

Computação Distribuída – Guião 02

Diogo Gomes & Mário Antunes

Fevereiro 2019

1 Introdução

O objectivo deste guião é desenvolver um sistema distribuído que implementa um restaurante *drive-through*. Existem 5 entidades distintas (ver Figura 1):

- Recepcionista: recebe os clientes e anota os pedidos (apenas um)
- Cozinheiro: recebe os pedidos e confecciona (dois)
- Empregado do balcão: entrega o pedido e recebe o pagamento (apenas um)
- Cliente: conduz pela vizinhança, quando tem fome espera no restaurante para efectuar um pedido (número configurável)
- Restaurante: entidade central que as restantes utilizam para se sincronizarem (apenas um).

Na versão centralizada do problema, o restaurante serve de entidade central. As restantes sincronizam-se entre si através de chamadas de funções remotas ao restaurante (RPC). Desta forma o acesso as regiões críticas está protegido pela entidade central. A cozinha é composta por 3 aparelhos:

- Grelhador: onde o cozinheiro pode preparar o hambúrguer (tempo médio de preparação: 30 milisegundos)
- Bebidas: onde o cozinheiro pode preparar as bebidas (tempo médio de preparação: 10 milisegundos)

- Fritadeira: onde o cozinheiro pode preparar as batatas (tempo médio de preparação: 50 milisegundos)

Os tempos de preparação seguem uma distribuição Gaussiana [1, 2] com a media indicada em cima e desvio padrão 5 milisegundos. Este aparelhos são únicos e **têm** de ser partilhados pelos cozinheiros. 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. Cada acção executada pelas entidades devera demorar um **tempo aleatório** configurável (distribuição gaussiana, utilize por omissão 10 milisegundos de média e desvio padrão de 5 milisegundos) [1, 2]. Para avaliar a correcta execução do algoritmo é obrigatório a utilização de logs [3]. Na Figura 2 está um exemplo das trocas de mensagens entre o cozinheiro e o restaurante.

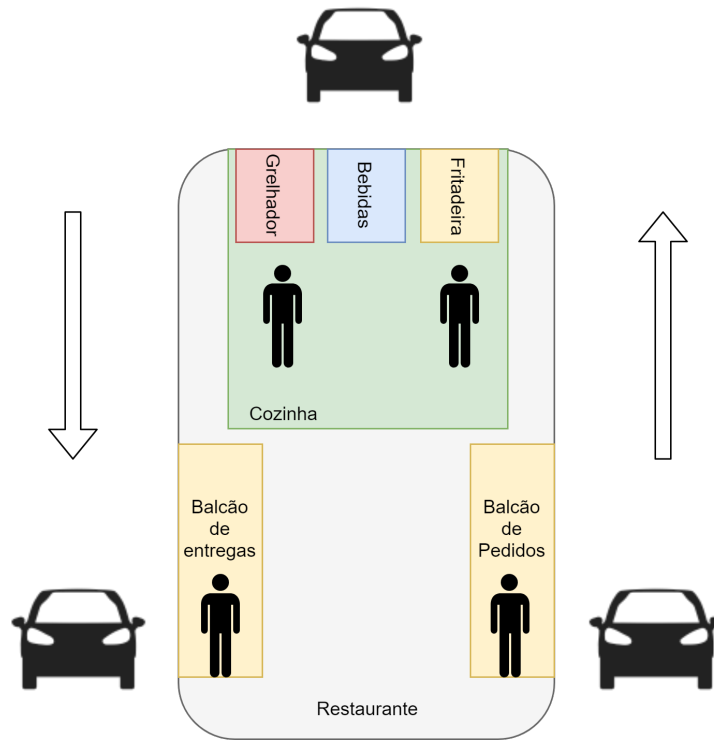


Figura 1: Esquema simplificado do *drive-through*.

