Algoritmo de Berkeley

O algoritmo de Berkeley permite sincronizar diversos computadores em um sistema distribuído. Para o entendimento do problema e da solução apresentados originalmente, veja o trabalho publicado "*The accuracy of the clock synchronization achieved by TEMPO in Berkeley UNIX 4.3BSD*", em https://doi.org/10.1109/32.29484 . E, se precisar de mais alguns exemplos, veja: https://lasarojc.github.io/ds_notes/time/physical/#algoritmo-de-berkeley

Além disso, esse algoritmo também pode ser visto nos capítulos dos livros adotados na disciplina: para isso, veja o Plano de Ensino disponibilizado.

Considere um sistema distribuído que tenha:

- mínimo de 3 processos que se comunicam para uma sincronização entre eles;
- o relógio de cada processo, no início do seu funcionamento, deve ser recuperado do relógio do sistema;
- a cada ciclo, o relógio local de cada processo deve ser modificado aleatoriamente em ±2s;
- cada processo tem o controle do seu próprio ciclo que acontece de forma aleatória (na pior das hipóteses, use ciclos de 10s).

Por outro lado, esse sistema decide quem é um Responsável por receber mensagens dos demais processos. Os demais processos, aleatoriamente, escolhem um valor (como, por exemplo, uma temperatura coletada por sensores) e envia esse dado para o responsável. Nesse envio, há também a hora local de cada processo que indica o tempo em que a coleta aconteceu.

Porém, se o Responsável por receber esses dados perceber que o registro da hora na mensagem estiver distante 1s da sua própria hora local, o Responsável inicia um clico de sincronização de relógio usando o algoritmo de Berkeley.

Pontos de atenção:

- não precisa considerar falha de mensagens;
- o processo de sincronização sempre acontecerá independentemente da concorrência de mensagens enviadas de mais de um processo;
- faça o registro/log das ações dos processos: início da sincronização: hora dos processos antes da sincronização; hora depois da sincronização; etc;
- se achar mais prático, pode limitar a execução de sincronização dos ciclos (por exemplo, 10 ciclos) – atenção com o término inesperado dos processos;
- seu código deve ser desenvolvido em Java, Python ou C/C++ - *preferencialmente*, nas 3 linguagens;
- é obrigatório o uso de computadores diferentes para visualizar o teste em ambientes reais – pode-se utilizar também máquinas virtuais no seu provedor de nuvem predileto ou até mesmo loca.