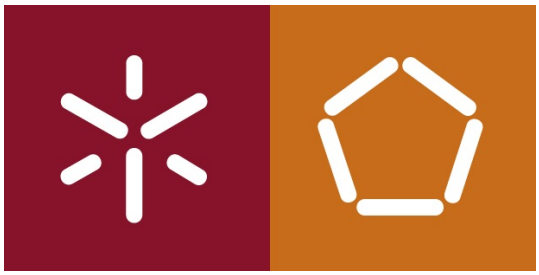

Universidade do Minho
Licenciatura em Engenharia Informática

Sistemas Operativos - Trabalho Prático

Grupo 22

Tiago Guedes (a97369)
Tiago Carneiro (a93207)
Gonçalo Moreira (a73591)

3 de maio de 2024

Índice

1 Introdução

No âmbito da Unidade Curricular de Sistemas Operativos desenvolvemos um projeto que visa a implementação de um serviço de execução e escalonamento de tarefas num computador. Para tal, foi necessário o desenvolvimento de um servidor denominado (programa **Orchestrator**) e um cliente (programa **Client**).

O cliente (programa **Client**) será usado como interface para que vários utilizadores possam requisitar certas tarefas ao servidor. Este servidor (programa **Orchestrator**) irá estabelecer contacto a pedido de cada um dos clientes que o contacta, executando as tarefas requisitadas por cada um.

Os utilizadores podem usar o cliente para pedir ao servidor que execute uma definida tarefa com uma determinada duração, assim como, requisitar uma consulta das atuais tarefas em execução e já terminadas.

É permitido os pedidos de vários clientes em simultâneo, logo, existe uma camada de controlo de concorrência que irá manter os pedidos sucessivos em fila de espera para serem processados e executados pelo servidor, segundo o nível de importância de cada pedido.

2 Comunicação

A comunicação entre o servidor e os clientes representa a parte fundamental deste projeto, já que sem isso não é possível realizar as funcionalidades requeridas. Para assegurar que cada cliente lê apenas a mensagem destinada a si, será necessário a criação de canais de comunicação separados para cada cliente.

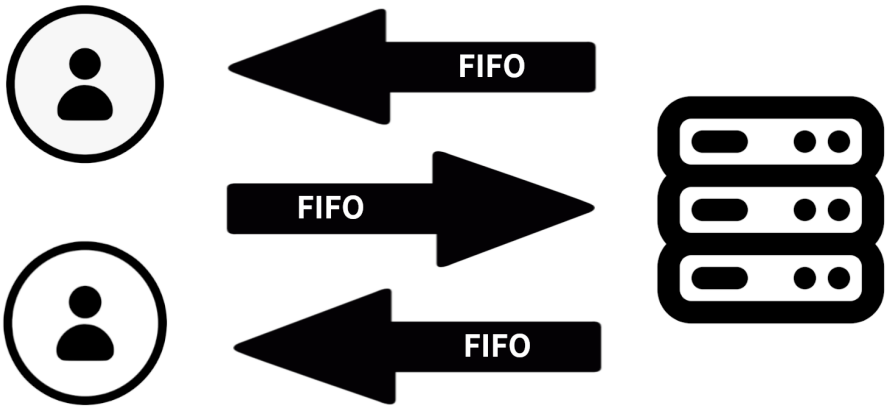

Figura 1: FIFOs Separados

2.1 Orchestrator

Mal seja iniciado o programa **Orchestrator**, com os devidos parâmetros (output folder, parallel-tasks limit), este cria um **FIFO** dedicado (server_fifo) para o qual será usado como pipe de leitura do pid do **Client** em contacto.

