Computação Distribuída – Guião de Avaliação 01

Diogo Gomes & Mário Antunes

Março 2019

1 Introdução

O objectivo deste guião de avaliação é implementar a simulação do Guião 2 (ver Figura 1) em **Peer-to-peer (P2P)**.

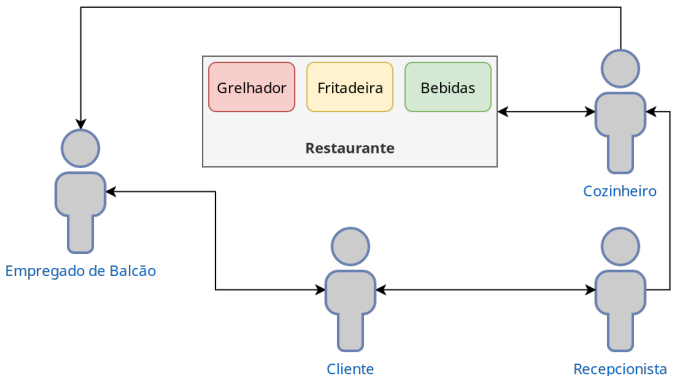

Figura 1: Esquema simplificado da simulação do Guião 2.

Como a implementação é **P2P** não existe uma entidade central para mediar a comunicação entre as entidades. Em vez disso as entidades deverão ser organizadas em anel (*token-ring* [1] mais detalhe em Seção 2), ou seja, cada entidade sabe quem é o seu sucessor e predecessor.

Como no Guião 2, a simulação do restaurante é composta por 5 entidades:

- Restaurante: entidade responsável pela gestão dos aparelhos de cozinha (fritadeira, grelhador e máquina de bebidas).

- Recepcionista: recebe os clientes e anota os pedidos
- Cozinheiro: recebe os pedidos e confecciona
- Empregado de balcão: entrega o pedido e recebe o pagamento
- Cliente: faz pedidos de menu periodicamente

A entidade Restaurante agora é responsável apenas pela gestão dos equipamentos da cozinha. Como antes a cozinha é composta por 3 aparelhos:

- Grelhador: onde o cozinheiro pode preparar o hambúrguer (tempo médio de preparação: 3 segundos)
- Bebidas: onde o cozinheiro pode preparar as bebidas (tempo médio de preparação: 1 segundo)
- Fritadeira: onde o cozinheiro pode preparar as batatas (tempo médio de preparação: 5 segundos)

Todas as acções demoram uma quantidade de tempo aleatório. Este quantidade de tempo segue uma distribuição Gaussiana [2,3] com a media 2 e desvio padrão 0.5 segundos.

Um pedido é constituído por pelo menos um item (hambúrguer, bebida, batata), mas pode ser qualquer combinação de itens até ao valor máximo de 5 itens. O cliente deve gerar os pedidos de forma aleatória.

Para avaliar a correcta execução do algoritmo é obrigatório a utilização de logs [4].

2 Criação do Anel e descoberta de entidades

Cada entidade vai receber um ID (ver Tabela 1). A ordem das entidades no anel é dado por um campo de **ID** (ver Figura 2). Cada nó do anel conhece apenas o seu sucessor, e o fluxo de mensagens no anel deve seguir essa direcção. As entidades devem organizar-se em anel de forma autónoma, através de mensagens "NODE_JOIN". Depois um um processo inicial de construção o anel deve ficar estável e pronto para comunicar. Podem assumir que não existe perca de mensagens (*localhost*) e que os nós não se desligam do anel (excepto em caso de erro na execução). Os nós podem ser ligado por qualquer ordem e devem formar um anel correcto.

Tabela 1: IDs e quantidade das entidades.

Entidade	Quantidade	IDs
Restaurante	1	0
Recepcionista	1	1
Cozinheiro	1	2
Empregado de balcão	1	3
Cliente	≥ 1	NA

