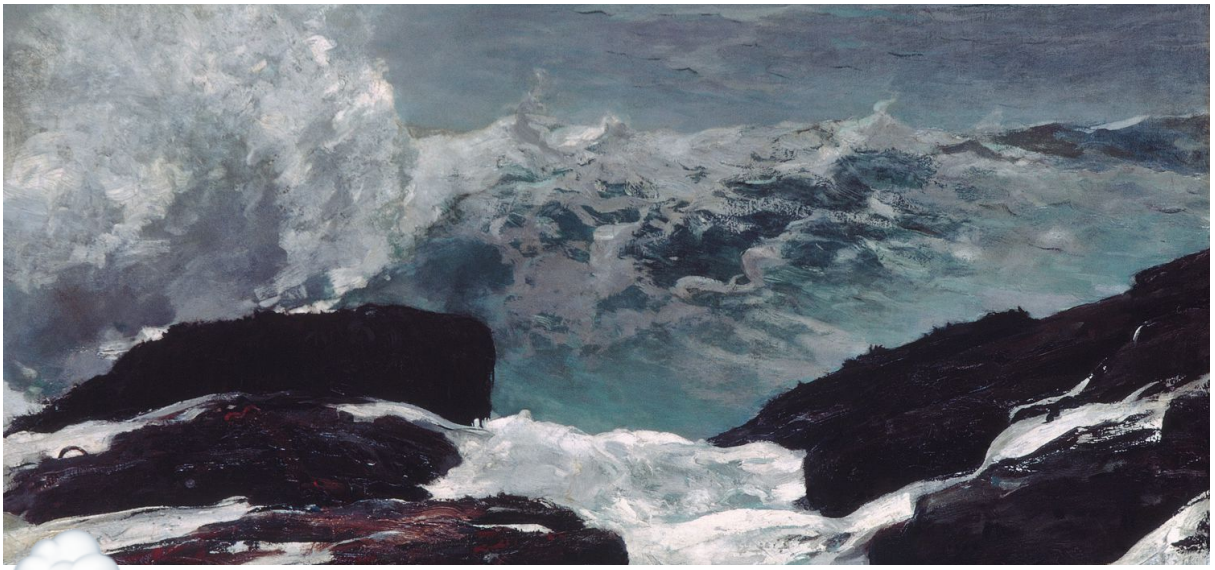


Cloud Computing Project

Team:

Akhil Kumar M (20CS01046)

Sidhartha Sai M (20CS01006)

Vinita Katlamudi (20CS01024)

Problem statement:

The project requires implementation of Lamport’s Mutual exclusion using logical clock. The project has to be implemented via sockets because you need to communicate between 3 devices. The critical section can be assumed to be access to a file on any one of the 3 machines. There need to be two threads per device. One thread shall create local event and send event; the send event should only be for mutual exclusion request. The other thread need to act as port listener and hence, receiver thread. The only catch here is to synchronise between sender and receiver thread to update the logical clock. The implementation needs to be done only in C++ or Java.

Approach:

1. Each Client has a Listener thread that listens all the time and two Senders that can be used while making request to both the other clients.
2. Accessing the File can be done by dedicating a socket on the system that has the file.
    - Either through the client or a seperate process; I chose to do it as a seperate process.
3. All the devices can access the CS as long as the socket is open.
4. CriticalSection.java handles can access to the text file called Critical_Section.txt
5. Using 3 locks and semaphores, maintain safe access.
6. Timestamps are saved and updated at every event.
7. Each Client maintains a Priority Queue for CS access and it's updated according to the responses and requests and increment resno.

Lamport's Algorithm for Mutual Exclusion:

Algorithm:

- **To enter Critical section:**
    - When a site S_i wants to enter the critical section, it sends a request message **Request(ts_i, i)** to all other sites and places the request on **request_queue_i**. Here, Ts_i denotes the timestamp of Site S_i
    - When a site S_j receives the request message **REQUEST(ts_i, i)** from site S_i, it returns a timestamped REPLY message to site S_i and places the request of site S_i on **request_queue_j.**
- **To execute the critical section:**
    - A site S_i can enter the critical section if it has received the message with timestamp larger than **(ts_i, i)** from all other sites and its own request is at the top of **request_queue_i**
- **To release the critical section:**

- When a site S_i exits the critical section, it removes its own request from the top of its request queue and sends a timestamped **RELEASE** message to all other sites.
- When a site S_j receives the timestamped **RELEASE** message from site S_i, it removes the request of S_i from its request queue.

As you can see in the figures given,
Device 1 requests first, REQs' both Device 2 and Device 3

Device 2 and Device 3 send RES; Device 1 receives them;

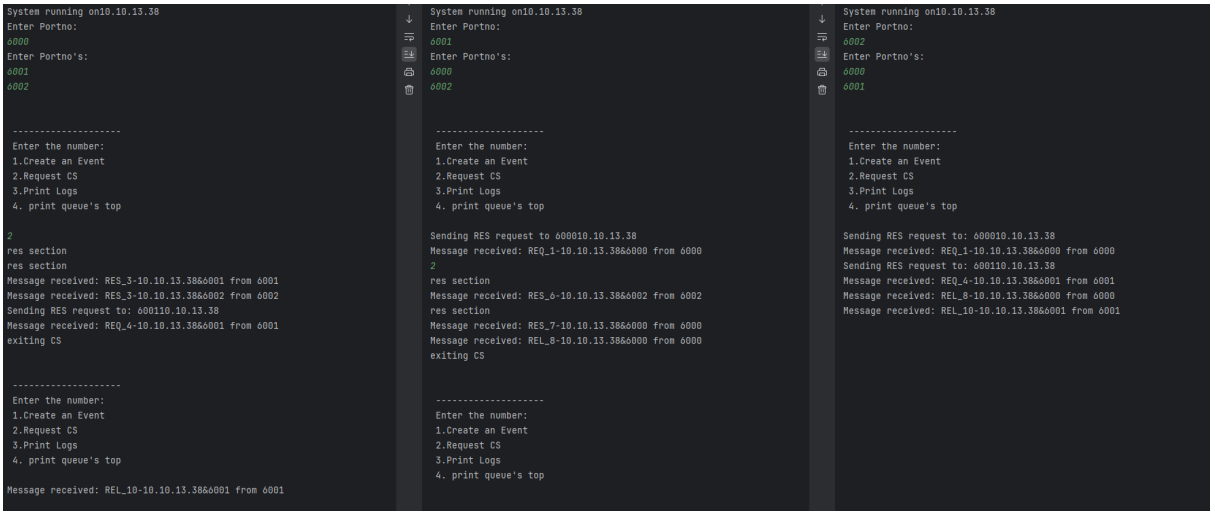
Device 1 accesses CS

Device 2 requests for CS, only receives 1 reply

Device 1 completes CS; sends REL(release) to both others and RES to Device 2

Device 2 has received 2 RES, accesses CS

Completes CS and sends REL to both.



As you can see the ports used were on the same device but different ports; but we create the sockets in a different device and use them, by hardcoding the IP address of the said devices.

```
51
52      // can be changed to work in multiple devices
53      add1 = "localhost";
54      add2 = "localhost";
55      add_cs="localhost";
56      port_cs=8080;
57      commun();
58
```