Universidade de Coimbra
Faculdade de Ciências e Tecnologias – Engenharia Informática
2021-2022

Arquitetura e mecanismos de sincronização

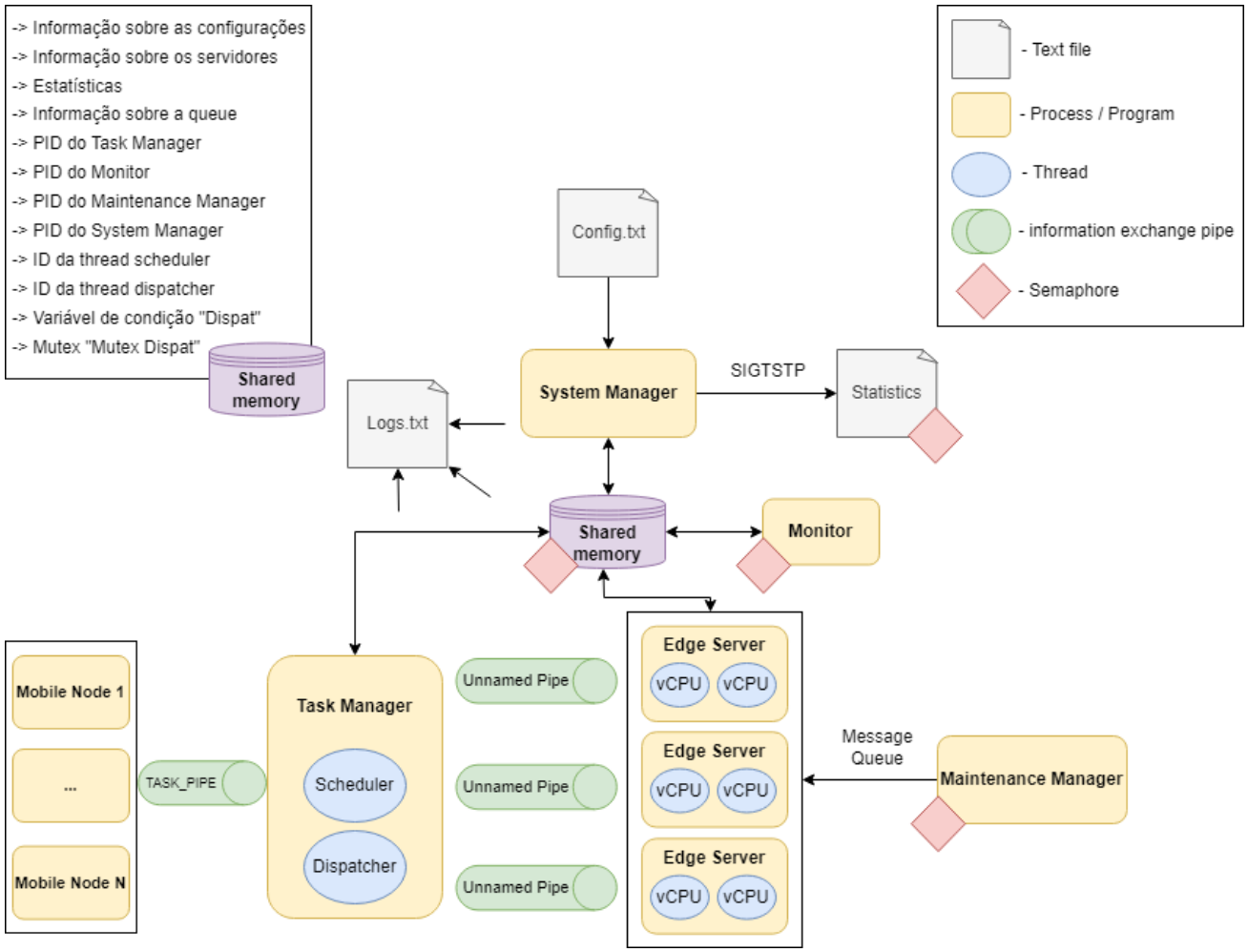

Figura 1 – Esquema representativo da arquitetura escolhida para o simulador

Mecanismos de sincronização e variáveis de condição:

Acima encontra-se a arquitetura do nosso projeto de Sistemas Operativos, já com mecanismos de sincronização implementados. Temos presentes seis semáforos:

- **Semáforo na memória partilhada** com o objetivo de limitar o acesso à mesma a um processo/thread de cada vez, de forma a otimizar o desempenho do programa aquando da sua execução.

- **Semáforo na escrita nos ficheiros de logs** com o intuito de limitar a quantidade de processos que escrevem neste ficheiro, otimizando assim o programa e assegurando que os dados são escritos na ordem correta.

Pedro Afonso Ferreira Lopes Martins (2019216826) João Carlos Borges Silva (2019216753)

• **Semáforo START:** maintanance manager, monitor, thread scheduler e dispatcher não começarem antes do servidor (também usado no processo task manager e a thread scheduler para sincronizar o uso da última).

• **Semáforo MONITOR:** Tem como objetivo bloquear o processo monitor, de forma a poder libertá-lo para trocar o estado do simulador.

• **Semáforo MAINTENANCE:** Serve para trocar o estado de manutenção do servidor, ficando à espera de um sinal para acabar a manutenção e assim resumir o progresso.

• **Semáforo SERVER_MAINTENANCE:** Serve para o servidor que está prestes a entrar em manutenção esperar pelos término das tarefas por parte dos vCPUs.

• **Semáforo FINISHING:** Para ficar à espera que todos os processos e threads criados acabem o seu trabalho, de forma a poder encerrar corretamente todos os recursos alocados. É usado no método responsável pela limpeza dos dados.

Também temos presentes duas variáveis de condição, **sched** é responsável por controlar e gerir a fila de tarefas que chegam do mobile_node (descartar as tarefas cujo prazo de execução tenha expirado e ordenar a fila pelo seu nível de prioridade) e a **dispat,** que é responsável por ativar a thread dispatcher, consoante necessário (quando há tarefas por executar ou vCPUs ficam livres, encontrando-se dentro da memória partilhada).

Por fim, temos duas variáveis incluídas na memória partilhada, **f** (condição para garantir o término das várias threads) e **monitor_high** (indicador do estado do servidor, representado por 1 se o servidor estiver em modo "high performance" e 0 em caso contrário) e um **mutex** para cada **vCPU**, com o propósito de o desbloquear quando a thread **dispatcher** escolher uma tarefa para ser executada por ele.

Divisão de tarefas:

	João Silva	**Pedro Martins**
Elaboração do relatório		✓
Desenvolvimento do Mobile Node		✓
Desenvolvimento dos métodos de funcionamento do simulador	✓	
Fim controlado do servidor	✓	
Elaboração e desenvolvimento da Arquitetura	✓	✓
Desenvolvimento da thread scheduler e dispatcher	✓	✓
Tempo dispendido	**80h**	**60h**