RELATÓRIO: GUIÃO DE AVALIAÇÃO 01

Computação Distribuida

Estudantes Responsaveis:
Camila Uachave (90711)
Jean Brito (82784)

Docentes Responsáveis:
Diogo Gomes
Mário Antunes

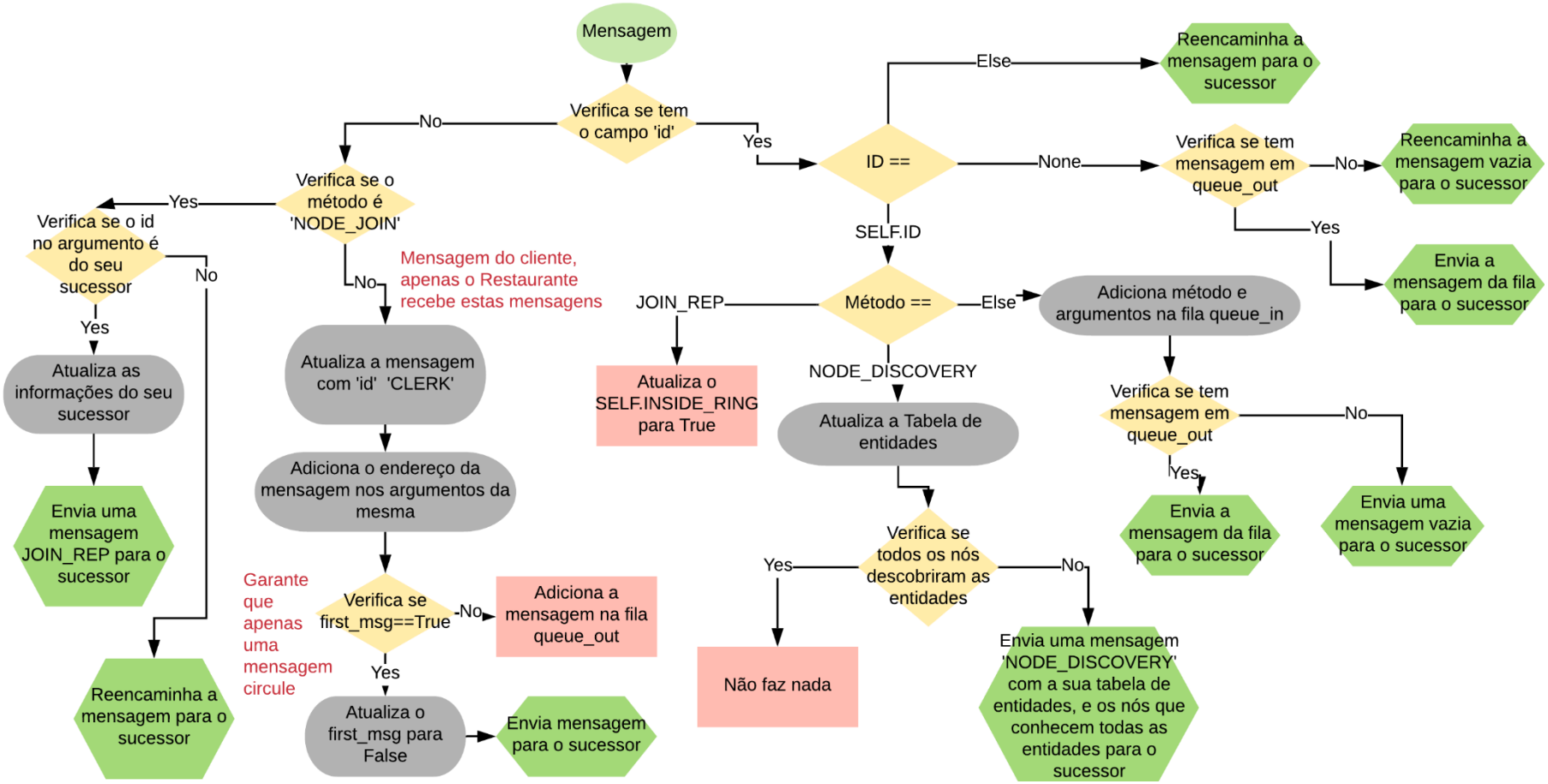
Criação do Anel

- É criado um bool, *inside_ring* inicializada a *False* para todas as entidades que recebem um endereço de encaminhamento, ou sucessor.(Todas começam a False, exceto o Restaurante);
- Enquanto não estiverem no anel, todas as entidades enviaram uma mensagem com o método ("NODE_JOIN") para o seu respectivo sucessor, esta mensagem, conterá em seus arguemntos o respectivo **ID** e o **endereço** da entidade;
- Quando um nó recebe um **NODE_JOIN,** ela verifica se o id da mensagem é o do seu successor, caso seja, ela atualiza o id e o endereço do seu successor, e envia um **JOIN_REP** para o successor, caso não seja, o nó simplemente reencaminha a mensagem para o successor atual;
- Quando um nó recebe um **JOIN_REP** ela atualiza o bool *inside_ring* para *True,* e para enviar mensagens **NODE_JOIN,** pois esta garantido que alguém tem a si como successor, então as mensagens chegaram a ele de alguma forma;
- Um resumo rápido sobre a descoberta, todos os nós enviam uma mensagem para o nó com o id 0, o id 0 vai receber todas, e verificar que o ID 1 é o seu successor, o id 0 envia para o id 1 o **JOIN_REP,** a partir deste momento, o id 1 para de enviar o **NODE_JOIN**, e o ID 0 passa a reencaminhar todas as mensagens para o id 1. Após algumas interações todos os nós terão um successor e as mensagens cessarão.

Descoberta de Entidades

- Na inicialização são criadas um *dict* **table** contendo a tabela das entidades como chave, e todos os valores são inicializados como *None* e uma variavel *bool*, **discovered** inicializada a False.
- Cada nó sabe qual é a sua própria chave, e atualiza sua tabela com a informação do seu próprio **id.**
- Após entrar no anel, cada uma dos nós envia uma mensagem **NODE_DISCOVERY** para o seu successor, com 'id' também do sucessor**.** Nos argumentos, é enviado a sua propria **table,** e uma lista **discovered_table** com a informação de quais nós tem sua tabela completa. Esta mensagem é enviada apenas uma vez por cada nó.
- Ao receber uma mensagem **NODE_DISCOVERY,** o nó atualiza sua tabela de entidades, se possuir todas as, ele passa a bool **discovered** para *True.* O nó coloca a informação da variavel **discovered** em sua posição na lista **discovered_table** e verifica se todas os nós tem suas tabelas completas, neste caso ele não faz nada. Caso algum nó não tenha sua tabela complete, é enviado uma mensagem **NODE_DISCOVERY** para o successor, com a tabela de entidades e a **discovered_table.**

