SISTEMAS OPERATIVOS

FACULDADE DE CIÊNCIAS E TECNOLOGIAS DA UNIVERSIDADE DE COIMBRA

Aeroporto

Bruno Ricardo Leitão Faria - 2018295474
Diogo Alves Almeida Flórido - 2018282583

1. Contextualização

"O sistema a ser desenvolvido deve simular um ambiente onde existem uma *Torre de Controlo*, duas pistas de aterragem, duas pistas para descolagem e voos a aterrar e a descolar. A torre tem uma posição fixa e controla os voos que se dirigem para aterrar no aeroporto e aqueles que aguardam uma oportunidade para descolar. A Figura 1 representa uma visualização do aeroporto, cujo problema subjacente é descrito abaixo.

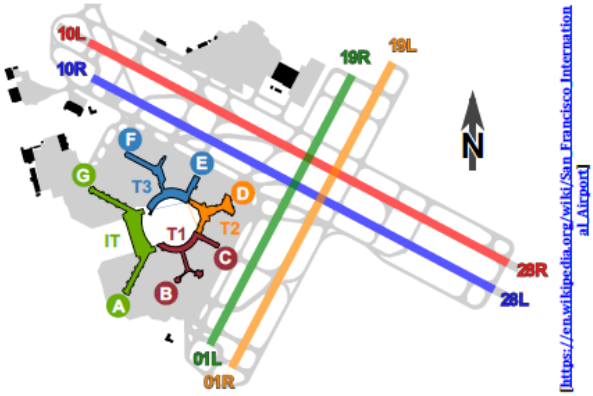
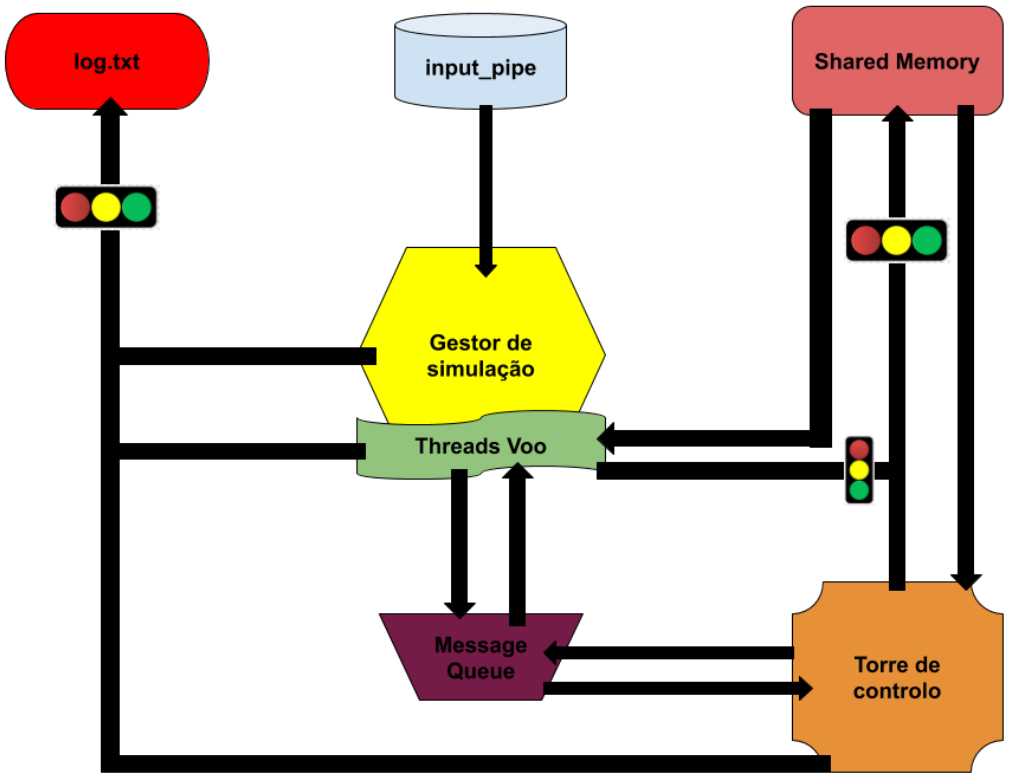


Figure 1 – Localização das pistas do aeroporto e *Torre de Controlo* (T2).

As pistas 01L e 01R são utilizadas para decolagens enquanto que as pistas 28L e 28R são usadas para aterragens. Devido às particularidades do aeroporto em questão, aterragens e descolagens não podem acontecer em simultâneo, enquanto que podem acontecer até 2 aterragens ou 2 descolagens em simultâneo. Na mesma pista, aterragens e descolagens têm de respeitar um tempo mínimo entre voos. Desta forma, é necessária uma *Torre de Controlo* particularmente eficiente capaz de garantir um alto *throughput* ao mesmo tempo que evita *starvation* dos voos.
Os voos chegam ao sistema com o objetivo de descolar ou aterrar, existindo uma fila de espera para cada um dos casos. Cada voo que deseja descolar é caracterizado pela sua hora desejada de partida. Cada voo com o objetivo de aterrar é caracterizado pelo tempo que demora até à pista (*Estimated Time of Arrival - ETA*) e pelo seu combustível disponível. Os aviões podem aterrar até ao seu combustível chegar a `0`. A partir desse momento apenas têm combustível de reserva e serão automaticamente desviados para outro aeroporto. O tempo é dado em *time units* e o combustível em *fuel units*, sendo que os aviões gastam uma unidade de combustível por cada unidade de tempo em voo.
A *Torre de Controlo* coordena a operação dos voos com o objetivo de maximizar o número de operações e minimizar os tempos de espera. Quando um voo se aproxima do aeroporto, a torre deve decidir se há condições para ele aterrar, caso contrário deve ordenar ao voo que circule no ar enquanto espera (*holding*). No entanto, há que respeitar o combustível disponível, sendo que apenas na extrema impossibilidade de o voo aterrar este deve ser desviado para um outro aeroporto. Durante o processo de descolagem ou aterragem, a pista em uso fica indisponível."