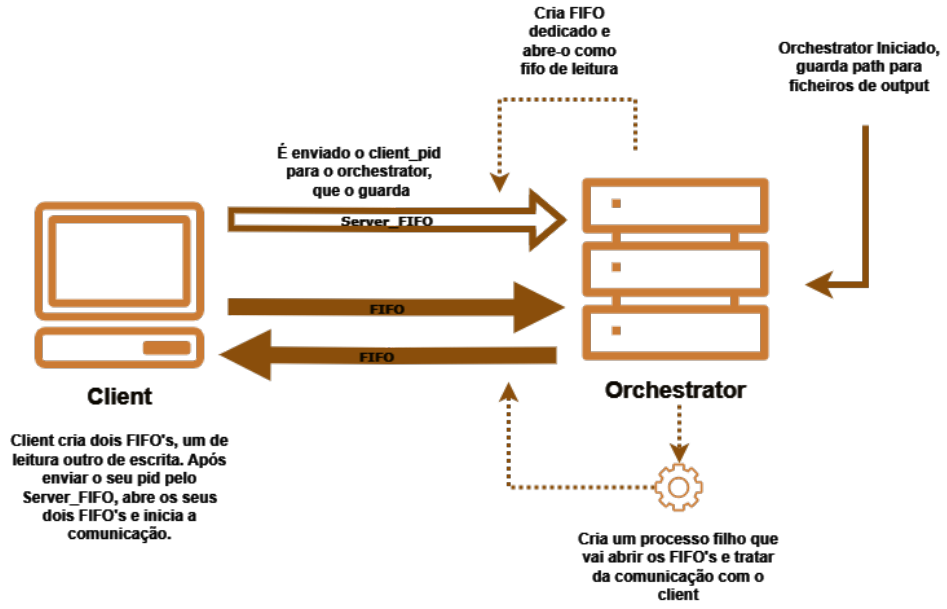

Figura 2: Estabelecimento de Conexão

2.2 Client

O cliente será responsável por criar dois pipes com nome (**FIFOs**). Um para comunicar os comandos a executar pelo servidor (pipe escrita) e o outro para receber o resultado gerado pelo servidor após a execução dos comandos providenciados (pipe leitura). Ambos irão ter associados aos seus nomes o pid do client relativo.

Para assegurar que o programa **Orchestrator** sabe o cliente para o qual está a falar, após iniciar os pipes de leitura e escrita, o cliente enviará o seu pid através de um **FIFO** dedicado ao servidor (descritor de **FIFOs**), para que este guarde essa informação e consiga conectar aos pipes do client. Após terminar a comunicação entres os dois, o cliente eliminará os pipes criados por este, garantindo espaço para outros clientes criarem os seus **FIFO's**.

3 Arquitetura de processos

Para o rápido e eficiente processamento e execução das tarefas a serem executadas pelo servidor, será necessário o uso de diversos processos.

Este uso de vários processos, é feito pelo servidor e permite uma maior eficiência na execução das várias tarefas dedicadas a este servidor. Todas as vezes que um cliente inicia um pedido ao servidor, este passa sempre por um método inicial de ligação entre o **Client** e o **Orchestrator**.

Logo após finalizar essa ligação é criado um processo filho que faz o handle da informação enviada pelos pipes do cliente. Este processo filho, também está incumbido de verificar a disponibilidade da queue de execução.

Dependendo do tipo de pedido por parte do **Client** (execute único, encadeado ou um status), serão criados os devidos processos para a rápida execução de cada um dos comandos especificados e que vão ter um *task_number* associado para permitir uma fácil identificação das tarefas e acesso à informação presente nas queue's.

No caso de um pedido de um execute encadeado, foi necessário a criação de vários processos, um para cada programa a executar e uma ligação de pipes entre eles que permite a transferência de informação de um processo para o outro, para que o seguinte possa executar o seu programa com os dados do output do anterior.

