Programação Distribuída - Trabalho 2

O segundo trabalho da disciplina consiste em desenvolver um programa distribuído que implemente o algoritmo de Berkeley para sincronização de diversos processos executando em nodos independentes. O programa deve receber como entrada uma configuração com os seguintes parâmetros:

- *id* é um número inteiro que identifica o processo;
- *host* é o hostname ou endereço IP da máquina (nodo) que executa o processo;
- *port* é o número da porta que o processo vai escutar;
- *time* é a hora com a qual o processo será iniciado;
- *ptime* é o tempo de processamento no processo (em ms).
- *adelay* é o atraso (adicional) ao tempo de comunicação da rede (para simular uma rede com maior atraso)

Descrição do algoritmo:

O algoritmo de Berkeley considera um grupo de processos distribuídos por nodos trabalhando de forma isolada, ou seja, nenhum processo possui uma referência externa de tempo. Um dos processos do grupo (mestre) será responsável por coordenar a atualização dos relógios de outros processos (escravos), sendo essa atualização realizada com um período predeterminado (a cada 30 segundos, a cada minuto ou a cada hora, por exemplo).

A cada iteração do algoritmo o mestre realiza o *polling* dos escravos, enviando uma mensagem *multicast* ou múltiplas mensagens *unicast* para os mesmos e realizando a coleta da referência local de tempo de cada um. Além disso, é possível agora para o mestre obter o RTT de cada escravo, sendo que essa informação deverá ser utilizada posteriormente para que seja compensado o atraso da rede na atualização de cada processo. O mestre deverá calcular a média dos tempos (incluindo-se nessa média), e deverá enviar o tempo correto a cada escravo, incluindo o atraso de um caminho (*one way delay*). Além disso, o mestre deverá ignorar nessa média tempos com um desvio maior que um valor predeterminado da média ou mediana inicial (10 segundos, por exemplo).

O relógio deverá ser mantido por software, ou seja, inicialmente o relógio do sistema poderá ser consultado para inicialização, sendo posteriormente gerenciado por sua aplicação.

Como alternativa ao uso do parâmetro *adelay*, o atraso relacionado ao tempo de comunicação da rede poderá ser realizado por meio da configuração das interfaces de rede com o uso da ferramenta *Traffic Control*[1]. A ferramenta pode ser utilizada com múltiplas VMs ou com o *Core Emulator*.

Saída do programa:

Cada processo deverá produzir individualmente sua saída, incluindo eventos de recebimento de mensagens, tempo de recebimento e ações realizadas, além de eventos de atualização de tempo. O mestre deverá apresentar os cálculos realizados, incluindo processos não considerados na média, RTT, valor da média e fatores de correção aplicados.

Entrega e apresentação:

O trabalho deve ser realizado em duplas ou trios e apresentado em laboratório em um ambiente distribuído. Qualquer linguagem de programação poderá ser utilizada para o desenvolvimento. Para a entrega, é esperado que apenas um dos integrantes envie pelo Moodle um arquivo *.tar.gz*, contendo o código fonte da implementação.

[1] https://netbeez.net/blog/how-to-use-the-linux-traffic-control/