P6+P7

Grupo F
Luiz Maurício
Rafael
Alexis
Luis

P6 - Nodos e grupos

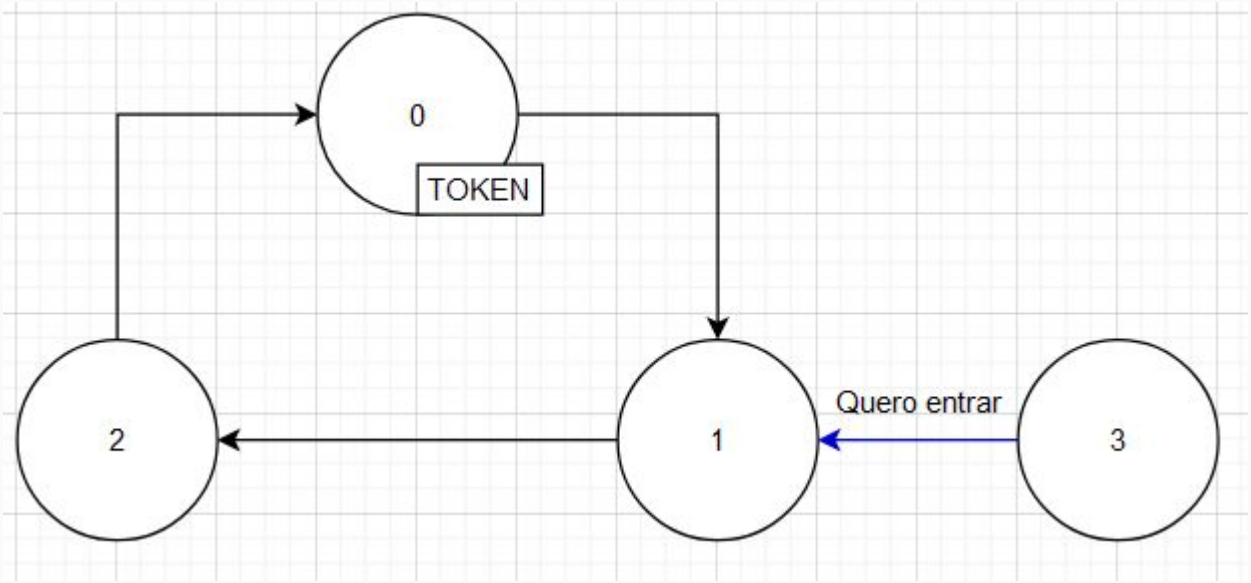
- Há um std::map onde um nodo guarda todos os nodos dos quais tem conhecimento e seus endereços

```
std::map<int, std::pair<std::string, int>> nodes;
```

- Há std::map onde o nodo guarda as informações de cada grupo ao qual pertence, como quais nodos fazem parte dele e se possui o token relativo ao grupo ou não

