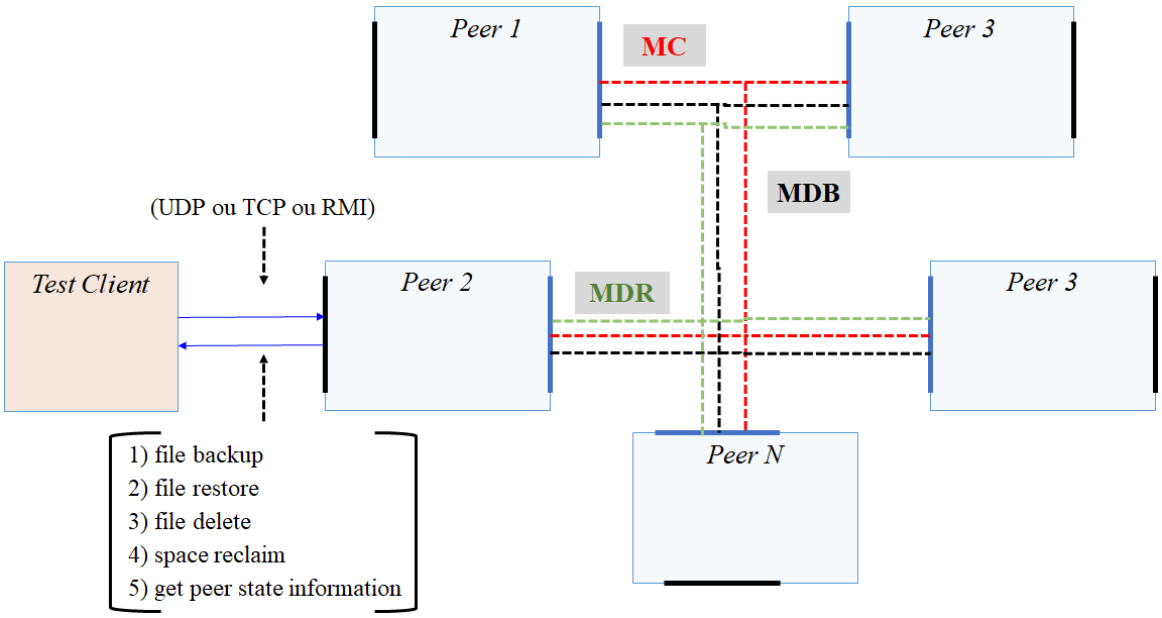
Notas sobre Trabalho 1

Já está disponível na página da cadeira o enunciado do primeiro projecto que vocês terão de desenvolver para avaliação.



Descrição sumária da estrutura e funcionamento do sistema a implementar:

- O trabalho a desenvolver constitui uma arquitectura P2P dedicada ao *backup* distribuido de ficheiros. Existirá assim um colectivo de *peers* que cooperará para armazenar de forma distribuida os ficheiros especificados, de acordo com um grau de replicação (número de cópias) também especificado. O sistema suportará também os serviços associados de *recovery*, *deletion* e *space reclaiming*;

- A arquitectura em questão compreende os dois seguintes tipos de componentes:
  - o *peer* – aplicação *p2p* que tanto recebe pedidos de outros *peers* como submete pedidos a outros *peers*. O sistema será assim composto por vários peers *idênticos* a correr simultaneamente e a interagir;
  - o *testClient* – aplicação que efectuará o interface entre o utilizador e um determinado *peer*. Apenas 1 *testClient* será empregue para intermediar a interacção entre o utilizador e um determinado peer;

- A comunicação entre os diferentes componentes proceder-se-á da seguinte forma:
  - o *testClient* comunica com o *peer* através (preferivelmente) de RMI;
  - os *peers* comunicam uns com os outros através de 3 canais multicast.

- Os canais multicast serão os seguintes:
  - MC – *Multicast Control Channel*. Canal empregue para a troca das mensagens de controlo do sistema. Todos os peers devem *estar à escuta* deste canal;
  - MDB – *Multicast Data Backup Channel*. Canal empregue nos procedimentos de *backup*;
  - MDR – *Multicast Data Recovery Channel*. Canal empregue nos procedimentos de *recovery*;

- Sobre a arquitectura descrita deverão ser suportados 4 serviços (protocolos) relacionados (oferecidos pelo *testClient* ao utilizador):
  - *Backup* – para replicar um ficheiro com um grau de replicação especificado;

  - Recovery – para recuperar um ficheiro que foi previamente replicado;
  - *Delete* – para eliminar do sistema um ficheiro antes replicado;
  - *Space Reclaiming* – para libertar espaço (disco) de um *peer* garantindo a preservação do *replication degree* dos ficheiros (*chunks*, para ser mais correcto) por ele armazenado;

