Programação Distribuída - Trabalho 2

O segundo trabalho da disciplina consiste em desenvolver um programa distribuído que implemente relógios vetoriais para ordenação de eventos (com ordem total). O programa deve receber como entrada um arquivo de configuração e um número que identifica uma das linhas do arquivo de configuração. Todos os processos devem possuir uma cópia desse arquivo. Cada linha do arquivo de configuração terá o seguinte formato:

```
id host port chance events min_delay max_delay
```

- *id* é um número inteiro que identifica o processo;
- *host* é o hostname ou endereço IP da máquina (nodo) que executa o processo;
- *port* é o número da porta que o processo vai escutar;
- *chance* é uma probabilidade (entre 0 e 1) da ocorrência de um evento de envio de mensagem. Por exemplo, o valor 0.2 significa 20% de probabilidade de ser realizado um envio de mensagem, sendo os 80% restantes eventos locais;
- *events* é o número de eventos que serão executados nesse nodo (recomenda-se aproximadamente 100 eventos);
- *min_delay* é o tempo mínimo de intervalo entre eventos (recomenda-se valores entre 100 e 300 ms);
- *max_delay* é o tempo máximo de intervalo entre eventos (recomenda-se valores entre 350 e 750 ms).

Execução do programa e descrição do algoritmo:

É importante que um mecanismo de sincronização inicial seja implementado para que todos os processos iniciem a execução do algoritmo ao mesmo tempo. Para isso, pode-se utilizar um grupo multicast. Cada nodo pode executar um evento local ou enviar uma mensagem para outro nodo, de acordo com a probabilidade estabelecida em sua configuração. Utilize datagramas para o envio de mensagens.

- Para eventos locais, incremente o relógio local;
- Para envio de mensagens, escolha aleatoriamente entre um dos outros nodos definidos no arquivo de configuração e envie uma mensagem contendo também o valor do relógio.

Devem ser gerados eventos com intervalo de *min_delay* a *max_delay*, e cada nodo deve executar diversos desses eventos (de acordo com a configuração, podendo ser um evento local ou envio de mensagem) e depois terminar sua execução. Pode ocorrer de um nodo tentar enviar uma mensagem para um nodo que já terminou sua execução, neste caso trate o erro de tal forma que interrompa a execução do processo que tentou enviar.

Saída do programa:

Cada processo deve individualmente produzir sua própria saída. Use a seguinte sintaxe para formatar a saída:

- Evento local: **i [c,c,c,c,...] L**, onde *i* é o ID do nodo local e *[c,c,c,c,...]* é o valor do relógio vetorial local;
- Envio de mensagem: **i [c,c,c,c,...] S d**, onde *i* é o ID do nodo local, *[c,c,c,c,...]* é o valor do relógio vetorial enviado e *d* é o ID do nodo destinatário da mensagem;
- Recebimento de mensagem: **i [c,c,c,c,...] R s t**, onde *i* é o ID do nodo local, *[c,c,c,c,...]* é o valor do relógio vetorial depois do recebimento da mensagem, *s* é ID do nodo remetente da mensagem e *t* é o valor do relógio lógico recebido com a mensagem.

Entrega e apresentação:

O trabalho deve ser realizado em duplas ou trios e apresentado em laboratório em um ambiente distribuído. Qualquer linguagem de programação poderá ser utilizada para o desenvolvimento. Para a entrega, é esperado que apenas um dos integrantes envie pelo Moodle um arquivo *.tar.gz*, contendo o código fonte da implementação.