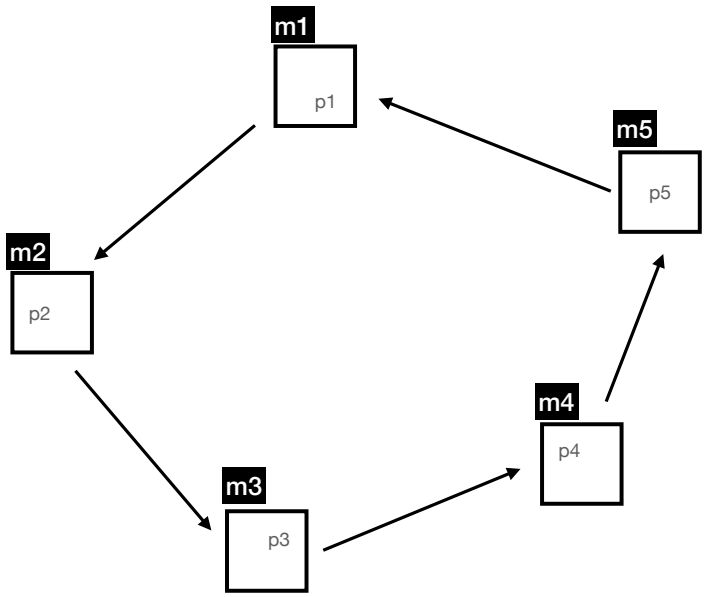
Trabalho Prático de Sistemas Distribuídos – 2021/22 –

Usando os templates que lhe foram fornecidos nas aulas de laboratório como ponto de partida, implemente protótipos de aplicações para os seguintes cenários.

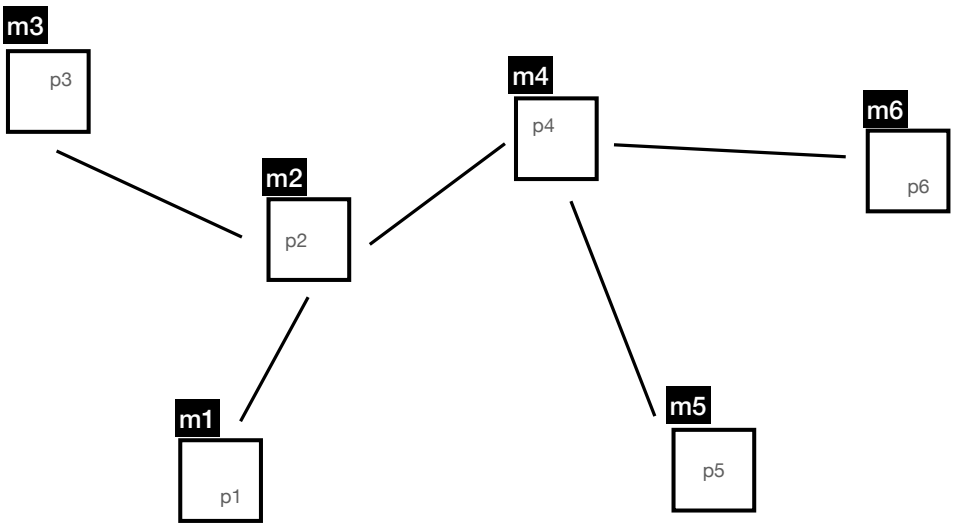
1. Algoritmo de "Token Ring"



1. capítulo 6 de van Steen e Tanenbaum ("Token Ring Algorithm");
2. crie um anel com 5 "peers" (`p1` a `p5`) como o representado na figura anterior;
3. note que cada "peer" deve estar numa máquina (`m1` a `m5`) no laboratório FC6 008;
4. cada "peer" tem apenas o IP do seguinte (e `m5` tem o IP de m1 para fechar o anel);

5. o cliente em cada "peer" tem uma shell que executa dois comandos possíveis: lock() e unlock();
6. um outro thread no "peer" executa um loop contínuo em que espera por uma mensagem (que só poderá vir do IP anterior) e envia-a para o nó que se segue no anel;
7. a mensagem (a "token") é constituída apenas por um inteiro;
8. o valor inicial da "token" é 0 e é incrementado de uma unidade por cada "peer" em que passa;
9. sempre que recebem a "token" os "peers" escrevem-na;
10. quando a shell executa lock() esse processo é interrompido, ficando a "token" no nó actual até o cliente executar unlock().

