EP1-AMQP

Adriano Elias Andrade
nº USP: 13671682

ARQUITETURA GERAL

OPERAÇÕES

Queue Declare

- Abre uma conexão
- Declara uma fila
- Fecha a conexão

Queue Consume

- Abre uma conexão
- Escuta uma fila

Publish

- Abre uma conexão
- Publica numa fila
- Fecha a conexão

ARQUITETURA GERAL

Queue Declare

- Abre uma conexão
- Declara uma fila
- Fecha a conexão

Queue Consume

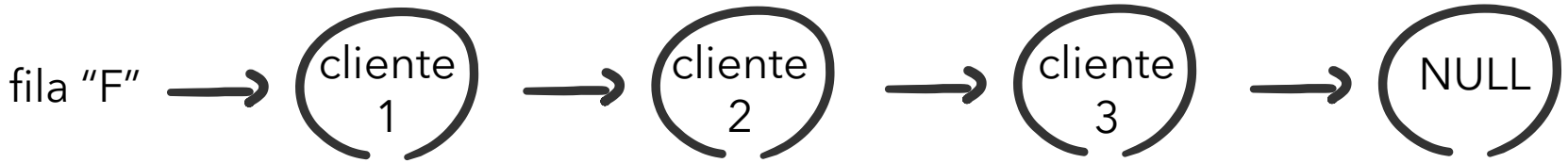
- Abre uma conexão
- Escuta uma fila

Publish

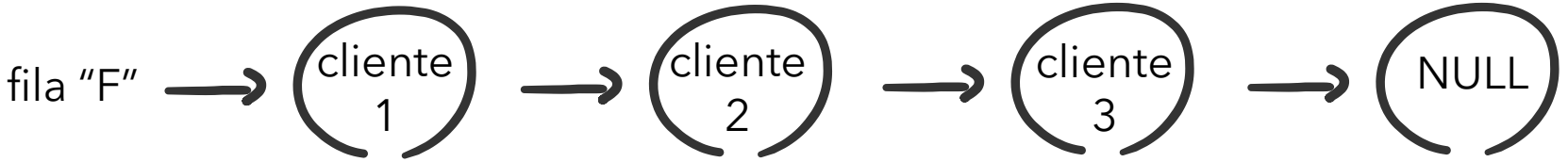
- Abre uma conexão
- Publica numa fila
- Fecha a conexão

FILAS DE CLIENTES

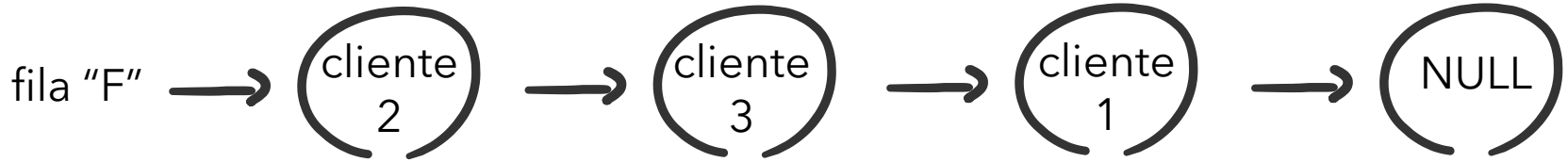
Para armazenar quais clientes estão numa fila, usamos uma lista ligada:



Na operação first() de uma fila, além de devolver o primeiro elemento, ele também é colocado no final da fila.



first()



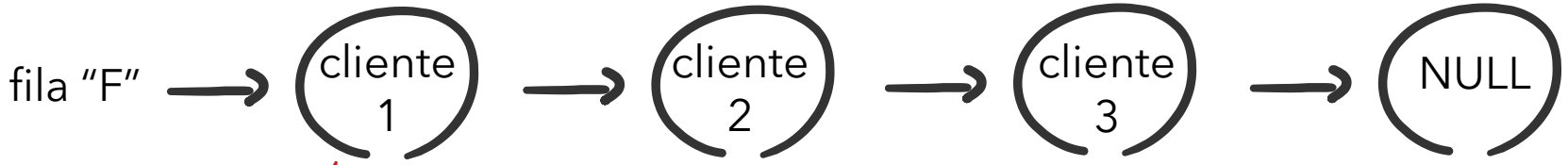
Com isso, o esquema de Round Robin já está funcionando.

FILAS DE CLIENTES

Para armazenar quais clientes estão numa fila, usamos uma lista ligada:

```
typedef struct{
    // Fila feita com lista ligada
    node_q* first;    // primeiro nó
    node_q* last;     // último nó
}queue;
```

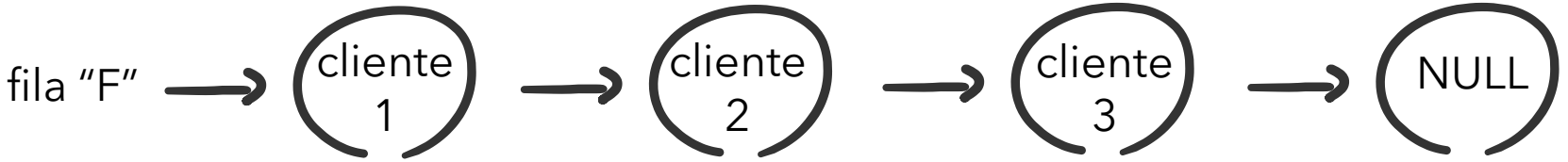
um ponteiro para o começo, e outro para o fim (para fazer inserções)




```
typedef struct node_q{
    int val;
    struct node_q* next;
}node_q;
```

“val” contém o connfd do cliente

Na op a fila, além de devolver o primeiro elem colocado no final da fila.

