Sistemas Distribuídos

Serviço de Backup Distribuído

Grupo T5G05:
João Nuno Fonseca Seixas - up201505648
Vicente Fernandes Ramada Caldeira Espinha - up201503764

Concorrência

Neste trabalho é possível usar concorrentemente os protocolos feitos, uma vez que foram implementadas *threads*, quer para ficar à espera de alguma mensagem num dos três canais de comunicação, quer para processar cada mensagem recebida ou quer para enviar uma resposta ou pedido aos outros *Peers*.

Para implementarmos usamos uma *ScheduledThreadPoolExecutor* em cada *Peer* de forma a termos uma estrutura que nos permitisse usar *threads* e além de usar *threads* também nos permitisse agendar uma *thread* (usado quando temos que dar algum *delay* à resposta de alguma mensagem por exemplo).

```
executer = (ScheduledThreadPoolExecutor) Executors.newScheduledThreadPool(300);
```

```
Random rand = new Random();
int randomNum = rand.nextInt(400);
//new thread here to process the received message (random between 0 and 400ms)
Peer.executer.schedule(new MCSendScheduled(packet), randomNum, TimeUnit.MILLISECONDS);
```

Nas imagens acima mostra como criamos essa estrutura e tem um exemplo de agendar uma *thread* com um tempo entre 0 e 400ms.

Para uma thread poder ser executada, tem que lhe ser passado como argumento um *Runnable*, ou seja, tem que ser passado um objeto de uma classe que implementa Runnable, que por sua vez tem que ter o método *run* para a poder implementar. Na imagem seguinte é a nossa implementação do método *run* para ler as mensagens que chegarem no *MulticastChannel* de Controlo.

```java
@Override
public void run() {
    while (true) {
        try {

            byte[] buffer = new byte[65000];

            DatagramPacket recv = new DatagramPacket(buffer, buffer.length);
            this.msocket.receive(recv);

            byte[] temp = Arrays.copyOf(buffer, recv.getLength());

            Peer.executer.execute(new ParserMessages(temp, "MC"));

        } catch (SocketException e) {
            System.out.println("Error receiving packet");
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```