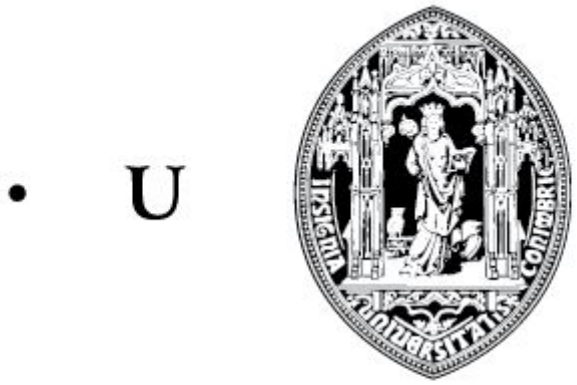
Relatório de Sistemas Operativos

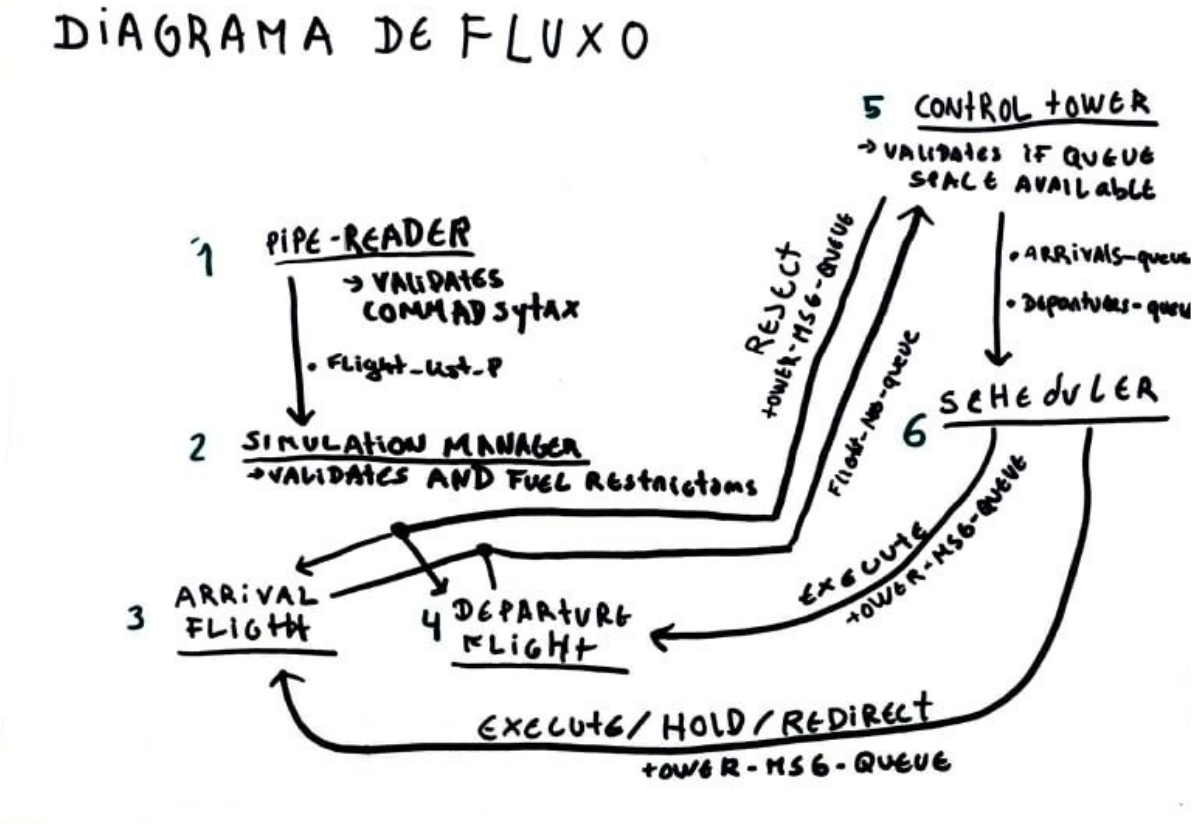


UNIVERSIDADE DE COIMBRA

2019/2020

Meta 2

Alexandre Cortez Santos - 2018277730
Diogo Ferreira Sobreira - 2018294861



Threads e Processos

1. Pipe Reader - Lê e valida comandos da named pipe. Converte comandos em voos e adiciona os mesmos à flight list
2. Simulation Manager - Lê voos da flight list e valida-os. Os voos válidos são convertidos em arrival ou departure threads
3. Arrival Flight - Envia informação sobre o seu voo para a control tower e aguarda instruções
4. Departure Flight - Envia informação sobre o seu voo para a control tower e aguarda instruções
5. Control Tower - Recebe voos das threads de voo. Adiciona estes às suas queues, e no caso de não haver espaço nas queues rejeita os mesmos.
6. Scheduler - Gere queues e envia instruções as threads de voo.

Fluxo

Inicialmente, o pipe reader lê comandos do named pipe e valida a sua syntax. Converte a informação dos comandos para voos (flight_t) e adiciona-os à lista ligada de voos, flight_list_p (flight_list_t*).

O Simulation Manager simula o tempo da simulação, e em cada intervalo de tempo, procura voos na lista com init igual ao instante de tempo atual. Estes voos, por sua vez, são convertidos em threads de voo (Arrival / Departure) com um nivel de prioridade (NORMAL_FLIGHT, PRIORITY_FLIGHT).

As threads de voo, enviam pela flight_message_queue informação sobre o seu voo para a control tower, e aguardam instruções da mesma.
A control tower recebe os voos e valida se podem ser inseridos nas suas queues. Se impossível, envia instrução de REJECT para a thread de voo correspondente usando a tower_message_queue.

O scheduler é responsável por autorizar aterragens, descolagens, holdings e redirects e enviar estas instruções pela tower_message_queue para as threads de voo.

