INSTITUTO TECNOLÓGICO DE AERONÁUTICA
CES-27

Relatório Laboratório 02

Professores:
Celso Hirata
Juliana de Melo Bezerra

Aluna:
Raíssa Batista de Miranda Pimentel

São José dos Campos, 5 de Outubro de 2020

Conteúdo

1 Implementação

Para a implementação do algoritmo *Ricart-Agrawala* de exclusão mútua, o processo executado recebe como argumentos seu identificador (que é único e crescente, iniciando em 1) e um vetor de portas que definem os processos que irão interagir entre si. O processo executa da seguinte maneira:

- Caso seja digitado o identificador do processo corrente, o seu relógio lógico é aumentado em 1. Além disso, é impresso o que foi digitado no terminal e o novo valor do relógio lógico.
- Caso o processo receba uma mensagem de outro processo contendo um valor de relógio lógico, se o valor recebido é maior do que seu relógio lógico corrente, o seu novo valor do relógio lógico é o recebido adicionado de 1. Caso contrário, o valor do relógio lógico é incrementado em 1. Além disso, é impresso o que foi recebido e o novo valor do relógio lógico.
- Caso seja digitado um "x" no terminal e o processo corrente não esteja usando a CS (*Critical Section*) ou solicitando-a, ele deve solicitar acesso à CS, usá-la e liberá-la de acordo com o algoritmo *Ricart-Agrawala*.
- Ao solicitar o uso da CS, seu relógio lógico é incrementado e o processo envia mensagens para todos os outros processos contendo um texto de "request" e o seu relógio lógico atualizado (simulando um *multicast*). O processo fica aguardando, de modo não bloqueante (por causa do uso de uma `goroutine` chamada `doServerJob`), todos os processos enviarem uma mensagem de "reply" para ele para assim ele poder entrar na CS.
- Caso o processo receba uma mensagem de "request", se ele está na CS ou solicitando-a e seu tempo de solicitação for menor do que a da mensagem de "request" recebida (em caso de empate de tempo, o desempate é feito no identificador do processo), essa mensagem é inserida numa fila. Caso contrário, seu relógio lógico é incrementado e é enviada uma mensagem de "reply" para o processo de origem da mensagem.
- O uso da CS corresponde a enviar uma mensagem, por meio da porta `:10001`, ao processo `SharedResource.go` contendo o relógio lógico corrente (que é incrementado antes de enviar a mensagem), um texto de "Oi" e o `id` do processo e, após esse procedimento, dormir por 5 segundos. Antes de enviar a mensagem para a CS, o processo escreve na tela "Entrei na CS" e, depois de dormir, o processo escreve na tela "Sai da CS" e envia mensagens de "reply" a quem estava na fila, incrementando o relógio lógico caso a fila não esteja vazia (simulando um envio simultâneo de várias mensagens de "reply"). Esse processamento é feito numa `goroutine` chamada `enterCS()`, de modo que o processo possa receber mensagens enquanto está na CS.
- Caso seja digitado um "x" no terminal e o processo corrente esteja usando a CS ou solicitando-a, imprime-se na tela "x ignorado".

2 Testes

2.1 Teste 1

Para demonstrar a robustez do código implementado, o programa foi testado ao executar o recurso compartilhado, em um terminal, e três processos, em três terminais distintos, para as portas `:10002`, `:10003` e `:10004`, seguindo o diagrama da Figura 1, em que o processo 1 pede a CS, a utiliza, e depois disso o processo 3 pede a CS e a utiliza. O recurso compartilhado está representado com a sigla "SR" e estados que estão na CS estão representados com a sigla "CS".