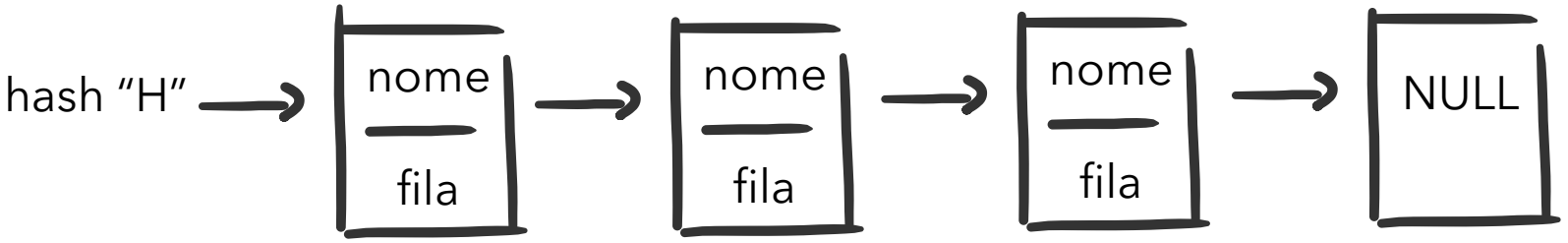


first()

Com isso, o esquema de Round Robin já está funcionando.

HASH DE FILAS

Para referenciar as filas declaradas de acordo com o seu nome, usamos um hash:



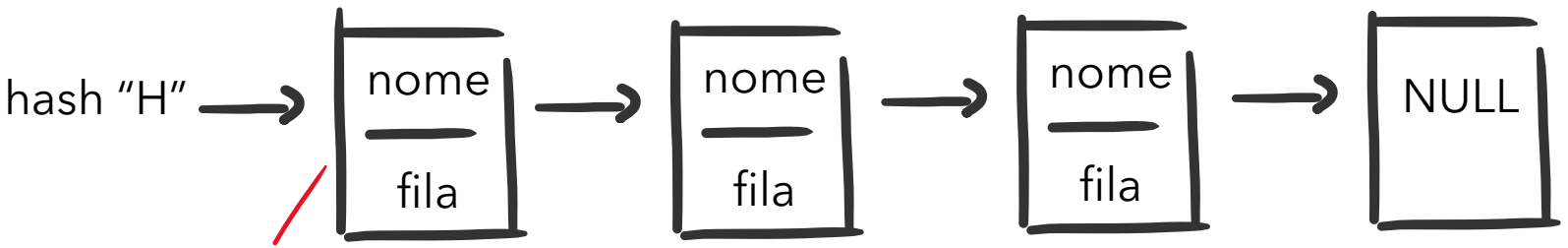
Para achar uma fila, percorremos o hash até encontrar o nome dela, ou chegar no final.

HASH DE FILAS

Para referenciar as filas declaradas de acordo com o seu nome, usamos um hash:

```
typedef struct{
    // Hash feito com lista ligada
    node_h* first;    // primeiro nó
    node_h* last;    // último nó
}hash;
```

um ponteiro para o começo, e outro para o fim (para fazer inserções)




```
typedef struct node_h{
    char* name;
    queue* q;
    struct node_h* next;
}node_h;
```

nó com nome e fila

Para achar uma fila, percorremos o hash até encontrar o nome dela, ou chegar no final.

QUEUE DECLARE

- Abre um cliente
- Cria a fila e coloca no hash
- Reads e writes das frames do protocolo
- Sai da thread do cliente

QUEUE DECLARE

- Abre um cliente

```
connection_establish(recvline, connfd);
```

Rotina responsável por:
- Connection.start
- Connection.tune
- Connection.open
- Channel.open

- Cria a fila e coloca no hash

```
queue* q = new_queue();
add(queues, queue_name, q);
```

new_queue(): cria a fila
add(): coloca a fila no hash

- Reads e writes das frames do protocolo

```
stop_connection = make_request(recvline, connfd, queues);
```

Função responsável por redirecionar para a rotina que o cliente pediu → **Nesse caso:**
- Queue.declare
- Channel.close
- Connection.close

- Sai da thread do cliente

```
printf("[Uma conexão fechada]\n\n");
return NULL;
```

QUEUE CONSUME

- Abre um cliente
- Inscreve o cliente na fila
- Reads e writes das frames do protocolo
- Fica num loop, escutando por mensagens