2. Rede P2P com "Push", "Pull" e "Push+Pull"



1. capítulo 4 de van Steen e Tanenbaum ("Information Dissemination Models");
2. crie uma rede com 6 "peers" (p1 a p6) correndo em outras tantas máquinas, m1 a m6, no laboratório FC6 008) como a topologia representada na figura anterior;
3. cada "peer" tem um cliente com uma shell que executa apenas 4 métodos (ver abaixo);
4. a construção da rede deve ser feita "peer" a "peer", usando mensagens do tipo register() (ver abaixo);

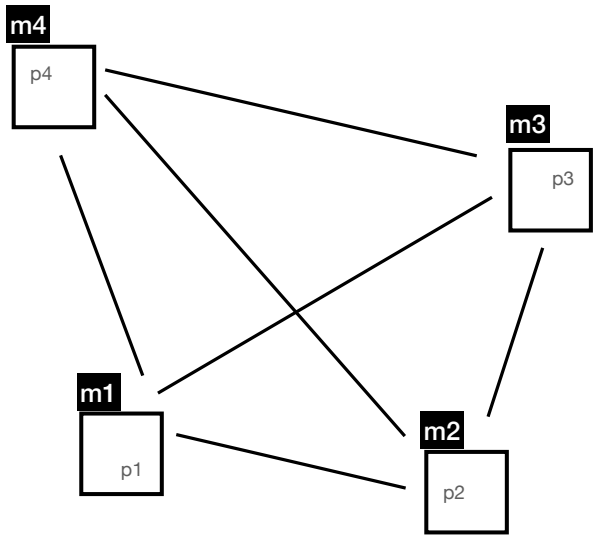
5. cada "peer" mantém uma tabela com os IPs dos nós que se registaram com ele;

6. cada "peer" mantém também um dicionário de palavras que vão sendo geradas periodicamente (e.g., arranjem um dicionário online e periodicamente retirem uma palavra à sorte e insiram no vosso dicionário - a estrutura de dados);

7. os comandos da shell (correspondentes à API do servidor) são (assumindo o cliente em ip1):

   - register(ip2) - que regista o "peer" actual com o "peer" em ip2 (se for bem sucedido a tabela de ip1 fica com ip2, a tabela de ip2 fica com ip1);
   - push(ip2) - o cliente envia para ip2 o conteudo do seu dicionário e ip2 actualiza as entradas em falta no seu dicionário;
   - pull(ip2) - o cliente pede a ip2 o conteúdo do seu dicionário e actualiza as entradas em falta no seu dicionário;
   - pushpull(ip2) - faz as duas operações anteriores (os dicionários dos dois "peers" ficam sincronizados temporariamente).

3. Algoritmo de "Reliable Totally Ordered Multicast"



1. capítulo 6 de van Steen e Tanenbaum ("Lamport Clocks" e "Totally Ordered Multicast");

2. crie uma rede com 4 "peers" (p1 a p4) noutras tantas máquinas do laboratório FC6 008 (m1 a m4) com a topologia representada na figura anterior;

3. cada “peer” tem uma tabela com os IPs de todos os restantes;
4. cada cliente de um “peer” corre uma shell que apenas recebe uma linha de texto e a envia para todos os outros “peers”, tal e qual numa aplicação do tipo “chat”;
5. o servidor em cada “peer” recebe o texto de clientes e imprime-o no monitor com o seu “timestamp” (ver abaixo sobre isto);
6. note que o problema é garantir que todos os “peers” vêm as mensagens pela mesma ordem;
7. implemente um Relógios de Lamport em cada “peer” para marcar as mensagens com “timestamps” relativos;
8. implemente o algoritmo de “Reliable Totally Ordered Multicast” (veja aqui uma descrição detalhada).

Condições Gerais

O trabalho prático deve ser realizado individualmente ou em grupo (2 pessoas no máximo). A constituição do grupo, com 1 ou 2 pessoas, deve ser comunicada até 12 de Novembro, via e-mail (nome completo dos membros e os respectivos números mecanográficos). Sugiro que use Java para a implementação. Se preferir outra linguagem fale comigo para eu validar. Pode usar sockets ou gRPC (eventualmente as 2 em aplicações diferentes). Assumindo a utilização do Java, o software produzido deve ser organizado em 3 packages:

- ds.trabalho.parte1
- ds.trabalho.parte2
- ds.trabalho.parte3

O trabalho deverá ser entregue até à data limite de 7 de Janeiro de 2022. Na semana seguinte será feita a sua apresentação com a presença obrigatória de todos os elementos do grupo. A entrega é feita enviando para lblopes@dcc.fc.up.pt um ficheiro .ZIP com estas packages e um ficheiro README.md (formato Markdown, verifiquem o output num browser) que, para além do nome completo dos membros do grupo e respectivo número mecanográfico, deve conter uma descrição de como compilar e executar cada uma das aplicações.

Bom trabalho,

Luís Lopes