

Gabriel Medeiros Lopes Carneiro
Paulo Arthur Sens Coelho
Erik Kazuo Sugawara

Relatório

Florianópolis, SC

2022

Gabriel Medeiros Lopes Carneiro
Paulo Arthur Sens Coelho
Erik Kazuo Sugawara

Relatório

Trabalho apresentado para avaliação de uma implementação de uma biblioteca para comunicação confiável e algoritmo distribuído, na disciplina INE 5418 - Computação Distribuída, sob Orientação do Prof. Dr. Odorico Machado Mendizabal

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Programa de Graduação

Orientador: Odorico Machado Mendizabal

Florianópolis, SC

2022

Gabriel Medeiros Lopes Carneiro
Paulo Arthur Sens Coelho
Erik Kazuo Sugawara
Relatório/ Gabriel Medeiros Lopes Carneiro
Paulo Arthur Sens Coelho
Erik Kazuo Sugawara. – Florianópolis, SC, 2022-
13 p. : il. (algumas color.) ; 30 cm.

Orientador: Odorico Machado Mendizabal

Relatório (Graduação) – Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Programa de Graduação, 2022.

1. Computação Distribuída. 2. Sockets. 3. Ordem Causal. 4. Ordem Total. 5.
Biblioteca.

CDU 02:141:005.7

Gabriel Medeiros Lopes Carneiro
Paulo Arthur Sens Coelho
Erik Kazuo Sugawara

Relatório

Trabalho apresentado para avaliação de uma implementação de uma biblioteca para comunicação confiável e algoritmo distribuído, na disciplina INE 5418 - Computação Distribuída, sob Orientação do Prof. Dr. Odorico Machado Mendizabal

Odorico Machado Mendizabal
Orientador

Florianópolis, SC
2022

Lista de ilustrações

Figura 1 – Estrutura de um nodo	9
Figura 2 – Estrutura do recebimento de mensagens	10
Figura 3 – Estrutura do envio de mensagens	10
Figura 4 – Envio de mensagens sendo entregue entre os processos.	11

Sumário

	Identificação	6
I	BIBLIOTECA DE COMUNICAÇÃO CONFIÁVEL	7
1	SOBRE O TRABALHO	8
1.1	Introdução	8
1.2	Metodologia	8
2	DESENVOLVIMENTO	9
2.1	Biblioteca	9
2.1.1	Nodo	9
2.2	Mensagem	9
2.2.1	Recebimento de mensagens	10
2.2.2	Envio de mensagens	10
2.3	Algoritmo Token Ring	10
2.4	Execução	11
2.5	Resultados	11
2.6	Conclusão	11
2.7	Referências	11
	ANEXO A – BIBLIOTECA	12
A.1	Código	12
	ANEXO B – ALGORITMO	13

Identificação

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Ciências da Computação
INE 5425 - Modelagem e Simulação

Contato

Gabriel Medeiros Lopes Carneiro
Email: gabriel.mlc@grad.ufsc.br

Paulo Arthur Sens Coelho
Email: paulo.sens.coelho@grad.ufsc.br

Erik Kazuo Sugawara
Email: erik.sugawara@grad.ufsc.br

Parte I

Biblioteca de Comunicação Confiável

1 Sobre o Trabalho

1.1 Introdução

Um sistema distribuído é caracterizado por componentes localizados em uma rede de computadores, que se comunicam e coordenam suas ações através de troca de mensagens e que podem sofrer influências internas ou externas, as quais podem prejudicar a sua interoperabilidade. Através da concorrência entre os componentes comunicantes, o estado de seus dados se tornam inconsistentes a medida que não existe uma sincronização. Seja por conta da ausência de um relógio global ou de uma memória compartilhada, se torna impossível realizar uma sincronização perfeita entre as trocas de mensagens. Além disso, em seu aspecto físico, componentes são suscetíveis a falha e podem impedir o funcionamento do sistema. Sendo assim, o objetivo deste trabalho é criar uma biblioteca de comunicação confiável que seja capaz de garantir a ordem no recebimento e envio de mensagens seguindo critérios de ordem causal e total, de modo que os participantes da comunicação consigam estabelecer troca de mensagens do tipo $1 : 1$ e $1 : n$.

1.2 Metodologia

Para realizar a criação da biblioteca foi escolhida a linguagem Python por sua versatilidade e legibilidade. Em conjunto, foi usado a biblioteca de Multiprocessos para permitir que os processos que instanciam os sockets sejam possíveis de serem executados em paralelo, garantindo o recebimento e envio de mensagens de múltiplos nodos. Cada nodo contém dois processos que instanciam sockets: o primeiro é utilizado para o recebimento de mensagens e o segundo é utilizado para enviar mensagens, uma vez que não conseguimos reutilizar o mesmo socket para realizar operações de envio e recebimento. Para garantir a entrega dos dados, foi utilizado o Protocolo TCP. No socket foi utilizado o domínio AF_INET para criarmos um socket capaz utilizar os endereços com seus respectivos Ip e Porta, dessa forma os nodos são capazes de comunicar entre si. Além disso, a garantia do recebimento de pacotes ponto a ponto facilita a elaboração de uma biblioteca mais robusta.

2 Desenvolvimento

2.1 Biblioteca

2.1.1 Nodo

A estrutura de cada nodo é composta por três tipos de buffer: do processo, de entrada e de saída. O de processo é responsável por armazenar as mensagens que precisam aguardar outras chegarem. O buffer de entrada armazena os IDs de cada mensagens e verifica se a mensagem já pode ser entregue. O buffer de saída adiciona os IDs em cada mensagem para cada um dos endereços de nodo. Para comunicação são utilizados dois sockets: um para recebimento de mensagens e outro para envios. Cada socket tem o seu próprio endereço, dessa forma um nodo tem dois sockets com diferentes portas. Na figura 1, podemos ver o esquema de um nodo.

