Universidade Federal da Grande Dourados
Faculdade de Ciências Exatas e Tecnologias
Bacharelado em Engenharia de Computação
Tópicos Avançados em Computação III
Trabalho - Controle de Tráfego Aéreo

Prof. M.Sc. Rodrigo Porfírio da Silva Sacchi

1 Introdução

Neste trabalho propõe-se um sistema para sincronização e gerenciamento de fluxo de tráfego aéreo que combina técnidas de computação distribuída, com algoritmos de relógios lógicos de Lamport e a comunicação entre processos, a fim de prever congestionamentos e racionalizar os recursos dos aeroportos.

2 Descrição do Trabalho

O trabalho consiste na implementação de uma aplicação `OpenMPI` para auxiliar no "controle de tráfego aéreo" de um conjunto n de aeroportos. Neste trabalho desconsideraremos informações meteorológicas e voos com prioridades, ou seja, todos os voos, para nós, terão a mesma prioridade, ou quase. O que diferenciará cada voo é tempo de decolagem e de aterrisagem nos aeroportos.

Para o tempo (pousos e decolagens), devemos considerar o algoritmo de relógios lógicos de Lamport, visto em sala de aula. Algoritmo:

1. Antes da execução de um evento (interno, envio ou recebimento):

   1.1 P_i executa: $lc_i = lc_i + d$ $(d > 0)$. Aqui, d não necessariamente é igual a 1;

2. No envio de uma mensagem m por um processo P_i, aplica-se a regra 1 e P_i envia m a um processo P_j, m levando de carona o valor $m.lc = lc_i$;

3. Quando um processo P_j recebe uma mensagem m com $m.lc$, P_j executa as seguintes ações:

   3.1 $lc_j = \max(m.lc, lc_j)$;

   3.2 Aplica a regra 1;

   3.3 Entrega a mensagem.

Além disso, dois pousos, ou duas decolagens ou um pouso e uma decolagem não devem ocorrer ao mesmo tempo em uma pista do aeroporto. Neste sentido, você deve priorizar da seguinte forma, obedecendo o algoritmo de relógios lógicos de Lamport:

1. Se são dois pousos ou duas decolagens, priorize aquele com maior tempo de voo, ou seja, aquele que está voando por mais tempo;

2. Se for um pouso e uma decolagem, priorize o pouso. O avião que está decolando pode esperar um pouco.

Considerando que podemos ter outros aeroportos, por exemplo, de códigos 2, 3 e 4, e a partir deste tempo, no trabalho também será necessário atualizar dados de pousos em todos os aeroportos, considerando relógios lógicos. Todos os aeroportos precisam trocar mensagens para que informem seus pousos e decolagens, e que divulguem às pessoas no saguão do aeroporto. Para que seja possível, cada aeroporto `origem` deve enviar mensagens para aeroportos `destinos`, para que esses atualizem os dados e apresentem aos passageiros.

Vamos ilustrar o que cada aeroporto deve apresentar na tela: cada aeroporto, em um instante do tempo t, contém informações sobre: o **código do aeroporto**, **número de pousos** e de **decolagens**, e seus voos. Para cada voo (pouso ou decolagem), tem-se o **código do voo**, **origem**, **horários de chegada/-partida** e o **tempo de voo**. Para os horários de pouso e decolagem, sempre serão utilizados tempos representados com um valor `int`. Por exemplo, em um instante t, tem-se a seguinte configuração:

Tabela 1: Exemplo de uma configuração para o aeroporto a_1. Aqui não estamos considerando se há pousos e/ou decolagens conflitantes.

Código:	1		
Pousos:	3	Decolagens:	4
Pousos	Origem	Horário de Chegada	Tempo de voo
21	2	2	2
35	3	7	6
23	2	5	4
Decolagens	Destino	Horário de Partida	Tempo de Voo
11	2	0	3
12	3	1	4
13	2	3	1
14	4	4	2

Para formar o **código de um voo**, combine o número contido no código do aeroporto com um inteiro, iniciando de 1 e incrementando-o. Por exemplo, na Tabela 1, a primeira decolagem do aeroporto a_1 é o voo de código 11, onde o primeiro 1 vem do código do aeroporto e o segundo significa que é o primeiro voo do dia decolando daquele aeroporto.

3 Detalhes de Implementação

Para que o controlador de voo do aeroporto j tenha conhecimento sobre todas as chegadas, cada um dos i-ésimos outros controladores de voos deverão enviar mensagens para o aeroporto j, contendo informações sobre destino, horário de saída de j e o tempo de voo. Para comunicação entre aeroportos (diferentes compudadores/processos), deverão usar a biblioteca de troca de mensagens `OpenMPI`[1].

Para que os processos (aeroportos) troquem mensagens entre eles, será necessária uma biblioteca de troca de mensagens. Neste trabalho deve ser usada a biblioteca `OpenMPI`.

Para uma melhor interface para exibição na tela, a sugestão pe utilizar a biblioteca `ncurses`, no `Linux`. Nem sempre ela está previamente instalada:

- https://howtoinstall.co/pt/ncurses-base

Para entender como pode utilizar a ncurses, veja:

- https://terminalroot.com.br/ncurses/

Data da entrega do trabalho: 14/08/2023. Para o desenvolvimento deste trabalho serão permitidos grupos de **dois alunos** ou de forma **individual**. A forma de entrega será através do *Google ClassRoom*.

[1] https://www.open-mpi.org/