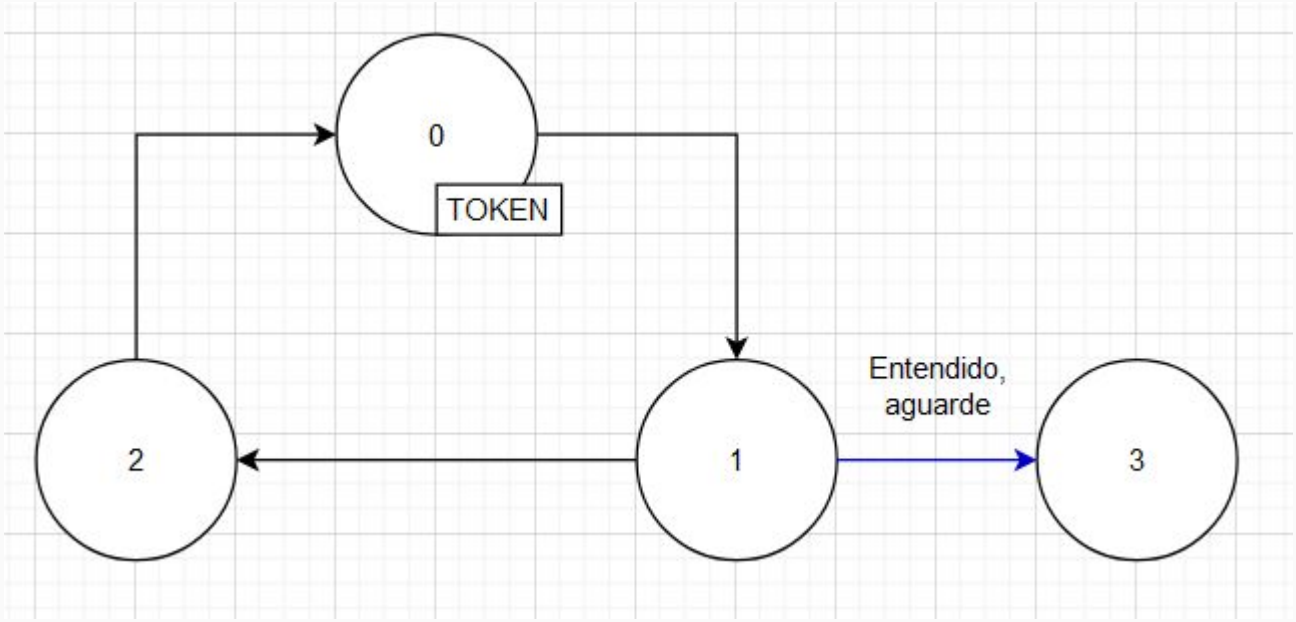
```
std::map<int, std::queue<int>> group_members;
std::map<int, bool> has_group_token;
```