As threads de voo ao receberem instruções provenientes da tower_message_queue com message type igual ao seu id de voo e executam a instrução indicada.

Estruturas de dados

Lista voos - flight_list_t

Lista ligada simples para guardar voos

Queue - queue_t

Fila ordenada por t_start e t_fuel

Nó queue - queue_node_t

Estrutura com informação sobre quando:

- Voo chega ao aeroporto (t_start)
- Voo finaliza aterragem (t_end), se iniciar ação em t_start
- Voo fica sem combustível (t_fuel)

No caso de arrivals, t_start = init + eta, t_end = t_start + L + dl, t_fuel = init + fuel
No caso de departures, t_start = init + takeoff, t_end = t_start + T + dt, t_fuel = 0 (irrelevante porque, departures não têm fuel)

Pistas - shm_lanes_t

Estrutura que contêm informação sobre quando as pistas vão estar disponíveis (t_end_1, t_end_2) e que tipo de voo as está a usar (type - DEPARTURE / ARRIVAL)

Algoritmo de escalonamento

1. Em cada instante de tempo, vê se o primeiro voo da arrival queue chegou ao aeroporto
2. Verifica se uma das pistas está livre e se as pistas estão a ser usadas para aterragens. Se livre, atualiza o t_end dessa pista com o seu t_end, e o type das pistas para ARRIVAL e faz pop da arrivals queue. Se não, não o insere na pista
3. Faz o mesmo para o próximo voo no topo da arrivals queue
4. Verifica se uma das pistas está livre e se as pistas estão a ser usadas para descolagens. Se livre, atualiza t_end dessa pista para current_time + t_end - t_start, e o type das pistas para DEPARTURES e faz pop da departures queue. Se não, não o insere na pista
5. Tenta fazer holding de voos que chegaram ao aeroporto, se não for possível, faz redirect do voo

Considerações

- Considerámos que o fuel tem que durar durante o eta / takeoff + L / T + dl / dt
- Apenas usámos duas pistas e ignorámos que é possível aterrar durante o dt ou levantar voo durante dl