Trabalho Final – Implementação de uma biblioteca para comunicação confiável e algoritmo distribuído

INE5418 – Computação Distribuída – UFSC
Prof. Odorico M. Mendizabal

1 Definição do trabalho

Você deverá desenvolver uma biblioteca de comunicação capaz de garantir entregas de mensagens respeitando critérios de ordem causal e total. Dessa forma, programas que usem a biblioteca irão usufruir dessas garantias no envio e recebimento de mensagens.

A biblioteca deve disponibilizar para o usuário as primitivas $send(id, m)$ e $receive(m)$ para mensagens destinadas a um destinatário (comunicação $1:1$), onde id é o identificador do destinatário e m é uma mensagem; e primitivas $broadcast(m)$ e $deliver(m)$ para mensagens destinadas a todos os participantes (comunicação $1:n$), sendo m uma mensagem e n o número total de participantes.

As mensagens do tipo $1:1$ deverá respeitar a ordem de entrega causal e mensagens $1:n$ respeitarão a ordem de entrega total.

Por razão de simplificação, você pode considerar que a mensagem m é composta por um *array* de bytes. A serialização/deserialização das mensagens, considerando que o usuário queira trocar dados de tipos diferentes de *byte array* fica por conta do usuário (a biblioteca não precisa disponibilizar o envio que mensagens codificadas em outros tipos).

As garantias de ordem devem ser respeitadas para um conjunto de processos participantes na comunicação. Observe que a ordem causal implementa, internamente, um vetor de relógios lógicos de n posições, onde n representa o número de processos. Para a ordenação total, também são garantidas a entrega de mensagens ordenadas para n processos participantes da comunicação. Para isso, a biblioteca levará em consideração um grupo de processos comunicantes definidos antecipadamente. Por exemplo, as informações sobre os n processos podem ser definidas em arquivos de configuração ou informadas pela linha de comando na inicialização dos processos. Você pode assumir que o número de processos participantes não muda durante a execução do programa.

A Figura 1 ilustra uma possível identificação dos processos em um arquivo de configuração.

```
//config_nodo0          //config_nodo1          //config_nodo2
processos = 3           processos = 3           processos = 3
id = 0                  id = 1                  id = 2
0 = localhost:6000      0 = localhost:6000      0 = localhost:6000
1 = localhost:6001      1 = localhost:6001      1 = localhost:6001
2 = localhost:6002      2 = localhost:6002      2 = localhost:6002
```

Figura 1: Exemplo de arquivos de configuração com 3 processos participantes. Cada coluna ilustra o arquico usado por cada um dos 3 processos.

1.1 Algoritmo distribuído

Para validar a sua biblioteca, você deverá utilizá-la para implementar algum algoritmo distribuído clássico da literatura. Por exemplo, você pode implementar algoritmos de exclusão mútua, eleição de líder, detecção de *deadlock*, detecção de término, etc.. A seguir seguem algumas sugestões de algoritmos, mas você pode escolher outros que não estejam listados.

- Ricart-Agrawala (exclusão mútua);
- Quorum-based (exclusão mútua);
- Maekawa (exclusão mútua);
- Bully (exclusão mútua);
- Chang e Roberts (eleição de líder);
- Mitchell e Merritt (detecção de deadlock);

2 Grupos e Entrega

O trabalho poderá ser realizado **em trios**. Cada grupo deve entregar o código fonte e um breve relatório com (i) instruções para executar o programa, (ii) explicação sobre as estratégias adotadas para implementar as primitivas de comunicação, (iii) explicação do algoritmo escolhido e (iv) uma análise da execução do programa considerando diferentes parâmetros. Você pode simular atrasos na entrega de mensagens (uso de `sleep` no recebimento de algumas mensagens) para ilustrar situações de interesse diversas.