Drive Through P2P

Realizado por: Luís Fonseca nº 89066

As entidades podem ser inicializadas em qualquer ordem; Não obstante é necessária a eventual ligação do *Restaurant* para que o processo de formação realmente comece.

Para facilitar a visualização dos *logs*, cada um dos ficheiros das Entidades (*Chef*, *Employee*, *Restaurant*, *Receptionist*) contém uma condição que permite que sejam inicializadas em terminais diferentes

```
if __name__ == '__main__':
```

Todas as Classes das Entidades contêm argumentos por *default*, podendo, obviamente, estes ser alterados.

```
def __init__(self, address=('localhost', 5000), id=0, init_address=None, timeout=3):
```

A Classe *Node,* encontra-se por Composição nas Entidades. Isto visando a que todos os Nodes do Ring sejam o mais semelhantes possíveis

Node

Constituído por 2 QUEUEs:

- uma **IN**, onde se colocam mensagem a ser processadas pela Entidade que o contêm
- uma **OUT**, onde são colocadas mensagens a ser enviadas, podendo estas ter ou não sido processadas pela Entidade

NODE JOIN

Bastante semelhante ao do guião de DHT.

NODE DISCOVERY

Após estar no Ring, o *Node* está atendo a outras mensagens de NODE_JOIN e NODE_DISCOVERY. Estas ultimas são enviadas com uma certa periodicidade à volta do Ring, até voltar ao *sender*. Estas mensagens contêm o Nome, ID, e nº de Nodes conhecidos pelo Node num dado momento. Permite portanto que todos os Nodes saibam quantos Nodes os outros conhecem. Isto possibilita assegurar que o processo de DISCOVERY só esteja concluído quando todas os Nodes se conhecerem entre si. É o Node com o ID mais pequeno (*usually* o *Restaurant*), que termina o processo de DISCOVERY, enviando uma mensagem RING_DONE à volta do mesmo.

TOKEN

É também o Node com ID mais baixo que cria e envia o TOKEN.

SIMULAÇÃO

Excluindo o TOKEN, qualquer Node está capacitado a receber mais dois tipos de mensagens: ORDER e PICKUP, originados do *Client.*
Qualquer TOKEN cujo ‘args’ não seja *None*, contém uma *key*: ‘*destiny*’ que indica o ID destino do Node. Quando esta mensagem chega a um Node, se este for o destino a mensagens é colocada na **IN_QUEUE**, caso contrário é colocada diretamente na **OUT_QUEUE**.

Receptionist

ORDER

Única mensagem que processa.
Gera um *UUID* a atribuir a cada pedido que recebe.
Envia uma mensagem de *acknowledgement* ao *Client* com o UUID do pedido.
Por fim envia também uma mensagem PLATE ao *Chef* com o Menu a confecionar e o UUID do pedido.

Employee

Constituído por um Dicionário

Onde se guarda ID : ADDR do *Client*.
Este é preenchido sempre que é recebida uma mensagem de **PICKUP.**

PICKUP

Uma das mensagens que processa, oriunda do *Client.*
Guarda o *UUID* que recebe e associa-o a um endereço, relativo ao *Client* no Dicionário.

DONE

Ao recebê-la, significa que o *Chef* acabou um dos pedidos.
Portanto, analisa qual o *UUII* para saber a qual *Client* tem de entregar o Menu.

Chef

Constituído por 2 QUEUEs:

- uma **PLATES**, onde se colocam os Menus a ser confecionados. É preenchida graças a mensagens **PLATE**.
- uma **PERMITIONS**, onde são colocadas as permissões que o *Restaurant* dá. Normalmente só tem sizeMax=1, mas é usado por razões de "sincronização". É preenchida graças a mensagens iniciadas por **REP** (de Reply)

new Thread

É constituído por uma Thread adicional, que simula o processo de *Cooking.* A razão da mesma é não impedir o *Chef* de receber mensagens, para posteriormente colocar nas QUEUES referidas anteriormente.

cooking

Sempre que houver um Menu em **PLATES**, é iniciado o processo de cozinhar. Os ingredientes do Menu são escolhidos aleatoriamente, para na eventualidade de uma das Maquinas dos *Restaurant* estar ocupada, se tentar outra. Este problema não acontece pois só esta presente um *Chef* na simulação, mas fica a ideia.
Assim, sempre que é necessário uma Máquina, um pedido é enviado ao *Restaurant*, e o processo (de *cooking*) bloqueia até alguma coisa ser colocada em **PERMITIONS**.

Funções de Set

Finalmente possuí duas funções que tudo o que fazem é indicar o tipo de mensagem a ser enviada ao Restaurante.

Restaurant

Constituído por 3 variáveis especiais:

- ***Grill***
- ***Fryer***
- ***Bottle***

Cada uma delas representa uma máquina no *Restaurant*. Quando a TRUE, indicam que dada máquina está disponível, FALSE funciona de forma análoga.

REQest

Um dos tipos de mensagem que processa. É oriundo do *Chef* no seu processo de cozinhar. Tem então uma função associada, que seleciona uma resposta a enviar, dependendo do tipo de pedido feito e do estado das maquinas

RESPonse

A outra mensagem que processa. Corresponde à devolução da máquina por parte do *Chef*, pelo que na função que tem associada, procura restaurar os valores de dada maquina.