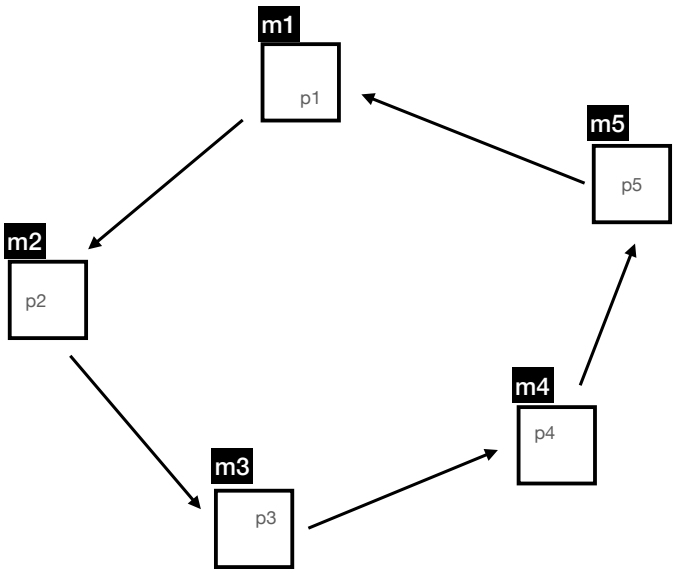
Trabalho Prático
Sistemas Distribuídos
– 2022/23 –

Partindo dos exemplos simples que foram apresentados nas aulas práticas, implemente protótipos de aplicações para os seguintes cenários.

1. Algoritmo "Token Ring"



1. deve ler primeiro van Steen & Tanenbaum (cap. 6, "Token Ring Algorithm");
2. crie uma rede em anel com 5 peers (dê-lhes os nomes `p1` a `p5`) como a representada na figura acima;
3. note que cada peer deve estar colocado numa máquina diferente (`m1` a `m5`);
4. cada peer conhece o endereço IP da máquina seguinte, onde está colocado o peer seguinte; `p1` sabe o IP de `m2`; `p2` sabe o IP de `m3`; ...; `p5` sabe o IP de `m1`, fechando assim o anel;

5. um cliente em cada peer tem uma pequena shell que executa apenas 2 comandos: lock e unlock;

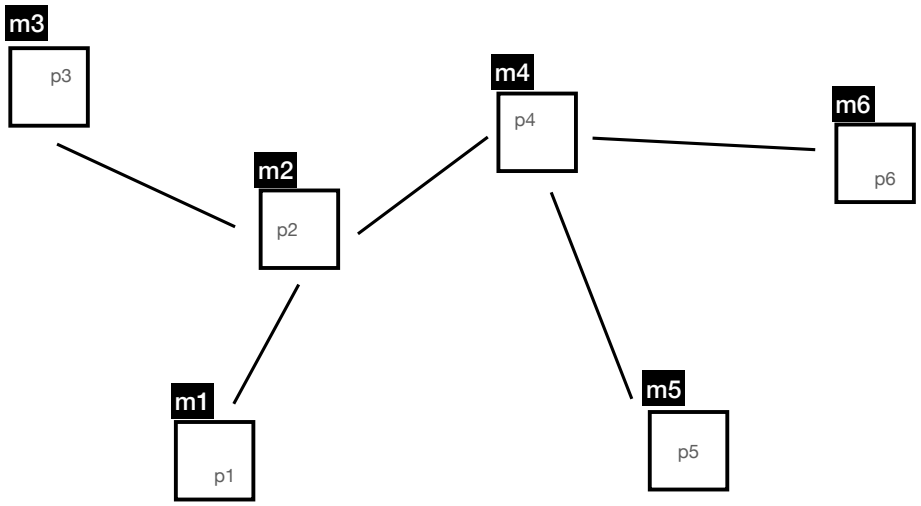
6. outro thread executa um ciclo em que espera por uma conexão de um peer (o anterior); quando se conecta, recebe de seguida uma mensagem que re-encaminha para o próximo peer no anel;

7. a mensagem (designada por "token"no algoritmo) apenas contém uma número inteiro; este valor é 0 no início da execução e é incrementado de uma unidade por cada hop realizado pela mensagem;

8. sempre que um peer recebe a token, imprime o seu IP e o valor actual da token no stdout;

9. quando o comando lock é executado na shell de um dos peers, a token deixa de ser propagada por esse peer até que o comando complementar, unlock, seja executado na mesma shell.

2. Disseminação de Dados via Gossiping



1. deve ler primeiro van Steen & Tanenbaum (cap.6, "Information Dissemination Models");

2. crie uma rede de 6 peers (p1 a p6), executando em máquinas diferentes (m1 a m6), com a topologia apresentada na figura acima;

3. a rede deve ser construída peer por peer usando um comando register a partir de peers diferentes (ver abaixo);

4. cada peer mantém uma tabela com os IPs das máquinas/peers que se registaram nesse peer;

5. o comando - register ip2 - regista o peer corrente, por exemplo p1, com o peer p2 no ip2 (se a operação for bem sucedida, a tabela de IPs para p1 fica com uma entrada ip2, a tabela de IPs para p2 fica com uma entrada ip1);

