INF01151 – SISTEMAS OPERACIONAIS II N

Prof. Weverton Cordeiro

Universidade Federal do Rio Grande do Sul

Fernando Vieira Utzig - 00324861 Thiago Leonel Rancan Bischoff - 00324856

Vitor Godoy de Fraga - 00308623

Trabalho prático parte 2

Replicação e eleição de líder

Descrição do ambiente de testes

Para testar o sistema, os participantes do grupo optaram por usar as máquinas pessoais e se conectar na mesma rede através do software do *Hamachi*, além de eventuais idas ao laboratório da informática para usar as estações. A seguir, a lista com a descrição dos ambientes utilizados:

Versão do Sistema Operacional: Linux 64 bits, GNOME Version 42.5
Distribuição: Ubuntu 22.04.2 LTS
Configuração: Modelo de hardware LG Eletronics N450-P.BESSP1, 4,0 GB Memória RAM, Processador Intel Core i5-3210M CPU @ 2.50GHz x 4, SSD 120GB
Compilador: gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

Nessa estação, foi emulado o Linux com 4GB de memória RAM em um Windows de 16GB de RAM através do software VirtualBox, criando uma máquina virtual de sistema.
Versão do Sistema Operacional: Linux 64 bits, GNOME Version 42.9
Distribuição: Ubuntu 22.04.3 LTS

Configuração: Máquina virtual configurada com 4,0 GB Memória RAM, Intel Core i7-9750H CPU 2.60GHz x 4

Compilador: gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

Funcionamento do algoritmo de eleição de líder implementado

O algoritmo de eleição implementado foi o do valentão (bully). O funcionamento começa quando um participante detecta que a estação líder não está respondendo a mensagem de descoberta, rodando a função *election*. Essa função envia uma mensagem com o conteúdo "ELECTION" para todos os participantes que têm o ID do processo maior que o próprio. Para isso, modificamos a estrutura dos participantes para salvar também o dado do ID do processo que está executando o serviço. Se a função achou um processo com ID maior que o próprio e mandou a mensagem, retorna 0, significando que não é o líder. Caso contrário, retorna 1, sinalizando que é o processo com maior ID e, consequentemente, o líder eleito.

Os participantes passaram a aguardar também uma mensagem de election. Se uma estação recebe uma mensagem de election, responde com uma mensagem de conteúdo "ACK" e roda a função de eleição.

Após acontecer a eleição de um líder através do retorno 1 da função *election,* a estação manager envia uma mensagem para todos os participantes do grupo sinalizando que foi eleita e executa as threads relativas ao manager do sleep management.

- Justificativa da escolha:

O algoritmo de eleição escolhido foi o do valentão, pois ele assume entrega confiável de mensagens, usando timeouts para detectar falhas e mensagens de confirmação (ACK)

Implementação da replicação passiva na aplicação

Para esse mecanismo utilizamos o método de replicação passiva. Quando há alguma falha no manager, que o leva ao encerramento dos seus processos, uma eleição é iniciada para eleger um novo manager, este eleito seria o RM primário. Essa falha é identificada pois os processos participantes utilizam a thread de monitoramento para identificar a perda do contato com o manager. Assim que um novo líder for eleito, ele necessita de uma réplica das tabelas e dos processos que estavam sendo administrados pelo manager anterior.

Para o envio da réplica da tabela, foi utilizado um processo já existente, o processo de monitoramento. Dentro deste processo, o envio da tabela, juntamente com a sua versão, é feito pela função "monitorParticipant".

Também, quando eleito um novo manager, é necessário que o participante eleito encerre as suas tarefas e inicie as tarefas de manager. Para isso foi criada uma função

“changeToManager” e uma “changeToParticipant”, elas são responsáveis pelo encerramento e execução das threads corretas.

- Desafios encontrados:

Um dos desafios encontrados foi conseguir manter as tabelas sempre atualizadas entre os participantes. Para resolver isto, foi usado um sistema de versionamento de tabela, ou seja, relógios lógicos.

Outro desafio encontrado foi a realização de testes, pois, para realizar testes com mais participantes, era necessário mais computadores. Como solução, utilizamos o software Hamachi para os integrantes do grupo conectarem na mesma rede.

Descrição dos problemas que a equipe encontrou durante a implementação:

Um dos problemas encontrados foi a comunicação entre os processos participantes, que se deu necessária para o algoritmo de eleição. Agora, além da comunicação entre manager e participante, também foi preciso implementar uma comunicação entre participante e participante. Para isto, utilizamos mais uma thread responsável por monitorar mensagens do tipo “ELECTION” vindas de outros participantes.

Outro problema, foi a detecção de falha do manager. Como solução, utilizamos a thread “ParticipantMonitoringThread” para analisar se o manager está respondendo ou não. Utilizamos também um sistema de 3 tentativas para checar se o que foi perdido foi apenas o pacote enviado, ou de fato, a conexão com o manager. Caso o contato realmente tenha sido perdido, uma eleição é começada.

Referências

[1] Sen, Siddhartha, Jacob R. Lorch, Richard Hughes, Carlos Garcia Jurado Suarez, Brian Zill, Weverton Cordeiro, and Jitendra Padhye. "Don't Lose Sleep Over Availability: The GreenUp Decentralized Wakeup Service." In 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pp. 211-224. 2012.