Guião N.º 5

# Sockets UDP 2 (em Java)

Sistemas Distribuídos

Ricardo Costa
rcosta@estg.ipp.pt

ESTGF | POLITÉCNICO DO PORTO

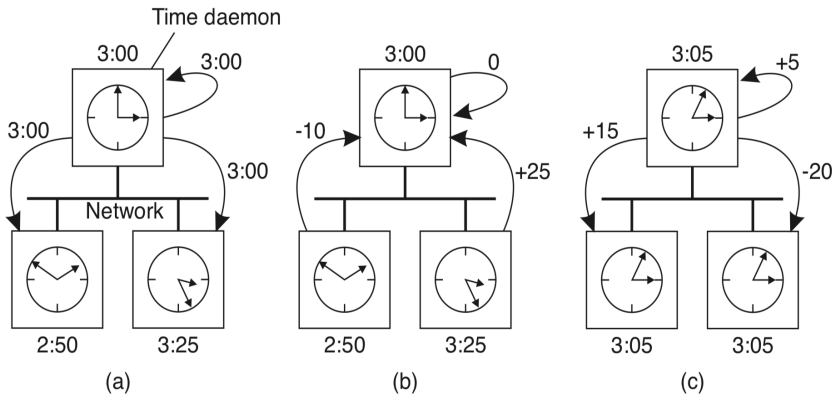
Novembro 2017

## 1 Sincronização de relógios

Embora cada computador tenha um relógio local é muito provável que o valor dos relógios dos diferentes computadores dum sistema distribuído sejam distintos. Algumas razães para os múltiplos valores incluem: 1) os relógios foram inicializados com horas distintas, 2) as frequências reais dos relógios são diferentes das nominais, conduzindo a atrasos/avanços.

O algoritmo de Berkeley permite a sincronização de relógios em sistemas distribuídos assumindo que nenhuma máquina da rede tem um relógio exacto.

- Uma máquina é eleita master usando um dos vários algoritmos de eleição existentes;
- O master questiona periodicamente todas as máquinas sobre o seu tempo local;
- Determina o Round-Trip-Time (RTT) das mensagens;
- Calcula uma média de todos os tempos recebidos, ignorando valores que distem mais do quem um intervalo pré-definido em relação á média;
- Informa cada uma das máquinas do valor do ajuste que devem proceder em relação ao seu tempo local.




**a)** O master pede a todas as máquinas o seu tempo local;

**b)** As máquinas respondem

**c)** Envia a todas as máquinas o ajuste em relação ao seu tempo local

# 2 Exercícios

Pretende-se que implemente uma versão simplificada do algoritmo de Berkeley.

1. Crie um servidor UDP que envia para um grupo de multicast um datagrama pedindo o tempo local de cada um dos membros desse grupo e recebe as respostas.

   Pode assumir que existe um número conhecido de clientes para recolher as respostas. Mesmo assim, deverá definifir um tempo máximo para a recepção de um datagrama para evitar que o servidor bloqueie indefinidamente à espera de um datagrama perdido.

2. Implemente um cliente UDP que responde ao pedido do servidor com o seu tempo local.

3. Altere o servidor de forma a que seja possível calcular o Round-Trip-Time de uma série de datagramas.

   O servidor deverá enviar uma série de datagramas UDP e contabilizar quanto tempo as respostas a esses datagramas demoram a ser recebidas.

   O programa deverá também indicar a percentagem de perda de pacotes.

Sugestão: pode usar os métodos *currentTimeMillis()* ou *nanoTime()* da classe System:

```
long startTime = System.nanoTime();
// ... Envio e recepção de datagramas ...
long estimatedTime = System.nanoTime() - startTime;
```

4. Altere o servidor de modo a determinar o ajustamento de cada um dos clientes e a comunicar a cada um deles.

5. Altere o cliente de modo a sincronizar o seu relógio de acordo com a informação recebida. Para evitar alterar o tempo do sistema crie uma classe que represente o tempo local.

6. Altere o servidor de modo a ele próprio também responder ao pedido. Para tal, o servidor deve passar a ter duas *threads*. Esta segunda *thread* funciona como um cliente.