QUEUE CONSUME

- Abre um cliente

```
connection_establish(recvline, connfd);
```

Rotina responsável por:
- Connection.start
- Connection.tune
- Connection.open
- Channel.open

- Inscreve o cliente na fila

```
queue* q = find(queues, queue_name);
if(q==NULL){
    /// printf("Queue does not exist\n");
    exit(1);
}
enqueue(q, connfd);
```

find(): busca a fila no hash
enqueue(): bota o cliente na fila

- Reads e writes das frames do protocolo

```
stop_connection = make_request(recvline, connfd, queues);
```

Função responsável por redirecionar para a rotina que o cliente pediu → **Nesse caso:**
- Basic.consume

- Fica num loop, escutando por mensagens

```
while(1){
    if(read(connfd, recvline, 1)==0){
        char* queue_name = find_and_dequeue(queues, connfd);
        printf("Cliente desconectou da fila \"%s\", port %d\n\n", queue_name, connfd);
        break;
    }
}
```

O cliente fica aberto até que se desconecte (a função read() retorna 0)
find_and_dequeue(): procura o cliente no hash e tira ele da fila

BASIC PUBLISH

- Abre um cliente
- Reads e writes das frames do protocolo
- Entrega a mensagem no esquema de Round Robin
- Sai da thread do cliente

BASIC PUBLISH

Abre um cliente

```
connection_establish(recvline, connfd);
```

Rotina responsável por:
- **Connection.start**
- **Connection.tune**
- **Connection.open**
- **Channel.open**

Reads e writes das frames do protocolo

```
stop_connection = make_request(recvline, connfd, queues);
```

Função responsável por redirecionar para a rotina que o cliente pediu → **Nesse caso:**
- **Basic.publish**
- **Channel.close**
- **Basic.deliver**
- **Connection.close**

Entrega a mensagem no esquema de Round Robin

```
queue *q = find(queues, queue_name);
if(q==NULL){
    printf("A fila não existe!\n\n");
    return;
}
int connfd = first(q);
if(connfd==NULL){
    printf("Fila vazia, mensagem não entregue!\n\n");
    return;
}

printf("Mensagem entregue na fila \"%s\", port %d\n\n", queue_name, connfd);
write(connfd, deliver_frame, deliver_size);
```

find(): acha a fila requisitada
first(): retorna o primeiro consumidor da fila e coloca ele em último
Por fim, dá write no connfd do consumidor

Sai da thread do cliente

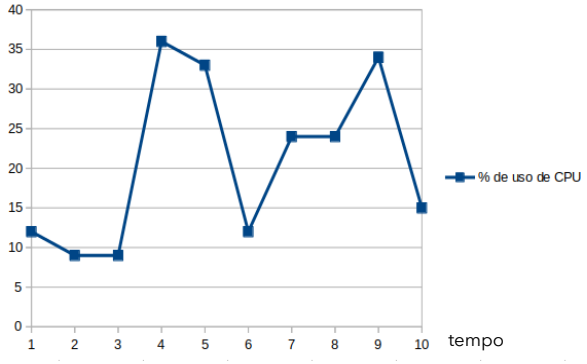
```
printf("[Uma conexão fechada]\n\n");
return NULL;
```

DESEMPENHO

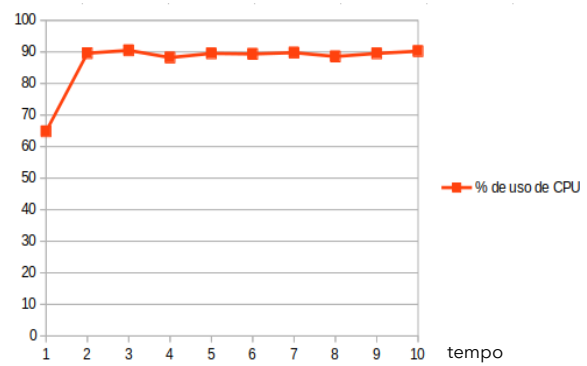
Os testes foram realizados numa máquina virtual kubuntu.

Para garantir que o uso de CPU fosse condizente ao processamento do programa, só foram medidos dados do port 5672.

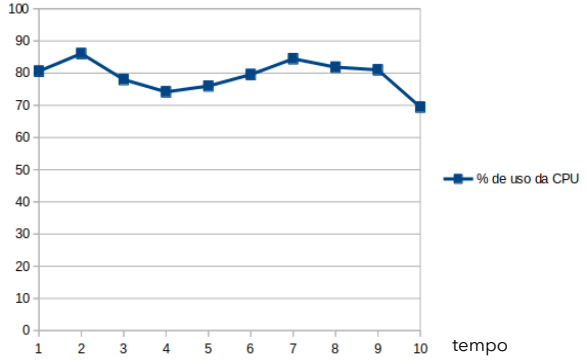
A seguir, testes com 0, 10, e 100 clientes:



0 clientes
0 bytes



10 clientes
111529 bytes



100 clientes
1207204 bytes