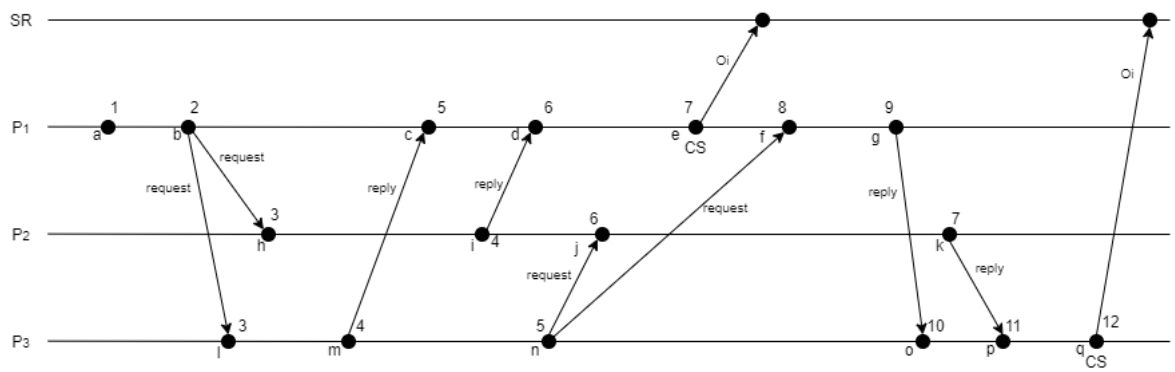

Figura 1: Diagrama do teste 1.

Vale ressaltar que a ordem das mensagens de "reply" podem ser trocadas caso o teste seja executado em outra máquina, uma vez que não há como controlar a ordem de recebimento. Nesse teste, por exemplo, a ordem dos estados `c` e `d` e dos estados `o` e `p` podem ser trocadas. Porém, os relógios lógicos finais e os resultados de uso da CS devem ser os mesmos.

Realizando o teste segundo esse diagrama, foi digitado 1 e depois "x" no terminal do processo 1, e, após ele imprimir "Sai da CS", foi digitado "x" no terminal do processo 3. O resultado obtido nos terminais está contido na Figura 2.