P6 - entrada no grupo



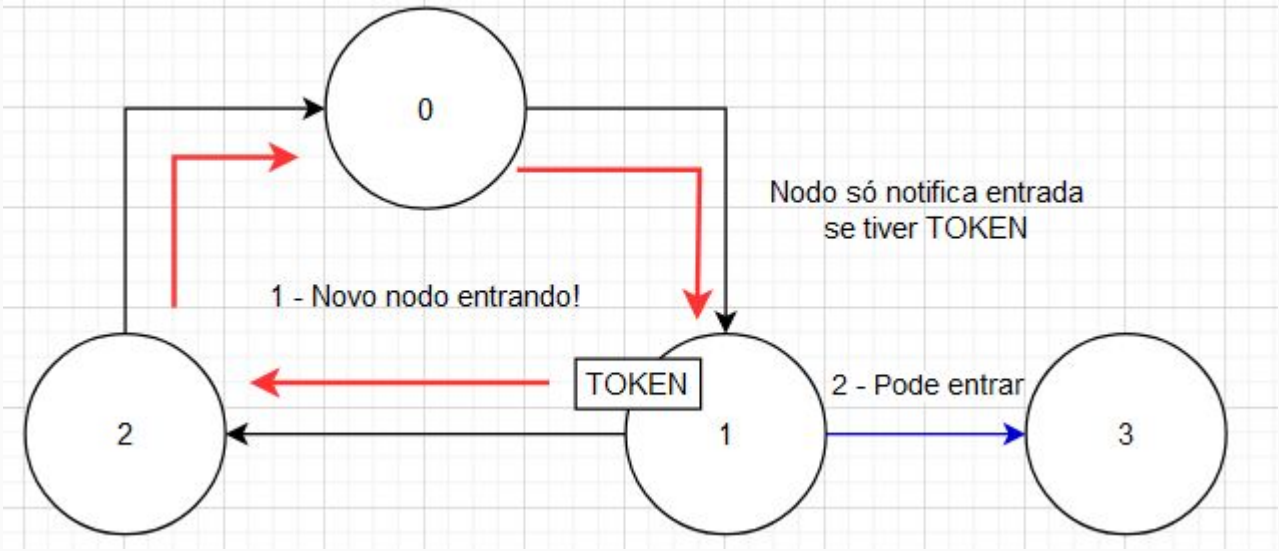
- Envio da requisição de entrada

P6 - entrada no grupo



- Confirmação do recebimento dela.
- Nodo 3 ainda não está no grupo

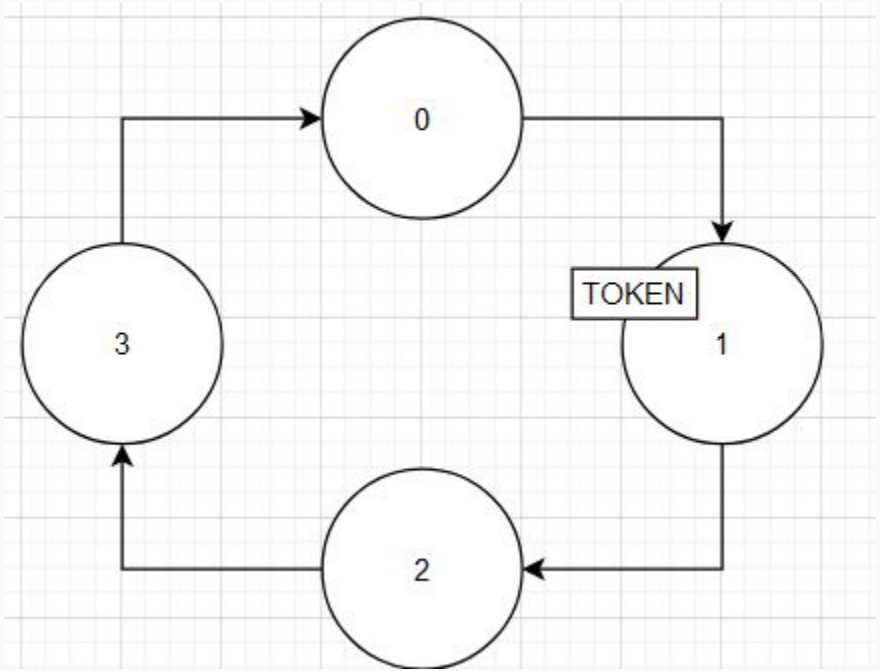
P6 - entrada no grupo



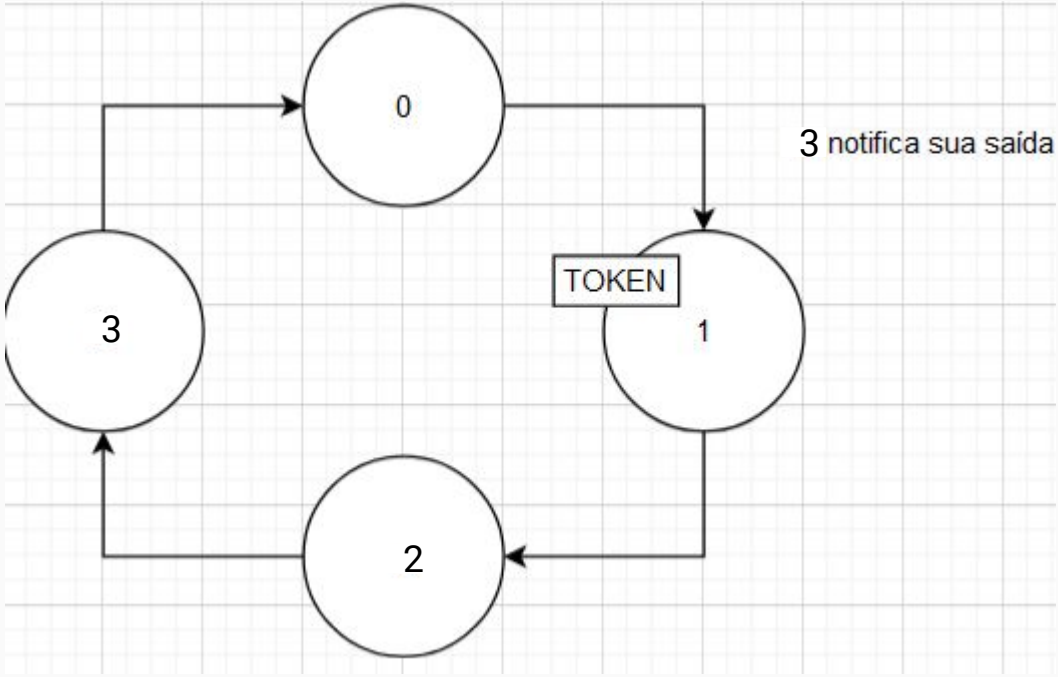
Quando Nodo 1 adquire o token e se torna “líder” do grupo:

