Sistemas Distribuídos 2014/2015 **MIEEC**
Prova de Programação **13 de Janeiro de 2015**

Duração: 150 minutos
Consulta: documentação instalada no computador
Código público: **YQA435** (`http://sigex.fe.up.pt/`)

---

1 Introdução

A prova consiste na resolução em Java ou em C de funcionalidades necessárias para a implementação do algoritmo do "bully", de Garcia-Molina, para eleição de um *leader*:

1. Deteção de falha, com cotação máxima de 16 valores.
2. Eleição do *leader*, com cotação máxima de 20 valores.
    - Cotação máxima de 18 valores se usar comunicação unicast em vez de multicast.

2 Notas Preliminares

Dica 1 - Para medir intervalos de tempo pode usar `Thread.sleep()/sleep()` (ver `man 3 sleep`).

Dica 2 - Para medir o *timeout*, pode usar `SO_RCVTIMEOUT` (ver `setSoTimeout()/setsockopt()`, e, se optar por C, `man 7 socket`).

Dica 3 - Use `System.currentTimeMillis()/gettimeofday()` para retomar uma temporização interrompida por algum evento.

Implementação em C Os nomes dos programas apresentados abaixo seguem as convenções usadas em Java. Se optar por uma implementação em C deverá seguir as convenções desta linguagem. Assim, neste caso, os nomes são idênticos aos especificados, exceto a primeira letra deverá ser minúscula e não maiúscula. Os argumentos da linha de comando são idênticos aos de Java

Se optarem pela implementação em C deverão ainda submeter uma **`Makefile`** que permita compilar os seus programas invocando `make`. Alternativamente, poderão submeter um ficheiro em ASCII de nome `compilation.txt` com os comandos para compilação dos vossos programas.

3 Deteção de Falha

Uma das funcionalidades requeridas pelo "bully algorithm" é a deteção de falhas. Nesta alínea deverá implementar um protocolo baseado no envio periódico de mensagens de **"ping"**.

I.e. um processo, **monitor**, deverá enviar uma mensagem de "ping" periodicamente a outro processo, **target**. Este deverá enviar uma resposta por cada mensagem de *"ping"* recebida.

Implemente este protocolo usando **TCP** e a linguagem que preferir (Java ou C).

A ausência duma resposta ou a interrupção da conexão deverão ser consideradas falhas do *target*.

Quer o *monitor* quer o *target* deverão imprimir na saída padrão (`System.out/stdout`) mensagens indicando: 1) o envio duma mensagem ; 2) a receção duma mensagem. As mensagens impressas deverão incluir as mensagens recebidas/enviadas ou seus sumários. Compete-lhe a si definir o formato das mensagens trocadas entre cliente e servidor.

Invocação dos programas em Java O Target deverá ser invocado da seguinte maneira:

```
java Target <port>
```

onde

`<port>` é o porto do *socket* onde o *target* receberá os pedidos de *ping*, e que usará para enviar as respostas.

O *Monitor* deverá ser invocado da seguinte maneira:

`java Monitor <addr> <port> <period> <timeout>`

onde

`<addr>` é o endereço IPv4 do *target*;
`<port>` é o porto do *socket* usado pelo *target*;
`<period>` é o periodo das mensagens de *ping*;
`<timeout>` é o *timeout* para deteção de falha do *target*.

4 Eleição

Nesta alínea deverá implementar uma variante do protocolo de eleição do Bully com comunicação multicast. (Se usar comunicação UDP unicast, a cotação máxima será 90%.)

A figura ilustra o algoritmo que difere do original na medida em que o multicast da mensagem `ELECTION` permite eliminar as mensagens de `BACKOFF` e de `HALT`. I.e., um processo que receba uma mensagem `ELECTION` de outro processo com um identificador inferior deverá cancelar a execução de qualquer protocolo de eleição que tenha iniciado. Por outro lado, se não estiver em modo de eleição e receber uma mensagem `ELECTION` deverá passar para o modo de eleição.

Essencialmente, o seu programa deverá realizar uma execução do algoritmo. Essa execução deverá ser iniciada após: a expiração dum *timeout* inicial, ou da receção duma mensagem `ELECTION`. Para evitar uma "avalancha" de execuções simultâneas, a duração do *timeout* inicial deverá estar uniformemente distribuída no intervalo $[0, \texttt{<id>} \times \texttt{<delay>}]$, onde `<id>` e `<delay>` são argumentos da linha de comandos (ver abaixo). Para gerar números aleatórios pode usar `java.util.Random/random()` (ver man 3 random()). Para permitir seguir a execução do protocolo, o seu programa deverá imprimir na saída *standard* todas as mensagens enviadas e/ou recebidas.

Dica- Procure implementar o protocolo como uma máquina de estados.

Invocação do programa em Java O programa deverá ser invocado:

`java Bully <addr> <port> <id> <T> <delay>`

onde

`<addr>` é o endereço IPv4 multicast a subscrever pelo processo;
`<port>` é o porto do *socket* do grupo multicast a usar pelo processo;
`<id>` é o identificador do processo (para determinar o *leader*)
`<T>` é um majorante do atraso de comunicação entre 2 processos em **ms**.
`<delay>` é o tempo (em **ms**) que deverá ser usado, juntamente com `<id>`, para determinar temporizações aleatórias (ver acima).

5 Documentação e Submissão

A documentação da API de Java pode ser encontrada no seguinte URL: `file:///opt/java-docs/`

Para submeter a prova deverá omprimir num ficheiro `.zip` único todos os ficheiros com a sua solução, i.e. ficheiros com o código (caso tenha desenvolvido o seu programa em C deverá também incluir os ficheiros com os comandos de compilação ou makefiles). **IMP.** O ficheiro `.zip` deverá incluir apenas ficheiros, não directórios.