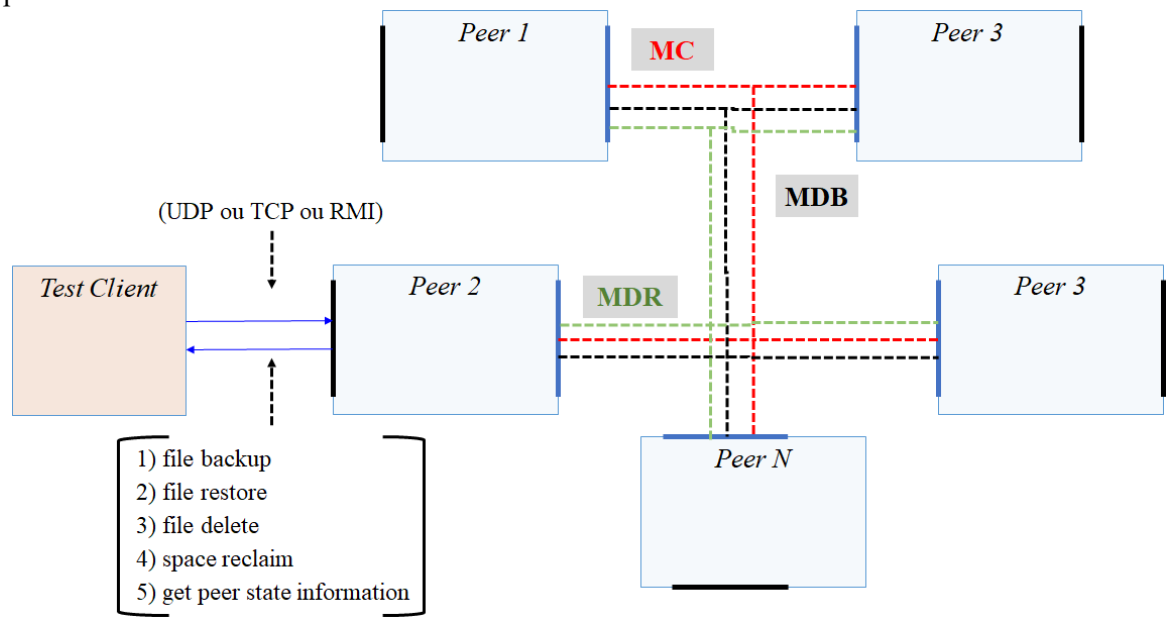
Trabalho 1 – Serviço de Backup Distribuído

1. Arquitectura Geral

Só para relembrar…



2. Objectivos para a Aula 5

O objetivo da 5ª aula TP é introduzir RMI no contexto do projeto.

A tecnologia RMI poderá ser empregue na comunicação entre o *test client* e os *peers* (vale 5%).
O objectivo é que vocês desenvolvam uma aplicação, cliente-servidor/peer, muito simples que permita desencadear (invocar do lado *peer*) os diferentes protocolos.

A interface de comunicação entre o cliente e os *peers* está descrita na Secção 4 da especificação do projeto. A forma de invocar a aplicação de teste está descrita na Secção 6 da mesma especificação.

Essencialmente, há que desenvolver um serviço RMI que permita a invocação, pelo cliente no peer, das 5 acções especificadas na secção 4:
1) fazer o *backup* de um ficheiro (não devem apagar o ficheiro no *test client* após o *backup* estar completo)
2) restaurar um ficheiro (o ficheiro original, se existir no *test client*, não deve ser substituído);
3) apagar os *chunks* de um ficheiro (não devem apagar o ficheiro original no *test client*)
4) definir o espaço em disco disponível para o *peer*
5) obter informação de estado do *peer*.

Obviamente, o desenvolvimento deverá ser feito incrementalmente.

Como temos estado a trabalhar sobre o subprotocolo de *backup*, devem concentrar-se em definir um interface RMI que permita ao cliente solicitar o *backup* de um ficheiro por parte de um determinado *peer*.

Um tutorial relevante para entenderem como trabalhar com RMI em Java é o seguinte:
https://docs.oracle.com/javase/6/docs/technotes/guides/rmi/hello/hello-world.html

Chamo-vos especialmente a atenção (no contexto do emprego de RMI) para o *rmiregistry*. Este é um processo separado (de qualquer uma das vossas aplicações), o qual é um servidor de nomes local. O *rmiregistry* permite a aplicações (por exemplo o vosso *peer*), registarem objectos remotos, i.e. objectos que oferecem métodos que podem ser invocados por outros objetos a executar em outras máquinas virtuais Java.
Esse registo é necessário para que (por exemplo, no caso do vosso trabalho), posteriormente os clientes possam obter referências para esses objetos (disponíveis nos *peers*) e desse modo invocar os métodos remotos.
Notem também que tanto para registar como para pesquisar um objeto remoto, o *rmiregistry* tem que poder aceder ao ficheiro *.class* com a interface remota implementada pelo objeto remoto.

De uma forma sumária o modo de emprego de RMI é o seguinte:
1. devem definir um Interface (p.e. *InterfaceX*) que estende o Interface *java.rmi.Remote*
2. este vosso interface deve declarar a assinatura de todos os métodos a serem fornecidos remotamente
3. cada um deste métodos deve fazer *throws* da excepção *java.rmi.RemoteException*
4. do lado servidor/peer devem criar uma classe (p.e. *ClassX*) que implementa o vosso Interface (*InterfaceX*), portanto, que implementa cada um dos métodos nele declarados
5. também do lado servidor/peer, para disponibilizar a funcionalidade remota, o vosso código terá de criar uma instância da classe *ClassX*, exportá-la para o Java RMI *runtime* para que este objecto possa receber os pedidos remotos, e regista-la no *rmiregistry* (associando-a a um determinado nome), para que ela possa sem procurada e encontrada pelos invocadores remotos
6. do lado cliente o vosso código necessitará de obter uma referência para o *rmiregistry*¸ e depois de obter o *stub* para o serviço remoto, empregando o *rmiregistry* para fazer o *lookup* desse *stub* e o nome antes estabelecido (aquando do registo pelo servidor/peer)
7. em posse do stub o código cliente podem então invocar os métodos remotos como se de uma normal invocação de métodos java se tratasse