1. Notifica membros de novo Nodo
2. Permite entrada de Nodo 3

P6 - entrada no grupo

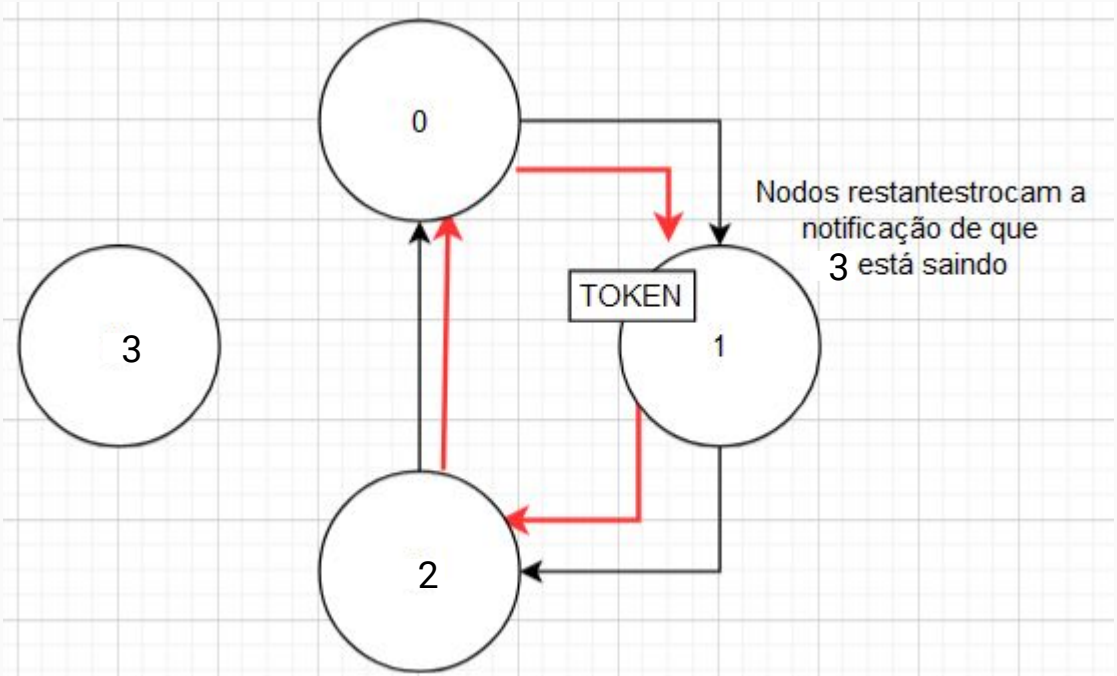


P6 - Saída Voluntária

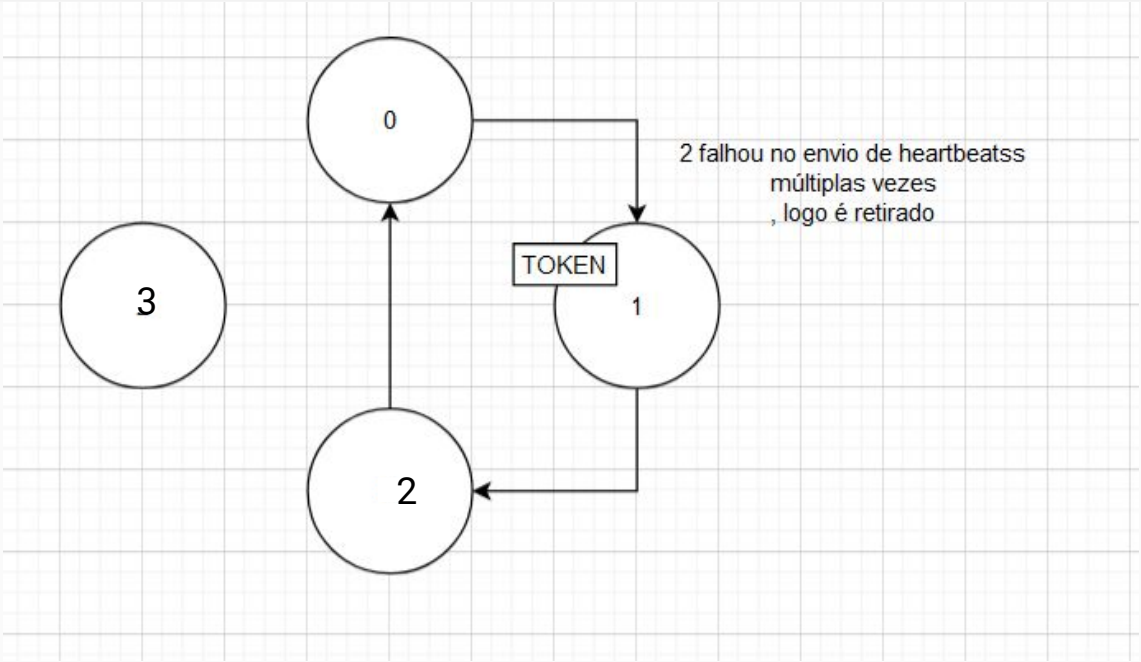


- Envia Mensagem de “LEAVE” para o grupo
