FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Relatório Intermédio

Configurações iniciais

Saulo José Mendes
Rui Moniz
Sistemas Operativos
13 de maio de 2023

Índice

Introdução

Neste presente relatório, apresentamos a estrutura de uma aplicação que utiliza de diversos mecanismos de sincronização e comunicação entre processos para criar gerir diferentes sensores conectados na mesma rede (IoT). Em pormenores, foram implementadas todas as funções necessárias para o funcionamento das consolas dos utilizadores, além dos processos individuais de cada sensores e o *System Manager,* todos os mecanismos que permitem o funcionamento simultâneo de todos os processos.

Funcionamento

Os sensores funcionam como processos independentes que enviam informação para o **System Manager** através de uma named pipe **(SENSOR_PIPE),** no **System Manager** existe uma thread **(Sensor Reader)** que é responsável por ler os dados da named pipe e inseri-los na sua respetiva internal queue **(SensorQueue)** para posteriormente serem processados pela thread **Dispatcher**.

À semelhança dos dados enviados pelos sensores, os comandos também são enviados ao system manager pela consola do utilizador. Contudo, a thread responsável por recebê-los é a Console Reader, através da CONSOLE_PIPE e guarda-os noutra internal queue, a CommandQueue. Tanto nos sensores como nos comandos, a inserção dos dados nas internal queues são controlados por semáforos de mutexes de modo a garantir a queue não atinge o tamanho máximo.

Especificamente, utiliza-se um mutex para guardar os tamanhos individuais de cada internal queue, um semáforo de contagem para impedir a inserção de novos nós assim que se atinge o limite da internal queue, e um semáforo para verificar se há requests na fila interna (independente de ser uma informação do sensor ou um comando). Deste modo, evita-se que o Dispatcher fique em espera ativa, e permite á Sensor Reader descarta os pedidos quando a fila estiver cheio (try wait), enquanto que a Command Reader sempre enviará os seus pedidos (wait).

A thread **Dispatcher** retira os dados das internal queues sendo esta ação sincronizada por um semáforo com o objetivo de impedir o uso excessivo da máquina (busy wait). Posteriormente, os **Workers** recebem os dados através de duas unnamed pipes (uma para os sensores e outra para os comandos).

De modo a otimizar o sistema e a utilização da memória foram utilizadas quatro estruturas de memória partilhada: uma com a informação dos sensores, alertas, workers e uma estrutura que guarda tanto as configurações do sistema (lidas do ficheiro de configuração) como o número atual de sensores, alertas e chaves. O acesso a estas zonas de memória partilhada são sincronizados por quatro semáforos (um para cada memória partilhada).

Por fim, o Alerts Watcher recorre a um semáforo binário, inicializado a 0, para evitar a espera ativa. Em concreto, sempre que algum worker atualiza alguma informação acerca de qualquer sensor, incrementa-se o semáforo, que leva a execução de um ciclo completo do Alerts Watcher. Caso algum alerta seja acionado, encaminha-se um aviso à consola do utilizador que o criou, e também se guarda a informação no ficheiro log.

Tempo despendido por cada integrante

Rui Pedro Moniz - 30 horas

Saulo José Piccirilo Mendes - 50 horas