3.1 Execução Única

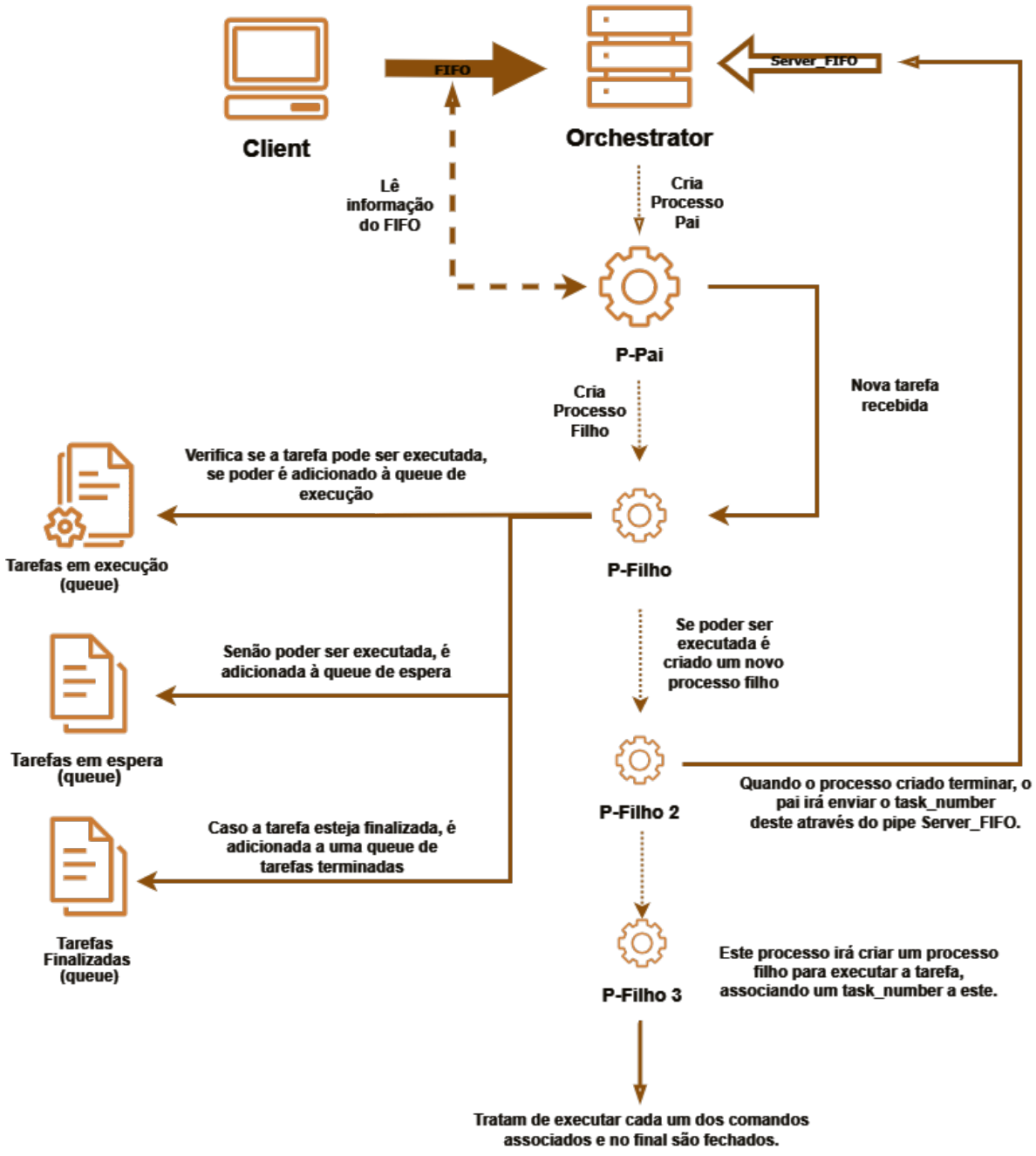

Figura 3: Diagrama de Execute Único

3.2 Execução Encadeada

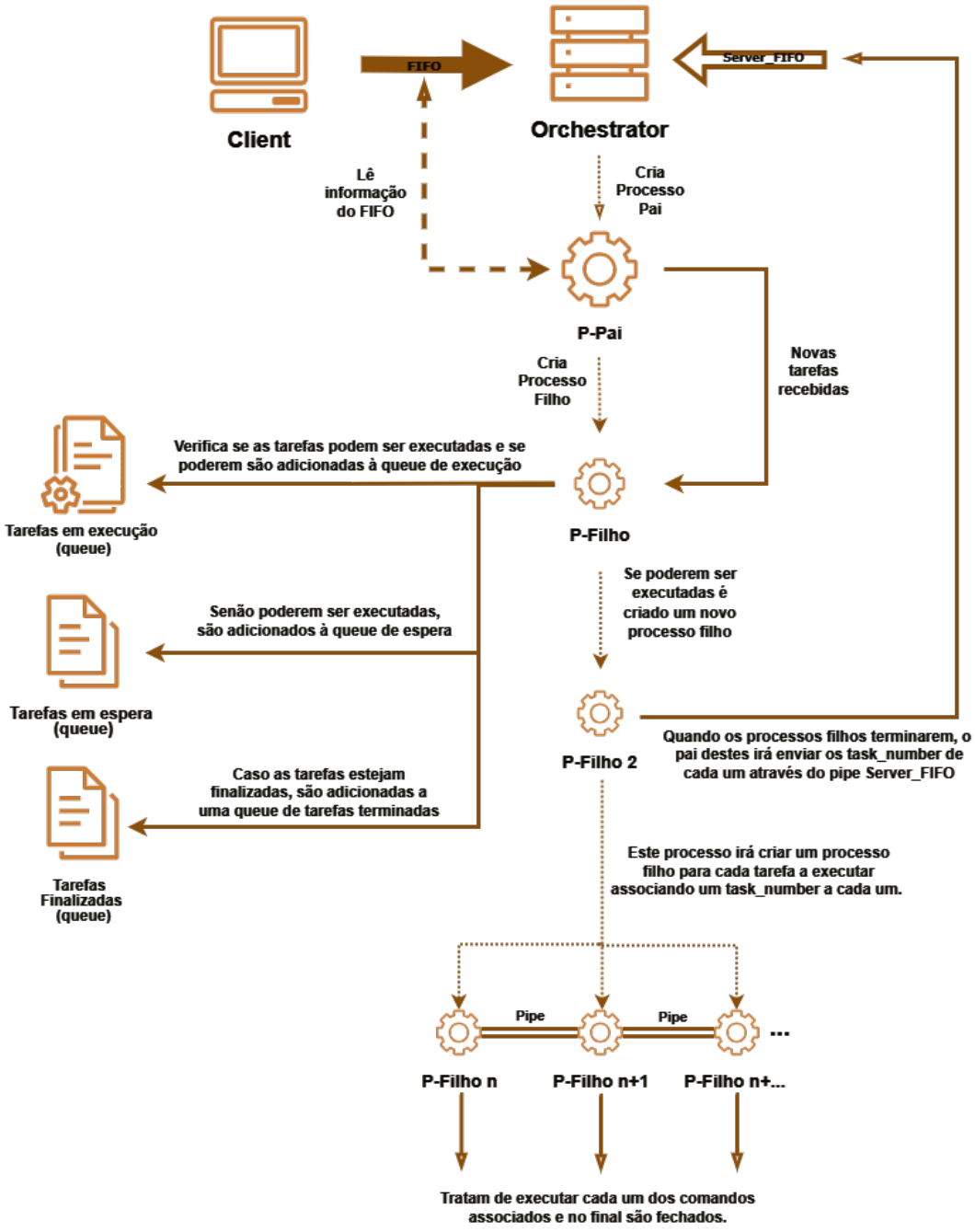

Figura 4: Diagrama de Execute Encadeado

3.3 Consulta Status

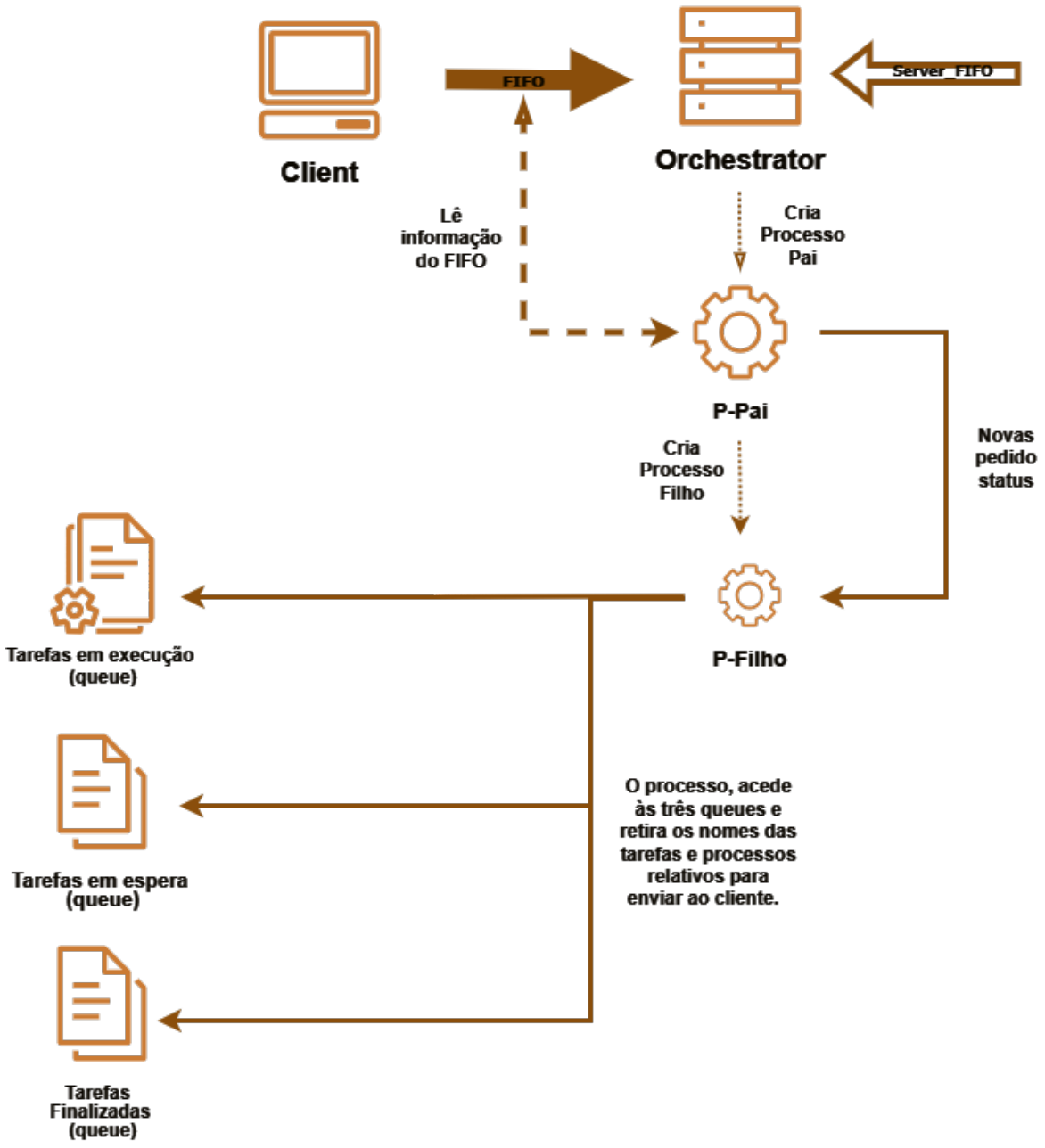

Figura 5: Pedido Status

4 Estruturas

Surgindo a necessidade de manter a informação relativa a cada pedido de cada cliente, foi necessário a criação de duas estruturas de listas ligadas, que permitirão manter estes dados guardados de uma forma eficiente.

Estas listas permitem guardar as informações relativas ao pedido do cliente, neste caso, a estrutura **LlCommand** permite manter todos os comandos (dependo se é execução única , encadeada ou um status) e seus respetivos argumentos.

