**Redes de Computadores**

**2017/18**

**Trabalho Prático 2**

**Addendum**

### **0 - AVISOS**

**28/11/2017 - 16h30**

Constatou-se que o débito médio disponível no teste #2 é excessivamente baixo para visualizar os filmes utilizando segmentos das qualidades mais elevadas. A imagem do teste #2 foi atualizada para corrigir esse problema. Para atualizar as imagens fazer:

(sudo) docker pull smduarte/rc17-t2-base

(sudo) docker pull smduarte/rc17-t2-test1

(sudo) docker pull smduarte/rc17-t2-test2

(sudo) poderá ser necessário em algumas instalações do docker.

**4/12/2017 - 18h30**

A forma de realizar a entrega do trabalho, as datas de entrega e as penalizações por atraso já estão definidas no ponto a seguir.

### **1 - INSTRUÇÕES PARA ENTREGA DO TRABALHO**

Para entregar o trabalho terá que:

* Gerar um ficheiro PDF contendo a listagem das sources, com páginas numeradas e com o código da listagem (como explicado em 1.1)
* Preparar um arquivo executável (JAR) designado por MyDashProxy.jar e que permita executar o proxy na forma

java -jar MyDashProxy.jar <url-base> <playout-delay>

(como explicado em 1.2)

Deve guardar depois o executável num arquivo ZIP, de modo a ter assim um

ficheiro para submissão na forma **MyDashProxy.jar.zip**

* Ter obtido os dados dos testes que são afixados pelo player/browser, pois vão ser pedidos no formulário de submissão
* De seguida submeter o ficheiro PDF, o arquivo JAR e as respostas, no formulário de entrega do trabalho (como explicado em 1.3)

#### **1.1 - Para preparar um ficheiro PDF com a listagem com linhas numeradas e código para submissão**

#### **Prepare primeiro um ficheiro .ZIP só com o código fonte das classes Java da sua solução do trabalho 2**

Para este efeito utilize o programa de criação de ZIPs da sua preferência. Não coloque neste ZIP senão os ficheiros java das classes por si desenvolvidas pois se colocar outros ficheiros inúteis será prejudicado.

#### **A partir do anterior ZIP crie um ficheiro PDF com os ficheiros fonte das suas classes Java concatenadas e com as linhas numeradas**

Para criar o ficheiro PDF com a listagem numerada das suas classes Java utilize o seguinte comando:

**bash paginar\_sources.sh <zip\_com\_sources>** // ou em alternativa

**./paginar\_sources.sh <zip\_com\_sources>**

O script **paginar\_sources.sh** está disponível no CLIP.

No fim obterá um ficheiro PDF com o código fonte da sua solução com as linhas numeradas e com um código identificador da listagem (canto superior esquerdo do PDF). Tal como no primeiro trabalho, deverão levar uma versão impressa deste PDF para o teste de aferição do trabalho 2.

#### **1.2 - Preparar o arquivo JAR executável que possa ser usado para demonstração**

Proceder como anteriormente fez no caso do Trabalho 1 para a criação de um JAR executável. Criar o JAR com a designação MyDashProxy.jar (o Anexo 1 tem de novo as instruções para a sua criação).

#### **1.3 - Submeta o trabalho fazendo UPLOAD do PDF (listagem) do arquivo JAR (MyDashProxy.jar.zip) referentes à sua solução**

Para tal aceda ao seguinte formulário:

https://goo.gl/forms/FGdzb4uQYhwBaHYB2

Siga as instruções (fazendo UPLOAD do PDF e do arquivo JAR e respondendo às questões do formulário). No final, ao submeter o formulário irá obter uma mensagem de sucesso da submissão.

Se tiver de fazer mais do que uma submissão, só a última será considerada.

#### **1.5 - Datas de entrega e penalizações**

A data final de entrega do trabalho 2 é o dia 8 de Dezembro de 2017 pelas 24 horas (meia noite).

Os trabalhos entregues até às 24 horas de dia 9 de Dezembro têm 0,5 valores de penalização.

Os trabalhos entregues até às 24 horas de dia 10 de Dezembro têm 1 valores de penalização total.

Os trabalhos entregues até às 24 horas de dia 11 de Dezembro têm 2 valores de penalização total.

Os trabalhos entregues até às 24 horas de dia 12 de Dezembro têm 3 valores de penalização total.

Os trabalhos entregues até às 24 horas de dia 13 de Dezembro têm 4 valores de penalização total.

Os trabalhos entregues até às 24 horas de dia 14 de Dezembro têm 6 valores de penalização total e assim sucessivamente.

Não são aceites entregas depois do dia 17 de Dezembro.

#### **1.5 - Classificação das diferentes soluções**

As notas finais dos trabalhos estão dependentes da nota do trabalho de grupo e da nota do teste de aferição individual. A seguir indicam-se as notas correspondentes aos trabalhos de grupo em função da sua qualidade global.

Como está indicado no enunciado, a solução deverá ser capaz **de pelo menos** mostrar os filmes usando duas (ou mais) resoluções distintas (*tracks*) apresentadas em alternativa (um ou mais segmentos numa resolução, seguidos de um ou mais segmentos na outra resolução, etc.). Se esta solução passar nos testes indicados, estiver razoavelmente bem relatada e com um código fonte aceitável, a mesma é classificada com uma nota de 10 a 12.

Uma solução para ser classificada com **Suficiente Mais ou Bom**, deverá realizar a adaptação dinâmica da resolução segmento a segmento. Se esta solução passar os testes, estiver razoavelmente bem relatada e com um código fonte de boa qualidade, a mesma poderá ser classificada com uma nota entre 13 e 15, de acordo também com os resultados dos testes.