- A operação do sistema será a seguinte:
  - o utilizador solicita um dos serviços suportados ao *testClient*;
  - o *testClient* constrói e envia ao *peer* o pedido correspondente;
  - o *peer* que recebe o pedido (designado como *initiator peer*) interage com os restantes *peers* (através dos canais *multicast* adequados) para levar a cabo os necessários procedimentos para assegurar o serviço solicitado (*backup*, *recovery*, *delete* ou *space reclaim*). Aos procedimentos desencadeados para suportar um determinado serviço (ou protocolo) chama-se (no guião) um sub-protocolo;
  - notem que o suporte de um determiando serviço pode despoletar um vasto conjunto de interacções envolvendo outros *peers* que não o próprio *initiator peer*;

- É de notar que (no contexto das operações/interacções que se desenvolvem para dar suporte aos serviços oferecidos pelo sistema) os *peers* não operarão sobre os ficheiros inteiros, mas sobre fragmentos destes, designados de *chunks*.
  Assim, aquando do *backup*, o ficheiro inteiro será fornecido ao *testClient* (pelo utilizador), que o fará chegar ao seu *peer*. Nesse ponto cada *peer* deverá fragmentar cada ficheiro no numero necessário de *chunks* para que cada um destes possa ser transportado no *payload* de um datagrama UDP (para poder circular nos canais de comunicação entre peers).
  Depois o *peer* deverá desenvolver o mesmo procedimento para cada *chunk* (*backup*). Estes procedimentos deverão ser executados concorrentemente, usando múltiplos *threads*.

- Da mesma forma, a recuperação (*recovery*) de um ficheiro implicará a recuperação individual de dada um dos *chunks* em que este foi dividido aquando do *backup*. O *initiator peer* tratará de obter (pedir aos outros *peers*) todos os *chunks* de um ficheiro, reconstituirá o ficheiro e só depois o devolverá ao *testClient* (e, portanto, ao utilizador);

- Em face do que é expresso acima fica claro que o *peer* deverá estar à escuta de pedidos vindos de dois lados: do *testClient* (através de uma ligação RMI) e do colectivo de *peers* (através dos 3 canais multicast). Um peer deve assim apresentar dois interfaces:
  - o interface para o *testClient* – serviço que permita ao *testClient* a submissão de pedidos de *backup*, *recovery*, *deletion*, ou *space reclaim* (preferencialmente usando Java RMI (vale 5%), mas pode no entanto ser implementado de outra forma, p.ex. usando TCP ou UDP);
  - o interface para os restantes peers – aqui referimo-nos aos mecanisos do *peer* que estão à escuta dos pedidos provenientes dos 3 canais multicast, portanto, à escuta de pedidos dos outros peers;

A acção dentro de um *peer* pode portanto ser desencadeada por um pedido vindo do *testClient* (utilizador) ou de um outro *peer*.

Notas gerais:

- Como não é possível testar os restantes protocolos/serviços sem implementar o protocolo de *Backup*, o recomendável é que comecem por implementar este protocolo. Para além disso, embora se pretenda uma implementação concorrente baseada em múltiplos *threads*, inicialmente podem fazer uma implementação mais simples, capaz de fazer o *backup* de um ficheiro com um *chunk* apenas;
- A ordem gobal mais adequada para implementarem os sub-protocolos é: *backup*, *delete*, *restore* e *reclaim*;

Notas sobre Trabalho 1

- Devem fazer uma implementação incremental. Ou seja implementar cada funcionalidade necessária isoladamente (p.ex. enviar/receber mensagens em multicast, calcular o identificador do ficheiro para *backup*, partir o ficheiro em *chunks*, etc.) e ir integrando essas funcionalidades gradualmente;
- Há um conjunto de possíveis *enhancements*, referidos ao longo do enunciado, que vocês podem implementar. Devem é implementar essas melhorias depois de completarem as versões base de todos os protocolos.
- A demonstração do vosso projecto requererá que seja possível executar os vossos *peers*, assim como o *testClient*, a partir da linha de comandos. Recomendamos assim que vocês criem um script para agilizar esse processo;
- É também necessário, para efeitos de teste, que a vossa implementação permita executar múltiplos *peers* no mesmo computador;
- Para além disso, a demonstração será realizada em PCs da sala de aula, assim para não serem penalizados por problemas súbitos que surjam durante a demonstração ensaiem o *setup* do vosso sistema antes da demonstração;
- Notem que a montagem do vosso sistema para demonstração valerá 5% da nota do projecto;
- Um dos testes a que os vossos *peers* serão sujeitos é um teste de interoperabilidade com os *peers* de outros grupos. Assim, podem e devem realizar esses testes entre as vossas implementações. No entanto, para o fazerem podem partilhar apenas ficheiros *.class* (pré-binários java) com outros grupos e não ficheiros *.java* (código fonte).
- Relativamente ao *replication degree* é de notar que o valor especificado (aquando do *backup*) é apenas o valor desejado. Um *chunk* pode acabar por ser replicado com um grau inferior, se p.ex. não houver um número suficiente de *peers*, ou superior.