- Mensagem é processada, nodo é removido da lista do grupo e

P6 - Saída Voluntária

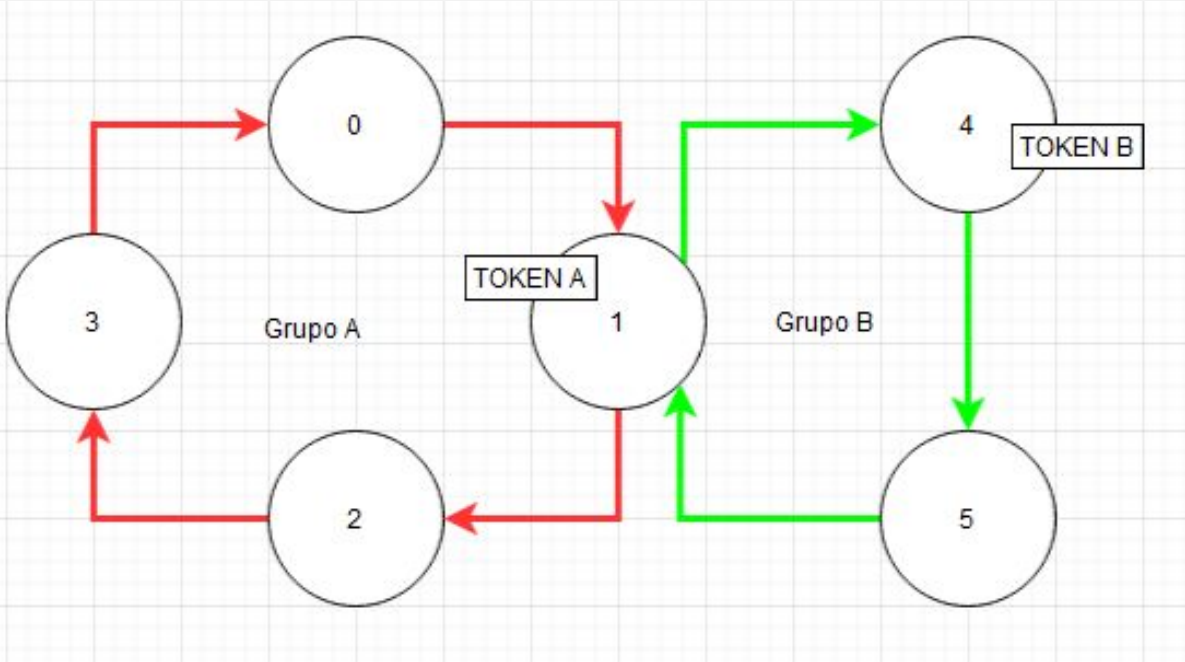


P6 - Saída por falha



- Caso um nodo falhe por múltiplos rounds no envio de heartbeats, é retirado do grupo
- Antes um falso positivo do que um falso negativo