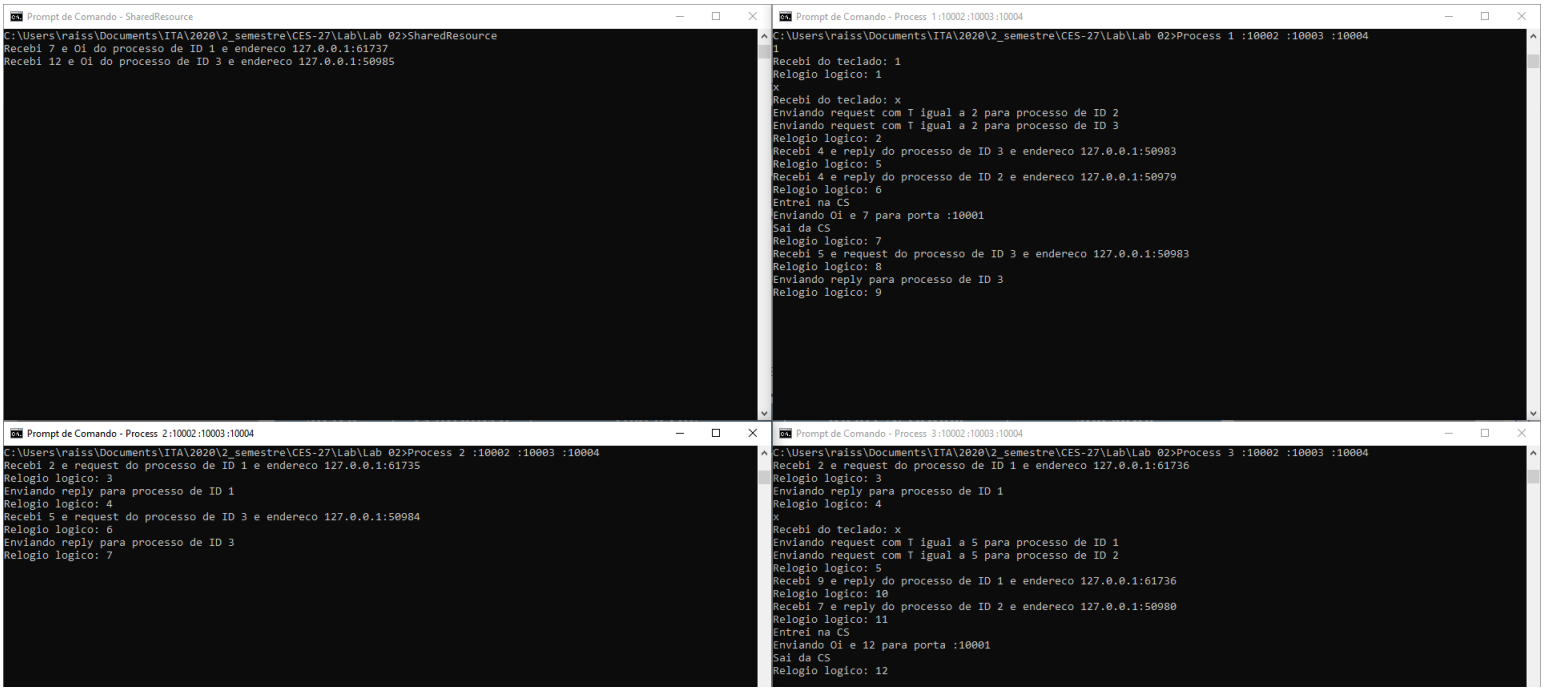

Figura 2: Resultado do teste 1.

De acordo com a Figura 2, pode-se afirmar que os resultados obtidos são satisfatórios, uma vez que é possível notar que as transições de cada processo respeitam o diagrama e as regras do relógio lógico de Lamport. Esse exemplo mostra também que o algoritmo implementado permite que um processo solicite a CS, use-a e a libere de forma que outro processo possa utilizá-la, ou seja, um processo não fica utilizando a CS para sempre, impedindo outras de acessarem.

2.2 Teste 2

Novamente para demonstrar a robustez do código implementado, o programa foi testado ao executar o recurso compartilhado, em um terminal, e cinco processos, em cinco terminais distintos, para as portas `:10002`, `:10003`, `:10004`, `:10005` e `:10006`, seguindo o diagrama da Figura 3, em que o processo 3 pede a CS, e, enquanto ele está utilizando-a, os processos 2 e 1 solicitam a CS, nesta ordem. O recurso compartilhado está representado com a sigla "SR" e estados que estão na CS estão representados com a sigla "CS".

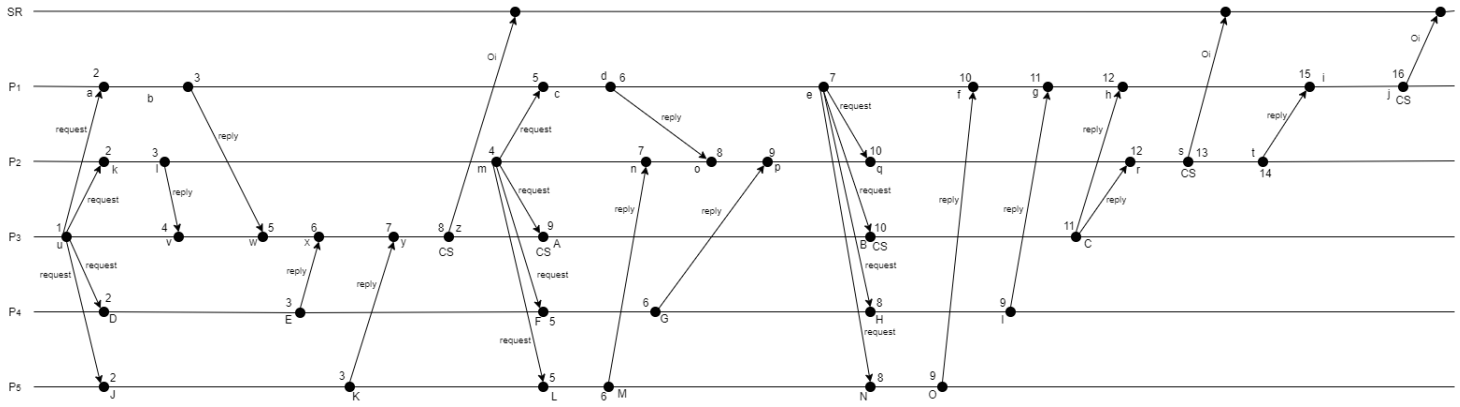

Figura 3: Diagrama do teste 2.