A estrutura **LinkedListProcess** guarda as informações externas do pedido, neste caso o pid do cliente que o fez, a prioridade de execução, uma lista **LlCommand** com os comandos e argumentos já guardados e o número total destes comandos. Não relativos ao cliente mas sim à funcionalidade do processo existe ainda o pid do processo filho criado para gerir este pedido e o nome do ficheiro de output dos comandos a executar.

Estas duas estruturas, mas principalmente a **LinkedListProcess**, irão ser usadas para definir as queues de execução, espera e pedidos completos.

5 Conclusão

Durante a realização deste trabalho prático enfrentámos vários desafios que nos obrigaram a pensar e refletir qual seria a melhor estratégia a adotar.

Desta forma, acreditamos que cumprimos com a maioria dos nossos objetivos, para além de termos implementado devidamente as funcionalidades básicas, a maioria das funcionalidades avançadas estão finalizadas e seguem o comportamento esperado.

É certo que podíamos ter feito algumas alterações que certamente melhorariam o projeto, porém por alguns constrangimentos ou mesmo por falta de visão da nossa parte não foi possível implementá-las.

Em suma, apesar de ter alguns objetivos não finalizados, chegamos à conclusão que o projeto final ficou em bom estado, com um resultado eficiente e de acordo com o esperado.