P6 - Múltiplos grupos



P7 - Gerador

- Parâmetros existentes para geração de requisições:
  - Total de requisições por thread
  - Chance de requisição ser do tipo escrita
  - Número de threads que rodam a função generate_message()
  - Sleep entre requisições é aleatório entre 10ms e 200ms

```
void generator::generate_message(){
    for(int i=0; i<total_ops; i++){
        // Generate a message
        int r = std::rand() % 100;
        Request req = Request(i, i, "SET"); // Defa
        if (r>write_chance){
            // Read operation if chance happens
            req = Request(i, i, "GET");
        }

        // Add the message to the queue
        message_queue.push(req);
        cv_send.notify_all();

        int rand_sleep = rand()%(200-10 + 1) + 10;
        usleep(100);
    }
}
```

P7 - Banco de dados

```
void bd::run()
{
    while (true) {
        Message received = comm->receive(); // Receive message from generator

        if (received.msg_type=="ERR") {
            //std::cout << "\n\n ERROR: Nothing to receive \n";
            continue;
        }

        req_amount++;

        Request req = Request::deserialize(received.content);
        if (req.type=="SET") {
            data[req.key] = req.val;
            Request result(req.key, req.val, "YES");
            comm->send(gen_id, result.serialize());
        } else {
            if (data.find(req.key) == data.end()) {
                Request result(-1, -1, "NOO");
                comm->send(gen_id, result.serialize());
            } else {
                std::vector<uint8_t> data_vector(sizeof(int));
                std::memcpy(data_vector.data(), &data[req.key], sizeof(int));
                comm->send(gen_id, data_vector);
```

Função que trata requisições recebidas

Uma thread é responsável por medir vazão por segundo

P7 - Logs gerados

- Vazão computada para 4 e 8 threads
- Registro de vazão realizado no lado do Banco de Dados

- Latência de requisições computada no lado do gerador

Vazão
4 threads

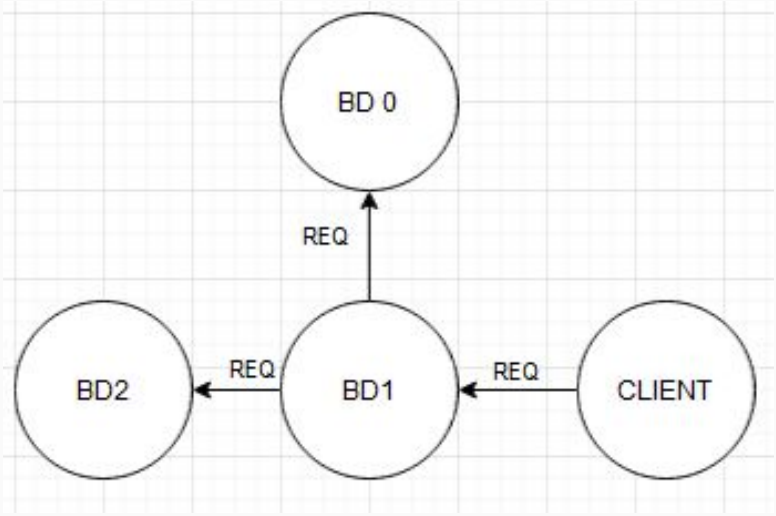
```
43,
25,
38,
21,
29,
41,
53,
40,
18,
15,
```

Latência
em ms

```
127,
178,
142,
198,
179,
179,
172,
```

P7 - Difusão de requisições

Infelizmente, a difusão de requisições entre múltiplas instâncias do banco de dados ainda não está funcional



P6 + P7

Obrigado!