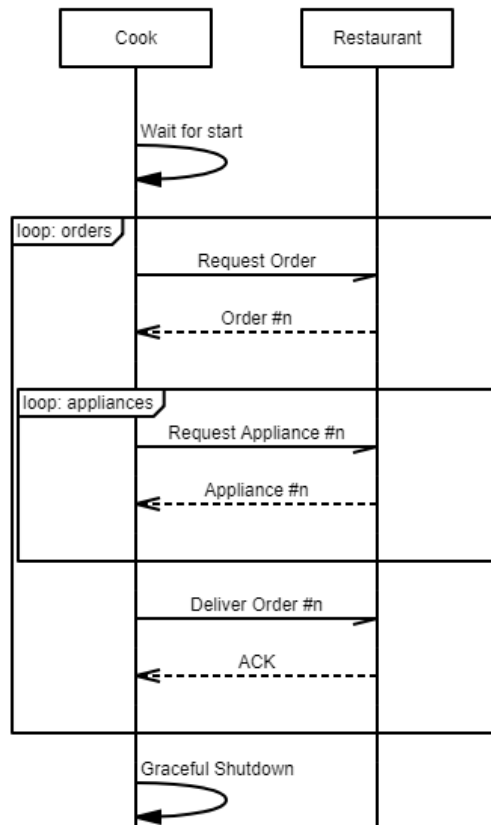


Figura 2: Trocas de mensagens pela entidade Cozinheiro com o Restaurante.

2 Objectivos

- Implementar o ciclo de vida das entidades descritas.
- Os pedidos dos clientes devem ser gerados de forma aleatória.
- Implementar a entidade central e garantir a protecção das regiões críticas.
- Comunicação através de ZeroMQ [4] com o socket tipo REQ/REP.
- As mensagens deverão ser codificadas em Pickle [5].
- implementar um mecanismo para começar o sistema apenas quando todas as entidades estão prontas

- implementar um mecanismo para terminar o sistema de forma graciosa [6].

3 Notas

- Como ponto de partida podem modelar as entidades como objectos [7, 8], uma classe para cada entidade (incluindo os aparelhos partilhados).
- Devido a restrições na rede Eduroam [9], o sistema apenas funcionará no seu computador local.
- Os objectos que servem de mensagens são livres (poderam ser strings, listas, tuplos, dicionários, etc.).
- Como ponto de partida podem usar o código partilhado em [10].
- Como desafio pode expandir otimizar a utilização dos aparelhos da cozinha (cuidado com deadlocks e starvation).
- O código deve ser armazenado/partilhado no GitHub [11] classroom - criar projecto via <https://classroom.github.com/a/YdxcWJTo> - (tutorial de git [12])

Referências

- [1] Python. random — generate pseudo-random numbers. <https://docs.python.org/3/library/random.html#module-random>, 2019. [Online; accessed 2019-02-24].
- [2] Python. time – time access and conversions. <https://docs.python.org/3/library/time.html#time.sleep>, 2019. [Online; accessed 2019-02-24].
- [3] Python. logging – logging facility for python. <https://docs.python.org/3/library/logging.html?highlight=log#module-logging>, 2019. [Online; accessed 2019-02-24].
- [4] ZeroMQ. Distributed messaging – zeromq. <http://zeromq.org/>, 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. signal – set handlers for asynchronous events. <https://docs.python.org/3.7/library/signal.html>, 2019. [Online; accessed 2019-02-24].
- [7] A Byte of Python. Object oriented programming. <https://python.swaroopch.com/oop.html>, 2019. [Online; accessed 2019-03-01].
- [8] Python. Classes. <https://docs.python.org/3/tutorial/classes.html>, 2019. [Online; accessed 2019-03-01].
- [9] STIC. Rede wireless – eduroam. <https://www.ua.pt/stic/PageText.aspx?id=15224>, 2019. [Online; accessed 2019-02-24].
- [10] mariolpantunes. drive-through. <https://github.com/mariolpantunes/drive-through>, 2019. [Online; accessed 2019-03-01].
- [11] GitHub. Dep. electrónica telecomunicações e informática. <https://github.com/detiuaveiro>, 2019. [Online; accessed 2019-02-08].
- [12] Roger Dudler. git – the simple guide. <http://rogerdudler.github.io/git-guide/>, 2019. [Online; accessed 2019-02-08].