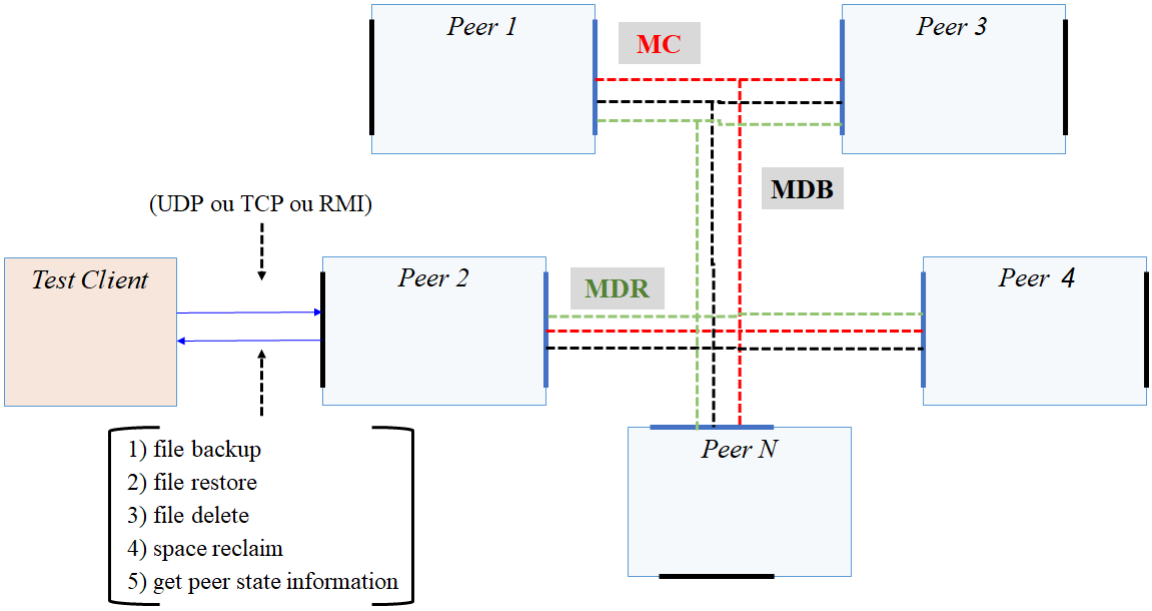
# 1. Introdução

Esta aula dedica-se à continuação do trabalho no projecto 1. Iremos analisar o *Space-Reclaimimg Sub-protocol*

# 2. Space-Recaliming Sub-Protocol




*Transcrito do Guião do Projecto 1*

…..

The algorithm for managing the disk space reserved for the backup service is not specified. Each implementation can use its own. However, when a peer deletes a copy of a chunk it has backed up, it shall send to the MC channel the following message:

```
<Version> REMOVED <SenderId> <FileId> <ChunkNo> <CRLF><CRLF>
```

Upon receiving this message, a peer that has a local copy of the chunk shall update its local count of this chunk. If this count drops below the desired replication degree of that chunk, it shall initiate the chunk backup sub-protocol after a random delay uniformly distributed between 0 and 400 ms. If during this delay, a peer receives a PUTCHUNK message for the same file chunk, it should back off and restrain from starting yet another backup sub-protocol for that file chunk.

…..

# 3. Operação

Quando um *peer* faz *backup* de um *chunk* (recebeu mensagem *Putchunk* e guardou *chunk*) fica corresponsável (junto com todos os outros *peers* que também fizeram *backup* desse *chunk*) por garantir que o *replication degree (rd)* efectivo do *chunk* nunca é inferior ao *rd* especificado pelo utilizador.

Assim, se algum evento ocorrer no sistema que venha reduzir o *rd* efectivo de um *chunk* de forma a que este passe a ser inferior ao *rd* desejado, os *peers* do colectivo responsável pelo armazenamento do *chunk* em questão devem levar a cabo os necessários procedimentos para repor o *rd*. Ou seja, se uma cópia de um *chunk* é eliminada de um *peer*, os restantes (que armazenam cópias do *chunk*) devem garantir que a cópia será recriada noutro *peer* (que não tenha ainda qualquer cópia do *chunk*).

Esta obrigação de reposição do *rd* não se aplica, obviamente, no caso de ser dada uma ordem de *Delete* de um ficheiro (e, logo, dos seus, *chuncks*) assim como no caso de um *peer* ficar offline (e, assim, os seus *chunks* inacessíveis).
No fundo, aplica-se especificamente no caso do emprego do protocolo *SpaceReclaim*.

Exemplo concreto da aplicação do que se expressa acima:

a) temos um sistema inicialmente composto por 3 *peers* (P1, P2 e P3);
b) P1, recebe um comando do *TestClient* para fazer *backup* de um ficheiro (composto apenas de 1 *chunk*) com *rd*=2. O P1 é assim o *peer* iniciador do protocolo;
c) o P1 desenvolve o processo de backup do *chunk* em questão ficando este armazenado em P2 e P3;
d) um novo *peer* (P4) junta-se ao sistema;
e) uma ordem se *SpaceReclaim* é enviada para P3 levando a que este tenha de eliminar o *chunk*;
f) P3 procede ao envio da mensagem *Removed* para o canal MC;
g) P2 (*peer*, que junto com P3, é corresponsável pelo *chunk*) escuta a mensagem *Removed* e apercebe-se que o *rd* efectivo desceu para 1 (1<2). Espera um intervalo aleatório, entre 0 e 400ms, e acaba por desenvolver o processo de backup do *chunk* em questão (envia a mensagem *Putchunk* para o canal MDB). Neste caso, o P2 fará, invariavelmente, isso pois não resta mais nenhum *peer* (do colectivo original que armazenou o *chunk*) para o fazer (o P3 foi quem ficou sem espaço e o P1 não pode participar no backup do *chunk*);
h) o P4 escuta a mensagem *Putchunk* e trata de armazenar o *chunk* e enviar a respectiva mensagem *Stored*.
i) P2 recebe a mensagem *Stored* e considera terminado o processo. O *rd* do *chunk* voltou assim a ser 2.