Trabalho Prático
Sistemas Operativos

Tiago Marques - 2022210638
Mariana Sousa - 2022215999

Licenciatura em Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra

May 13, 2024

1 Mobile User

O *Mobile User*, para gerar os pedidos de autorização para cada serviço, usa 3 threads, uma para cada serviço (redes sociais, vídeo e música), e outra thread para receber alertas através da Message Queue. Para sincronização, utilizamos um mutex para controlar a modificação do número máximo de solicitações, de modo que somente uma thread modifique este valor.

2 BackOffice

O *BackOffice* apenas recebe estatísticas periódicas através da Message Queue e envia comandos através do named pipe, para solicitar estatísticas. Para isto criamos uma thread dedicada exclusivamente à receção das estatísticas. Neste caso, quando o programa escreve no named pipe ou lê da Message Queue, é sincronizado internamente.

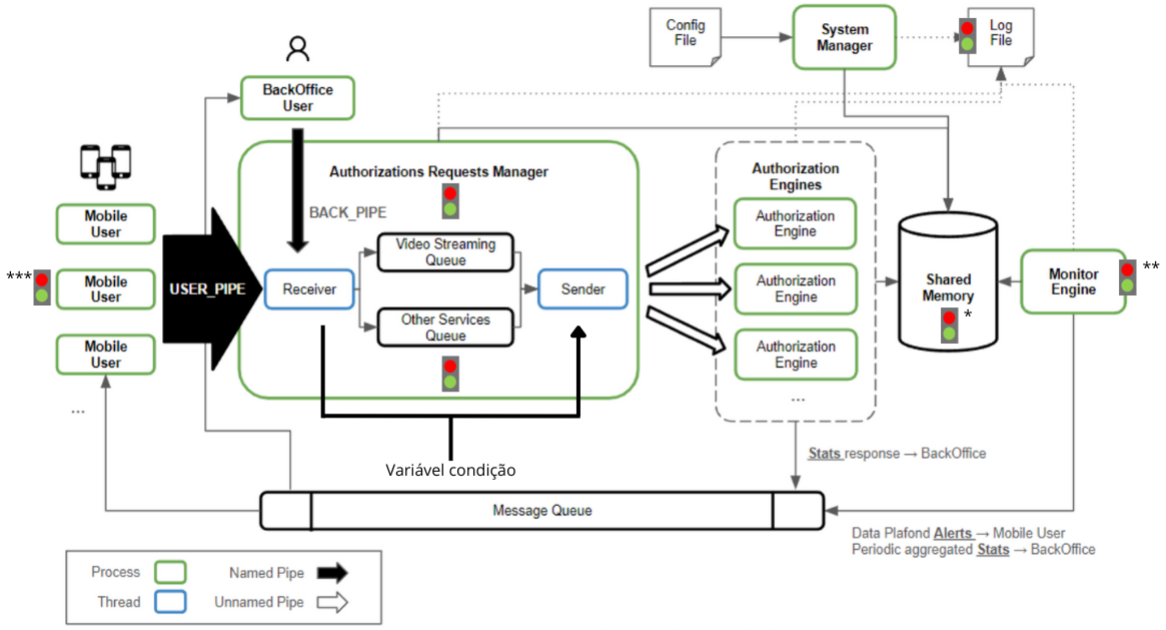
3 System Manager

Este cria a *Shared Memory* que contém um ponteiro para os dados dos users, um ponteiro para um array de ponteiros, que indica a disponibilidade dos *Authorization Engines*, uma estrutura com as estatísticas e uma variável running. Também cria a Message Queue e escreve o id desta num ficheiro. Usamos um semáforo para evitar concorrência na variável running.

- **Authorization Requests Manager** - Aqui é criado tanto os named como os unnamed pipes, as threads *Sender* e *Receiver* e as estruturas de dados internos. Para o acesso das filas, utilizamos um mutex, para não existir corrupção de dados, a adicionar,remover e verificar se está vazia.
- **Receiver** - Esta thread recebe como argumentos as estruturas de dados internos e vai receber os pedidos através dos *named pipes* e adiciona-os nas respetivas filas. Este contém uma variável de condição para enviar sinais para o *Sender*, para este saber quando são adicionadas mensagens às filas.
- **Sender** - Esta thread é responsável por ler os pedidos das filas e enviar para o *Authorization Engine* disponível. Cria também *Authorization Engine* extra e remove quando não é necessário. Sempre que necessário atualiza as estatísticas. Para isto são utilizados 3 semáforos um para cada componente da *Shared Memory*(`users, authorization_free e stats`)

- **Authorization Engine** - Responsável por executar os pedidos autorização, usa os mesmos semáforos que o *Sender*.
- **Monitor Engine** - Verifica o plafond de dados de cada *Mobile User*, e gera alertas para estes, também é responsável por enviar estatísticas periódicas para o *BackOffice*. Para isso, criamos 2 threads: uma para gerar alertas e outra para as estatísticas. Utilizamos um semáforo que funciona como sinal, inicializado a 0, para gerar os alertas.

Com este trabalho, foi despendido cerca de 50h por elemento. No fim, podemos afirmar que o objetivo foi concluído.



* 4 semáforos para a Shared Memory
** Semáforo inicializado a 0
*** Mutex dentro de cada Mobile User

Figura 2: Arquitetura técnica do simulador