U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Sistemas Distribuídos

Relatório do Primeiro Projeto

MIEIC

(14 de abril de 2020)

Grupo T7G08

Diogo Ferreira de Sousa (up201706409@fe.up.pt)

Maria Gonçalves Caldeira (up201704507@fe.up.pt)

Introdução

Este relatório tem como objetivo descrever e explicar a execução simultânea de protocolos.

Execução simultânea de protocolos

Para a execução simultânea de protocolos o grupo optou por usar estruturas de dados como o *ConcurrentHashMap*, estrutura preferível em ambientes de Multithread, e também usamos o *Thread.sleep()*.

```
private ConcurrentHashMap<String, Integer> storedOccurrences;
```

Figura 1: *ConcurrentHashMap* na classe PeerData

```
DatagramPacket packet = new DatagramPacket(message, message.length, mdb.getAddress(), mdb.getPort());
socket.send(packet);
Thread.sleep( millis: 400);
```

Figura 2: Uso do *Thread.sleep()* na função *backup* da classe Peer

No que toca à classe Peer, esta tem um atributo para cada canal: MC (ChannelControl), MDB (ChannelBackup), MDR (ChannelRestore). Esta classe, ao ser instanciada, cria um thread para cada um destes atributos.

```
private static ChannelBackup mdb;
private static ChannelControl mc;
private static ChannelRestore mdr;
```

Figura 3: Declaração dos canais multicast na classe Peer

```
new Thread(mc).start();
new Thread(mdb).start();
new Thread(mdr).start();
```

Figura 4: Execução das Threads

Deste modo, após a identificação do protocolo na classe TestApp, é possível criar uma thread para esse protocolo, permitindo a execução simultânea.