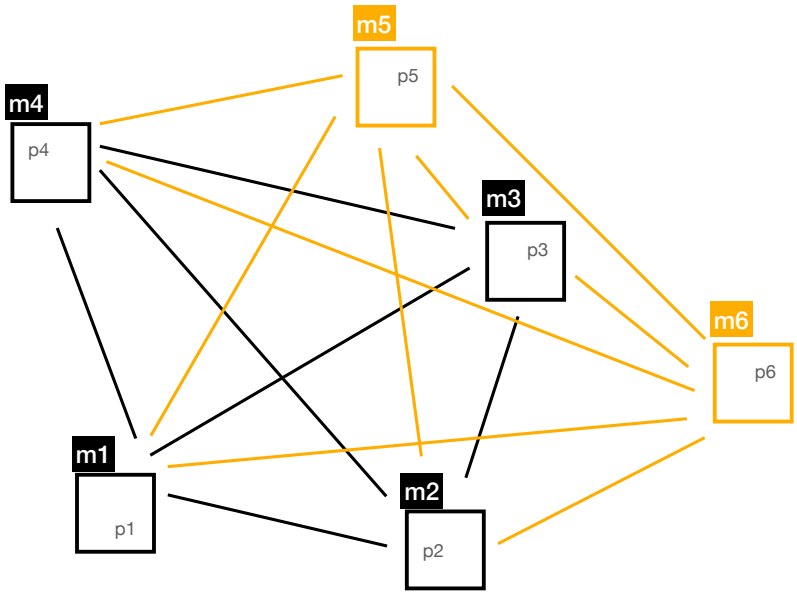
6. cada peer mantêm uma lista de palavras que cresce no tempo (inicialmente vazia); as palavras são geradas de forma aleatória como um processo de Poisson (veja código template disponibilizado no Moodle) com uma frequência de 1 evento a cada 30 segundos; a nova palavra é adicionada à lista;

7. as palavras devem ser selecionadas aleatoriamente de um ficheiro (o mesmo ficheiro para todos os peers - sugestão: obtenha um dicionário em texto na Web); num qualquer instante, cada um dos 6 peers terá uma coleção diferente de palavras;

8. de cada vez que uma palavra é gerada por um peer, deve ser disseminado para os restantes; aplique o Algoritmo de Gossiping;

9. note que deverá verificar se um peer já viu uma determinada mensagem recebida; neste caso o peer deve parar de disseminar a mensagem com probabilidade $1/k$ (use $k = 5$).

3. "Totally Ordered Multicast"



1. deve ler primeiro van Steen & Tanenbaum (cap.6, "Lamport Clocks" and "Totally Ordered Multicast");

2. crie uma rede de 6 peers (p1 a p6) cada um numa máquina diferente (m1 a m6) com a topologia especificada na figura acima;

3. cada peer tem uma tabela de IPs contendo todos os IPs das restantes 5 máquinas;

4. implemente "Relógios de Lamport" em cada peer de tal forma que as mensagens por ele enviadas possam ser adequadamente timestamped;

5. assuma que os peers p5 e p6 são réplicas de um mesmo serviço, e.g., uma base de dados, e que recebem pedidos dos restantes 4 peers;
6. o objectivo é garantir que p5 e p6 vêem a mesma sequência de mensagens vindas dos restantes peers; por outras palavras, que os peers da rede concordam com uma ordem das mensagens na rede antes de estas serem entregues às duas réplicas em p5 e p6;
7. os peers p5 e p6 podem ser implementados como bases de dados "mock"; apenas recebem as mensagens e imprimem-nas; não é necessário implementar uma base de dados real;
8. os outros peers geram pedidos para p5 e p6 de acordo com uma distribuição de Poisson com uma frequência de 1 por segundo; cada pedido é enviado a todos os restantes peers (cujos IPs estão na tabela de IPs do peer);
9. para garantir que todos os pedidos provenientes dos peers p1 a p4 são processados pela mesma ordem em p5 e p6 deve implementar o Algoritmo de "Totally Ordered Multicast"usando Relógios de Lamport para colocar timestamps nas mensagens (veja aqui outra descrição detalhada da aplicação do algoritmo).

General Remarks

Este trabalho prático pode ser feito individualmente ou em grupos de no máximo 2 alunos. **A composição dos grupos deve ser enviada por e-mail para mim até no máximo 28 de outubro**. Por favor, envie os nomes completos dos membros do grupo e os respectivos identificadores numéricos no Sigarra.

Sugiro que usem Java para implementar os 3 cenários. Pode usar Java Sockets ou gRPC (ou ambos em diferentes cenários). Se estiverem interessados em usar outra linguagem de programação, verifiquem comigo primeiro. Supondo que usa Java, o software que você produzirá deve ser organizado em 3 pacotes da seguinte forma:

- ds.assignment.tokenring
- ds.assignment.gossiping
- ds.assignment.multicast

O código deve ser enviado por e-mail para lblopes@dcc.fc.up.pt com o assunto DS2223:SUBMIT com um .ZIP anexado. Este arquivo deve conter o código fonte dos 3 pacotes e um arquivo de texto README com o nome e identificadores numéricos dos membros do grupo, junto com um pequeno texto explicando como compilar e executar os exemplos.

O prazo para envio do código é 6 de janeiro de 2023. Na semana seguinte, cada grupo fará uma demonstração das inscrições para mim (dia/hora a ser agendada posteriormente). **Esta demonstração é obrigatória e todos os membros do grupo devem estar presentes. O não cumprimento deste requisito resultará em uma nota de "0" no trabalho prático**.

Espero que gostem do trabalho,

Luís Lopes
lblopes@dcc.fc.up.pt