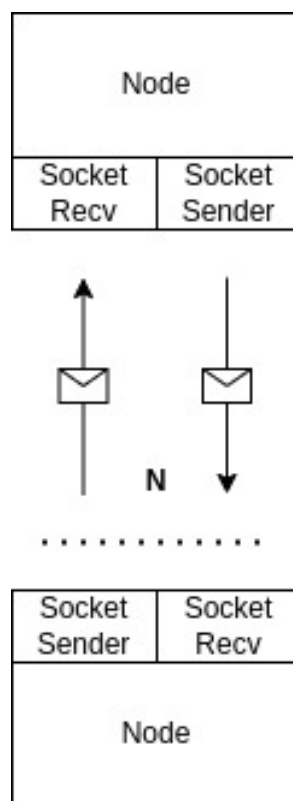


Figura 1 – Estrutura de um nodo

2.2 Mensagem

Internamente as mensagens são compostas por quatro atributos: dados a serem enviados, id da mensagem, id do processo e endereço do processo. Cada mensagem é

transferido na forma de bytes, onde é realizado um parse dela. Em nossa biblioteca, enviamos uma string com o seguinte corpo: DataTxt # MsgID # ProcessID # ProcessAddress.

2.2.1 Recebimento de mensagens

O método responsável por receber as mensagens verifica se a mensagem recebida está no buffer do processo. Se sim, ignora a mensagem. Do contrário, verifica se ela está no buffer de entrada e se já pode ser entregue. Se a mensagem está em seu estado válido, então a mensagem é entregue.

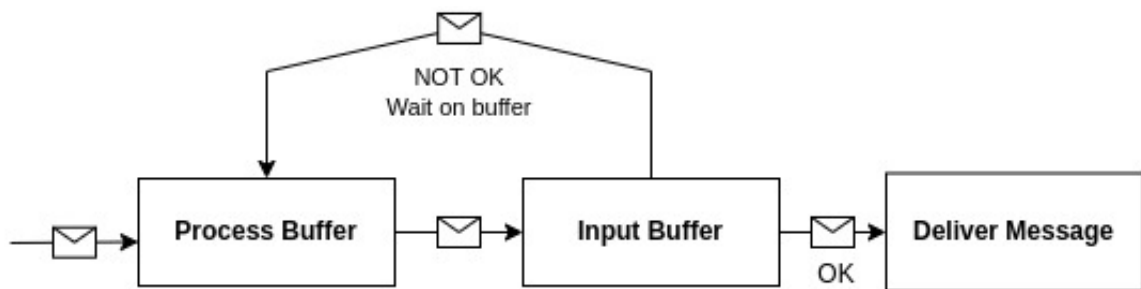


Figura 2 – Estrutura do recebimento de mensagens

2.2.2 Envio de mensagens

Para enviar as mensagens é necessário adicionar o ID em cada uma delas antes do envio. Sendo assim, foi utilizado um buffer de saída que armazena os IDs de cada mensagem para cada um dos comunicantes.

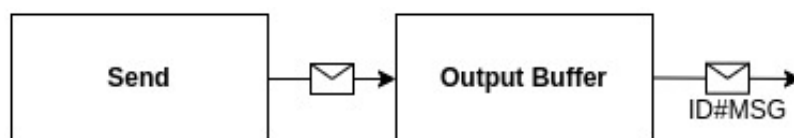


Figura 3 – Estrutura do envio de mensagens

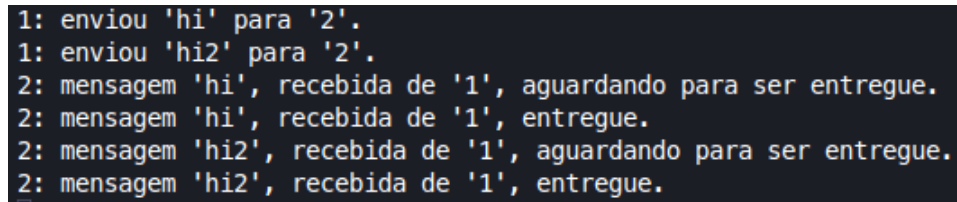
2.3 Algoritmo Token Ring

Nesse algoritmo é construído um anel lógico com os processos comunicantes. Em sua inicialização o processo 0 recebe um token. Ele deve circular pelo anel passando de um processo k para o $k + 1$. Quando o processo recebe o token ele pode executar a região crítica (uma única vez) e, após sair, passa o token para o próximo processo. Se o processo não desejar entrar na região crítica, deve apenas passar o token para o próximo processo.

2.4 Execução

2.5 Resultados

Para verificar o funcionamento das mensagens, realizamos um teste onde o nodo 1 envia duas mensagens para o nodo 2. Na figura 4, é mostrado a saída indicando que a mensagem foi entregue corretamente.



```
1: enviou 'hi' para '2'.  
1: enviou 'hi2' para '2'.  
2: mensagem 'hi', recebida de '1', aguardando para ser entregue.  
2: mensagem 'hi', recebida de '1', entregue.  
2: mensagem 'hi2', recebida de '1', aguardando para ser entregue.  
2: mensagem 'hi2', recebida de '1', entregue.
```

Figura 4 – Envio de mensagens sendo entregue entre os processos.

2.6 Conclusão

2.7 Referências

ANEXO A – Biblioteca

A.1 Código

ANEXO B – Algoritmo