**Faculdade de Engenharia da Universidade do Porto­­­­­­­­**



**Mestrado Integrado em Engenharia Informática e Computação**

**Sistemas Distribuídos**

**Grupo t5g04**

Bruno Piedade – up201505668

Danny Soares – up201505509

**Descrição de design e implementação de concorrência**

Para permitir a execução concorrente de múltiplos pedidos, foi usada uma pool (classe *ThreadPoolExecutor*) para gerir a execução das diferentes *threads*. Para cada pedido recebido é instanciado um objeto *Runnable* para processar a tarefa e no caso do protocolo BACKUP e RESTORE guardada uma referência para o mesmo num *Map* (classe *ConcurrentHashMap*) para permitir posteriores notificações aquando da receção de uma mensagem relevante para o pedido que está a ser processado.

No servidor existem três *threads* que escutam constantemente os canais *multicast* e criam uma nova *thread* para processar qualquer mensagem recebida. Para o MC(*multicast control channel*) é criada uma thread da classe *ControlProtocol* que processa as mensagens do tipo STORED, GETCHUNK, DELETED e REMOVED; para o MDB(*multicast data backup* channel) uma *thread* da classe *StoreChunk* que processa as mensagens do tipo STORECHUNK; para o MDR(*multicast data restore channel*) uma *thread* da classe *Chunk* que processa as menagens do tipo GETCHUNK. Estas *threads* realizam o *parsing* do conteúdo da mensagem, processam as mesmas e caso seja necessário enviam a reposta correspondente.

O envio de mensagens é feito através de três *sockets multicast* partilhadas para todo o servidor cujos acessos competitivos são geridos pela classe *TwinMulticastSocket*.