2. Diagrama do Sistema Desenvolvido



3. Explicação

a. Gestor de Simulação

Este é o processo que serve como "main". Nele são criados a Torre de Controlo, a Memória Partilhada, a Message Queue e as threads dos voos. O seu funcionamento baseia-se na verificação constante de novos comandos no input_pipe, na validação dos mesmos e na introdução deles numa lista ligada por ordem de tempo de início das threads de voo. Para se poder saber o tempo decorrido e iniciar threads de voo no instante certo, o Gestor de Simulação cria também outra thread (chamada ftimer). Optámos por uma thread, pois consideramos este o meio mais fácil de correr código em simultâneo com o processo acessando a diversas variáveis em comum. O Gestor de Simulação escreve no log.txt quaisquer comandos considerados inválidos.

b. Shared Memory

A Shared Memory é uma estrutura de dados que será tanto acedida pela Torre de Controlo como pelas threads de voo. Nela estão presentes os dados referentes às estatísticas, uma array dinâmica de strings com instruções para cada voo ativo e outra array dinâmica com ponteiros para unnamed semaphores que irão indicar a cada voo quando a sua instrução foi mudada. Optámos por uma array dinâmica de strings pois pareceu-nos a maneira mais fácil de diferenciar instruções mas também partilhar os novos ETA nos casos das manobras de holding. Criámos a array de unnamed semaphores para que as threads de voo não precisassem de estar sempre a aceder à memória partilhada. Alterações nos dados da Shared Memory são sincronizadas através de um named semaphore.

c. Message Queue

A Message Queue é utilizada como meio de comunicação entre as threads de voo e a Torre de Controlo. Após a criação de cada voo, ela recebe uma estrutura com uma struct para cada tipo diferente de voo para além de um long mtype com valor 1 ou 2 conforme o tipo de voo. A struct do tipo do voo criado é preenchida com os inteiros necessários (início da descolagem ou ETA e combustível) enquanto que a outra struct ficará vazia. Após a receção da estrutura por parte da Torre de Controlo, a Message Queue também ficará encarregue de receber de volta uma estrutura com mtype 3 e um inteiro referente ao índex da array dinâmica presente na Shared Memory onde vão estar as instruções dadas a esse voo.

d. Threads de Voo

Após criadas, cada thread de voo fica encarregue de receber da Message Queue o index da instrução e do unnamed semaphore presentes nas arrays dinâmicas da Shared Memory que lhe correspondem. Após isso ficam apenas à espera que o unnamed semaphore correspondente incremente para obter uma nova instrução e ajustar o seu comportamento à mesma. Cada Thread escreve o que lhe foi ordenado no log.txt e atualiza as estatísticas na Shared Memory.

e. Torre de Controlo

Este processo está encarregue de receber mensagens por parte da Message Queue com a indicação de novos voos, de os ordenar e de lhes enviar instruções através da Shared Memory. Para isso decidimos criar duas threads: A primeira ordena os voos em duas listas ligadas destinadas para cada tipo de voo baseada no tempo de início da descolagem para o caso das descolagens e no ETA, na diferença entre combustível e ETA e no fator emergência para o caso das aterragens. A segunda thread decide com que voos as pistas deverão ser ocupadas dando prioridade às aterragens nos casos em que os tempos coincidem ou quando a aterragem é de emergência e não daria tempo para uma descolagem terminar antes da sua chegada. Controlamos a disponibilidade das pistas tal como a alternação entre os tipos de voo através de dois named semaphores que irão ser incrementadas e decrementadas tanto pela Torre como pelas threads de voo.

f. Log.txt

Este é um ficheiro que se mantém aberto ao longo da execução e onde várias fontes diferentes vão escrever. As suas escritas são sincronizadas através de um named semaphore de forma a não existirem conflitos.

g. Signals

No Gestor de Simulação e na Torre de Controlo criámos uma resposta ao sinal SIGINT e outra ao sinal SIGUSR1. Após acionado o sinal SIGINT, os processos esperam até as array dinâmicas em memória partilhada ficarem com os elementos todos a NULL, ou seja, até as threads terminarem. De seguida a Torre de Controlo termina e o Gestor de Simulação espera, fecha o input_pipe e o log.txt e limpa os semaphores, Shared Memory e Message Queue até terminar a sua execução.

Tempo total despendido no projeto: ±50 horas;
Sendo que na maior parte do tempo, os dois membros trabalharam em simultâneo.

Logo o tempo despendido por cada um deve se aproximar do tempo total.