Vale ressaltar que a ordem das mensagens de "reply" podem ser trocadas caso o teste seja executado em outra máquina, uma vez que não há como controlar a ordem de recebimento. Nesse teste, por exemplo, a ordem dos estados `v`, `w`, `x` e `y`, dos estados `n`, `o` e `p`, dos estados `f`, `g` e `h` podem ser trocadas. Porém, os relógios lógicos finais e os resultados de uso da CS devem ser os mesmos.

Realizando o teste segundo esse diagrama, foi digitado "x" nos terminais no processo 1, 2 e 3 na seguinte ordem: terminais dos processos 3, 2 e 1. Assim, o caractere só foi processado pelo processo 2 após receber a mensagem de "request" do processo 3 e o caractere só foi processado pelo processo 1 após receber as mensagens de "request" dos processos 3 e 2. Assim, o envio do caractere "x" nesses processos só ocorre depois da impressão do recebimento e envio de algumas mensagens. O resultado obtido nos terminais está contido nas Figuras 4 e 5.

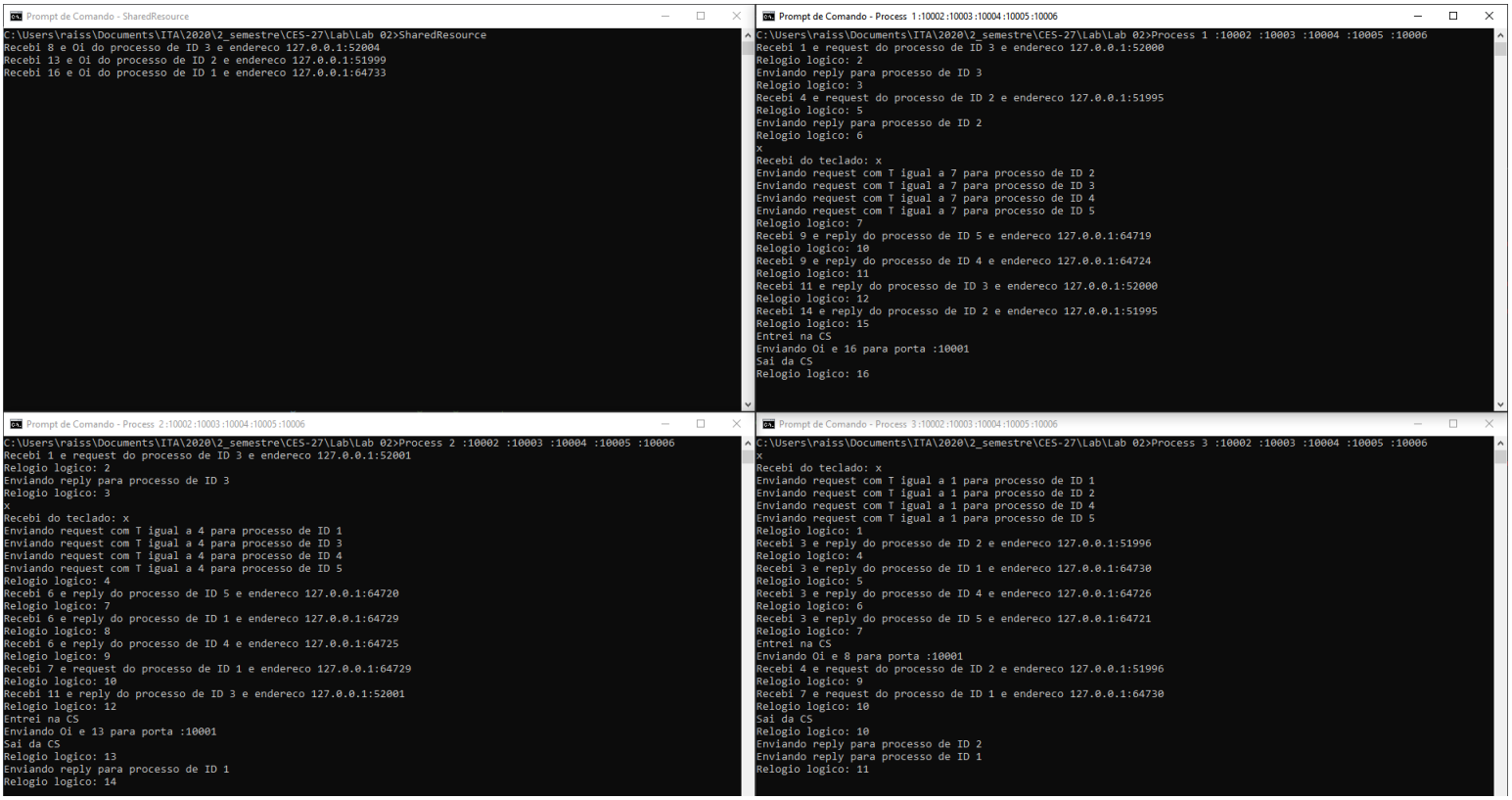

Figura 4: Resultado parcial do teste 2.

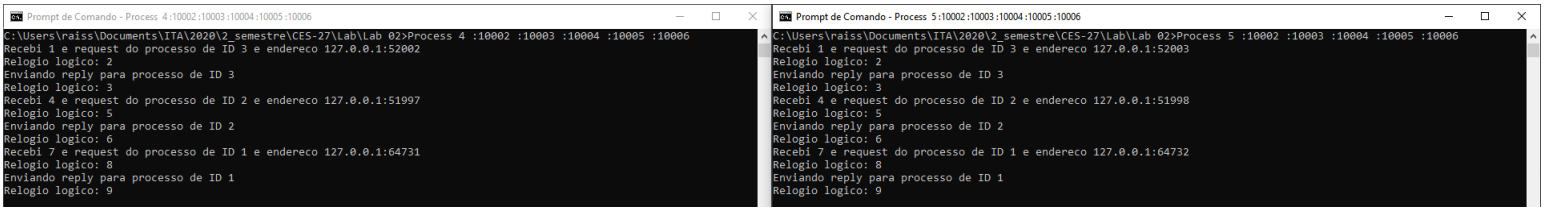

Figura 5: Resultado parcial do teste 2.

De acordo com as Figuras 4 e 5, pode-se afirmar que os resultados obtidos são satisfatórios, uma vez que é possível notar que as transições de cada processo respeitam o diagrama e as regras do relógio lógico de Lamport. Esse exemplo mostra também que o algoritmo implementado atende às propriedades de *safety*, *liveness* e *ordering*, já que nenhum processo acessou a CS ao mesmo tempo (mesmo solicitando em instantes próximos), todo processo que solicitou acesso à CS pôde acessá-la e o acesso foi ordenado pelo valor do relógio lógico em que se enviou a mensagem de "request", ou seja, de solicitação da CS. Além disso, esse exemplo mostra que, mesmo utilizando a CS, o processo pode receber mensagens e, caso sejam de "request", ele as enfileira e, assim que ele sai da CS, envia mensagens de "reply" para os processos que enviaram essas mensagens de "request".

3 Conclusão

Após realizados os testes sobre os códigos implementados nesse laboratório, verificou-se que todos eles produziram os resultados esperados, e, portanto, acredita-se que a solução proposta está funcionando adequadamente para implementar o algoritmo *Ricart-Agrawala*.