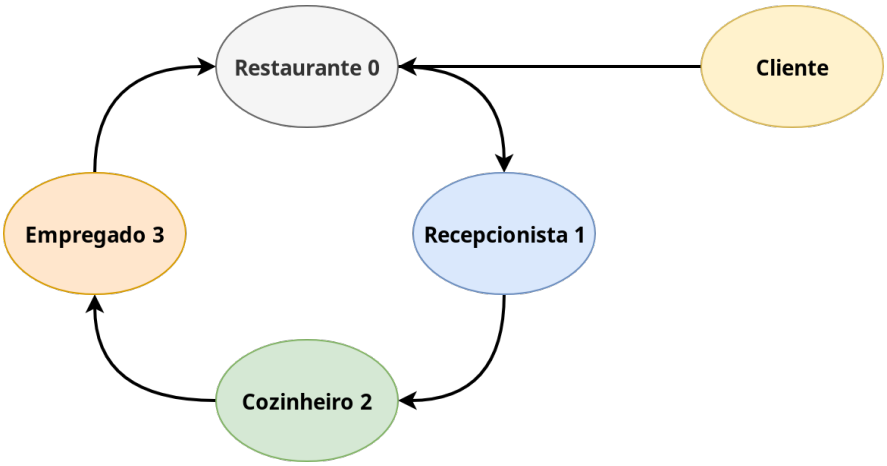

Figura 2: Anel de comunicação com as 5 entidades

Depois do processo de criação do anel, os nós devem proceder a descoberta das entidades. Ou seja, cada nó deve questionar o anel para mapear o ID dos nós em entidades, por exemplo ID 0 corresponde ao restaurante. Desta forma sempre que uma entidade necessite de comunicar com alguma entidade pode mapear essa entidade num ID do anel. As mensagens usadas para este efeito deverão ser "NODE_DISCOVERY" Depois do processo de descoberta a simulação pode começar normalmente.

O anel deve garantir a correcta comunicação. Se uma entidade receber uma mensagem que não é destinada a si mesma, deve propagar para o seu sucessor. Desta forma a mensagem será propagada pelo anel até a entidade destino.

Os cliente não fazem parte do anel (como visível na Figura 2).

3 Objectivos

- A codificação de mensagens deverá ser feita em pickle [5]; utilizando por base um dicionário do tipo: {"method":XXXXX, "args": XXXXX}.
- Comunicação através de UDP [6].
- Implementar uma rede em anel e descoberta de entidades.
- Implementar a comunicação das entidades na rede em anel.
- Implementar o ciclo de vida das entidades.
- Implementar a simulação com um cliente.
- Aumentar o número de clientes.

4 Notas

- É fornecido um código base que deve ser usado para implementar a solução [7].
- O código fornecido poderá ser melhorado ao longo do tempo, por isso é recomendado que configurem o git upstream [8] e verifiquem periodicamente a existência de novo código.
- É recomendado o uso de uma mensagem de token para a correcta propagação de mensagens no anel.
- As entidades que formam o anel devem ser capazes de manter a comunicação do anel enquanto executam o seu ciclo de vida.

5 Avaliação e Data de Entrega

O trabalho deverá ser submetido, até dia 22/04/2019, para o repositório git [9] correspondente (`https://classroom.github.com/a/g7_208gg`). O trabalho pode ser em grupos de dois ou individual. No entanto os alunos são incentivados a publicar periodicamente o código do projecto. Além do código os alunos devem entregar uma apresentação (em pdf) onde expõem

os pontos mais importantes da implementação. A apresentação deve ter no máximo 5 diapositivos.

Os seguinte pontos serão considerados para a avaliação:

- *Setup* de simulação (12 valores)
  - Criação do anel
  - Comunicação dentro do anel
  - Descoberta das entidades
  - Implementação de cada entidade
- Implementação da Simulação (5 valores)
  - Simulação correcta para um cliente
  - Simulação com mais de um cliente
- Performance e elegância da solução (3 valores)

Referências

[1] Wikipedia. Token ring. `https://en.wikipedia.org/wiki/Token_ring`, 2019. [Online; accessed 2019-03-19].

[2] Python. random — generate pseudo-random numbers. `https://docs.python.org/3/library/random.html#module-random`, 2019. [Online; accessed 2019-02-24].

[3] Python. time – time access and conversions. `https://docs.python.org/3/library/time.html#time.sleep`, 2019. [Online; accessed 2019-02-24].

[4] Python. logging – logging facility for python. `https://docs.python.org/3/library/logging.html?highlight=log#module-logging`, 2019. [Online; accessed 2019-02-24].

[5] Python. pickle – python object serialization. `https://docs.python.org/3/library/pickle.html`, 2019. [Online; accessed 2019-02-24].

[6] Python. Socket programming howto. `https://docs.python.org/3/howto/sockets.html`, 2019. [Online; accessed 2019-02-08].

[7] mariolpantunes. chord. https://github.com/mariolpantunes/restaurant-p2p, 2019. [Online; accessed 2019-03-18].

[8] GitHub. Configuring a remote for a fork. https://help.github.com/en/articles/configuring-a-remote-for-a-fork, 2019. [Online; accessed 2019-04-15].

[9] Roger Dudler. git – the simple guide. http://rogerdudler.github.io/git-guide/, 2019. [Online; accessed 2019-02-08].