Processamento das Mensagens



Funcionamento das Entidades

- Clerk:
  - Faz o controle dos pedidos com um contador de tickets, ao receber um pedido **ORDER**, reencaminha as mensagens dos clientes para o **Chef** com o pedido e o ticket, e para o **Waiter** com o ticket e o endereço.
- Chef:
  - Recebe os pedidos, os processa os pedidos e armazena em dois *dict*, **recv_orders** e **deliever_orders** com as chaves sendo os **tickets** dos pedidos;
  - O chef processa, e envia de forma aleatória cada item do pedido com o **ticket** para o Restaurant;
  - Toda vez que recebe um item fica pronto, o **Chef** atualiza **deliever_orders** e verifica se todos os itens estão prontos, neste caso ele envia o pedido para o Waiter.
- Restaurant:
  - Possui uma fila para cada um dos seus aparelhos, ao receber um pedido de algum item do Chef, adiciona na lista do aparelho o ticket do pedido;
  - Quando o aparelho esta disponível, ele verifica se há algo na lista, e depois verifica se o aparelho terminou o item, neste caso envia para o Chef, o item e o **ticket.**
- Waiter:
  - Recebe os pedidos do **client** e as entregas do **Chef,** controla os pedidos atravez de dois dicts, um com os endereços dos clientes e outro com os pedidos, toda vez que recebe uma mensagem verifica se o cliente solicitou e se o pedido esta pronto, caso ambos estejam, envia para o cliente o pedido.

Observações

- As mensagens utilizadas contém sempre 3 campos: {id, method, args}. As únicas exceções são as mensagens recebidas e entregues aos clientes, e as mensagens **NODE_JOIN**.
- As entidades utilizam duas funções para interagir com seus nós, **queuein(),** que retorna as mensagens respectivas da entidade quando há algo a ser processado, quando não, retorna *None. A segundo Função é* ***queueout()*** *que simplemente adiciona na fila do nó as mensagens que devem ser enviadas.*
- Os nós só geram as mensagens **NODE_JOIN e NODE_DISCOVERY**. Após isso esperam uma mensagem do cliente e quando chega, apenas uma mensagem cirula por entre os nós.
- Apenas o **CLERK**, e o **WAITER** enviam mensagens para fora do anel, estas mensagens são enviadas independetentemente das mensagens circulando no anel.
- Apenas o Restaurante recebe mensagens de fora, e inicia a comunicação dentro do anel, o Restaurante garante que apenas uma mensagem circula internamente.
- Todas as entidades recebem a tabela de entidades contendo o ID de cada uma, para ser usado nas mensagens que serão colocadas nas filas **queueout().**
- Waiter é a unica entidade que não utiliza o **queueout.**