**Valorizações por implementação de opções extra:**

Suporte para a versão 1.1 do protocolo HTTP para o diálogo entre o proxy e o servidor de segmentos (servidor docker) poderá ser valorizada até 3 valores mais.

Suporte para servir vários *browsers* em paralelo poderá ser valorizada até 3 valores mais.

A nota máxima do trabalho de grupo é 20.

### **2 - ROADMAP PROPOSTO**

Com o enunciado presente...

1. Preparar e testar o ambiente de execução.
   1. lançar o servidor com o comando indicado no enunciado;
   2. testar o servidor, apontando o browser para o servidor;
   3. usando o browser, confrontar a estrutura dos filmes com o indicado no enunciado.
2. Compreender o problema e âmbito da solução.
   1. Entender a arquitectura da solução;
   2. Identificar na arquitectura o componente que precisa de desenvolver e o seu papel.
3. Desenvolvimento incremental da solução.
   1. Enumerar os filmes disponíveis no servidor;
   2. Processar os ficheiros **descriptor.txt**;
   3. Descarregar os segmentos de um filme na devida sequência;
   4. Ser capaz de receber pedidos do player HTTP;
   5. Entregar segmentos em respostas aos pedidos deste;
   6. Confirmar a solução com o teste1.
4. Refinamento da solução
   1. Atender aos requisitos de adaptação da qualidade da imagem ao débito disponível;
   2. Confirmar e afinar a solução recorrendo ao teste2.
5. Endereçar as opções valorativas
   1. Concorrência para suporte de vários utilizadores;
   2. Suporte na mesma solução dos protocolos HTTP 1.0 e HTTP 1.1

#### **3 - FAQ - RESPOSTA A PERGUNTAS FREQUENTES**

1. Instalação do DOCKER (LINUX)
   1. Basta seguir as instruções na página do docker para a sua versão.
      1. <https://docs.docker.com/engine/installation/>
      2. poderá ser necessário usar:

**sudo docker run ...**

1. Instalação do DOCKER (WINDOWS)
   1. Primeiro, decidir qual das duas opções de instalação se vai usar. No entanto, recomenda-se a versão **DOCKER TOOLBOX (CE)**.

A versão DOCKER TOOLBOX baseia-se no VirtualBox e instala uma pequena máquina virtual linux onde são, de facto, executados os *container* docker. É ainda necessário garantir que o componente Hyper-V do windows está desligado. [Instruções aqui…](http://www.poweronplatforms.com/enable-disable-hyper-v-windows-10-8/)

No BIOS, acessível no arranque, é necessário ter o suporte de virtualização no CPU ligado.

* 1. Ver o link <https://docs.docker.com/docker-for-windows/install/>
  2. Optar por instalar a versão acessível em Stable Channel

1. Instalação para MAC OS:
   1. Ver em: <https://docs.docker.com/docker-for-mac/install/>

***Nota***: tenha em conta que os computadores no laboratório têm o docker instalado pelo que terá condições aí de realizar o trabalho ou testar de acordo com as instruções de entrega, independentemente de ter ou não uma instalação própria.

## 

## 

## 

## **ANEXO 1**

## **Como construir um jar executável**

Para obter uma versão estável do seu **MyDashProxy.jar** construa um arquivo jar executável com o código do seu proxy. Notar que é apenas necessário incluir no **jar** executável o cliente e as classes de que este necessita. Para o efeito proceda da seguinte forma:

### **Usando o ambiente Eclipse**

Siga as instruções na documentação Eclipse que pode encontrar em: <https://eclipse.org/documentation/>

Procure na documentação da sua versão de Eclipse:

**Creating JAR Files > Creating a New Runnable Jar**

Depois de criar o jar executável pode testar se está correto e funciona, executando-o (como se fosse uma aplicação), verificando que exibe o mesmo comportamento do código do cliente que desenvolveu e que antes tinha testado.

### **Usando a ferramenta jar numa linha de comando (shell)**

Compile o cliente (com javac) obtendo todas as classes correspondentes à implementação e execução do cliente que desenvolveu. Criar na diretoria que contém esses ficheiros um ficheiro txt (ex., **“manifest.txt”**) com uma única linha

**Main-Class: classname**

em que **classname** será a classe com o Main do seu cliente (ex. **MyDashProxy**)

Para criar um jar executável chamado **MyDashProxy.jar** (que será um único ficheiro ou arquivo com extensão **.jar**) basta correr o seguinte comando jar na mesma diretoria:

**jar cvmf manifest.txt MyDashProxy.jar \*.class**

Para verificar em qualquer momento o conteúdo do arquivo jar executável que foi criado dê o comando:

**jar tvf MyDashProxy.jar**

Deverá obter um output da forma:

0 Thu Oct 13 11:37:12 WEST 2016 META-INF/

96 Thu Oct 13 11:37:12 WEST 2016 META-INF/MANIFEST.MF

5526 Thu Oct 13 11:29:12 WEST 2016 MyDashProxy.class

… etc. …...

Para testar que pode executar o seu jar executável **MyDashProxy.jar** dê o comando:

**java -jar MyDashProxy.jar**

Deve observar o mesmo comportamento da execução do seu cliente que colocou no arquivo executável

Note que, como se refere nas instruções de entrega, o seu ficheuro Jar terá que ser submetido ainda num arquivo ZIP, ou seja: **MyDashProxy.jar.zip**