Sistemas Distribuídos – COS470

2019/1

Trabalho Prático 3

1 Objetivo

O objetivo deste trabalho é projetar e implementar o algoritmo do valentão (*bully algorithm*) que é um algoritmo distribuído para eleição de líder. A implementação deve ser *multi-threaded* e deve usar recursos de sincronização (semáforos, monitores, etc) podendo ser escrita em C, C++, Java, ou Go, esta última uma linguagem de programação aberta projetada para sistemas distribuídos e utilizada pela Google.

Além da implementação, você deve testar e avaliar seu programa. Você deve preparar um relatório, com no máximo 6 páginas, com as decisões de projeto e implementação das funcionalidades especificadas, assim como a avaliação dos estudos de caso. O relatório teve ter uma URL para seu código. Por fim, você deve preparar uma apresentação de no máximo 10 minutos sobre seu trabalho (como no relatório), a ser realizada no horário da aula. O trabalho pode ser feito individualmente ou em dupla.

2 Funcionamento e *threads*

Cada processo do sistema distribuído executa exatamente o mesmo programa. Ou seja, você deve construir apenas um programa que possui toda a funcionalidade e que será executado diversas vezes (na mesma máquina ou em máquinas diferentes). Cada processo deve ter um identificador que será usado para enviar mensagens para os processos. Além disso, cada processo deve ter um número único, que será usado para definir o líder no caso de eleição.

Seu programa deve ter ao menos três *threads*:

1. Thread responsável por gerenciar interface com usuário. Ou seja, este thread lê comandos do teclado (ou interface gráfica) e toma as respectivas ações.

2. Thread responsável por receber mensagens e tomar ações. Esta thread recebe mensagens de outros processos e toma as respectivas acções.

3. Thread responsável em detectar a presença do líder. Esta thread periodicamente verifica se o líder está presente e toma as respectivas ações. Esta verificação periódica dever ser feita aguardando um tempo, usando a função *sleep()*.

Repare que as threads possivelmente irão compartilhar estruturas de dados, que devem ser protegidas adequadamente. Por exemplo, cada processo deve ter uma variável que guarda o identificador do atual líder do sistema.

Todas as threads devem ser bloqueantes, ou seja, nenhuma delas pode executar em *busy wait*.

3 Mensagens

Seu programa deve ter ao menos cinco tipos de mensagens:

1. `ELEICAO`: Mensagem que indica o início de uma nova eleição. O corpo desta mensagem deve conter o identificador do processo que gerou a mensagem e seu respectivo valor para fins de eleição.

2. `OK`: Mensagem enviada ao processo P que iniciou a eleição por um processo Q que tenha valor maior que P. A mensagem deve conter o identificador do processo Q.

3. `LIDER`: Mensagem enviada por um processo P que iniciou a eleição depois de um tempo Δ (parâmetro). A menagem deve conter o identificador do processo O.

4. `VIVO`: Mensagem enviada por um processo P a outro processo Q qualquer.

5. `VIVO_OK`: Mensagem enviada por um processo Q ao receber `VIVO` caso Q seja líder no momento.

Para facilitar, todas as mensagens devem ser strings com tamanho fixo dado por F bytes e separadores. Os separadores servem para indicar diferentes campos dentro da mensagem, tais como o código de mensagem e o identificador do processo. Por exemplo, se $F = 10$ bytes, a mensagem `ELEICAO` poderia ter o seguinte formado `1|3|19|000`, o que neste caso indica que o separador é o caractere `|` e a mensagem `ELEICAO` tem código 1 (cada mensagem deve ter um código), que o processo que gerou a mensagem tem identificador 3, e que seu valor para fins de eleição é 19. Repare que os zeros no final servem apenas para garantir que a mensagem sempre tenha F bytes.

4 Interface e ações

A interface do processo deve responder ao seguintes comandos, vindo do teclado ou da interface gráfica:

1. Determinar se o atual líder está operacional: Ao ler uma tecla o processo aciona o processo de verifica se o atual líder está respondendo.

2. Falhar o processo: Ao ler uma tecla o processo, o processo emula uma falha deixando de enviar e responder a qualquer mensagem. Ou seja, ele recebe as mensagens, mas as ignora.

3. Recuperar o processo: Ao ler uma tecla o processo, o processo emula seu regresso ao sistema distribuído, voltando a enviar e responder mensagens.

4. Gerar estatísticas: Ao ler uma tecla o processo, o processo deve imprimir algumas estatísticas, tais como o atual líder do sistema e número de mensagens de cada tipo (mais detalhes abaixo).

Para facilitar, assuma que o processo em falha acompanha as eleições para líder e sabe quem é o líder. Ou seja, ao receber a mensagem `LIDER` o processo registra quem é o atual líder do sistema (mas sem responder a nenhuma mensagem) Ao recuperar, o processo passa a usar este líder.

Ao detectar uma falha, seja de forma pro-ativa ou através do mecanismo periódico, um processo deve iniciar a eleição de líder. Você pode implementar um mecanismo para tornar esta condição de corrida favorável, ou seja, para que o sistema tenha um melhor desempenho.

5 Medição e avaliação

Cada processo deve contar quantas mensagens de cada tipo foram enviadas e recebidas desde o início do processo (um contador para mensagens enviadas e outro para mensagens recebidas). Estes valores devem ser impressos ao comando do usuário. Execute seu programa com diferentes parâmetros para avaliar os seguintes cenários:

1. Determine o número médio de mensagens enviadas por eleição. Para calcular este número, emule uma falha do líder (e depois da eleição, a sua recuperação) 10 vezes, e faça a média. Repare que as falhas devem alternar entre os dois processos que serão líder.

2. Determine o número médio de mensagens enviadas por eleições em falhas consecutivas. Para calcular este número, emule uma falha no líder do sistema, e depois uma falha do próximo líder, e assim por diante, sem que nenhum desses processos se recupere, até termos apenas um processo. Para obter uma média, realize este teste 10 vezes e faça a média.

Para estas métricas, apenas mensagens relacionadas a eleição devem ser consideradas.

As duas métricas acima devem ser calculadas para sistemas com $n = 2, 4, 8, 16$ processos. Trace um gráfico dos valores do número médio de mensagens em função de n (duas figuras, uma para cada métrica definida acima).