Redes de Computadores (RCOMP) – 2015/2016

Segundo trabalho prático

• API *Berkeley Sockets* • Desenvolvimento de aplicações de rede em linguagem C. • Desenvolvimento de aplicações de rede em linguagem Java.	•

Condições gerais de realização do trabalho

- É realizado em grupo, sendo os grupos constituídos por 3 a 4 alunos que devem pertencer à mesma turma PL de Redes de Computadores.

- É realizado fora do horário letivo, sendo no entanto apoiado em algumas aulas práticas laboratoriais (PL).

- Deverá ser desenvolvido usando apenas as **linguagens C e/ou Java**, não serão aceites trabalhos desenvolvidos noutras linguagens.

- É entregue (relatório), no serviço MOODLE de apoio à disciplina. A data limite para entrega do projeto é 22 de maio (às 23:55). Os alunos com estatuto especial podem entregar o projeto até 12 de junho (às 23.55). **NO MOODLE NÃO SÃO ACEITES SUBMISSÕES FORA DO PRAZO.**

- O projeto deve ser submetido em formato ZIP via MOODLE contendo o relatório em formato PDF e os ficheiros de código fonte.

O nome do ficheiro a utilizar para entrega no serviço MOODLE deve ser:

TR2_{TURMA PL}_{NºALUNO}_{NºALUNO}_{NºALUNO}[_{NºALUNO}].zip

, por exemplo: **TR2_2DE_1022222_1033333_1044444_1055555.zip**

- Os trabalhos apenas serão classificados após a sua apresentação, a realizar numa aula PL na semana seguinte à sua submissão. Para os alunos com estatuto especial que entreguem o projeto até 12 de junho, a apresentação realiza-se em data a fixar posteriormente.

- Os alunos com estatuto especial que pretendam usufruir do prazo alargado devem formar grupos separados. Não devem integrar grupos em que existam alunos sem estatuto especial.

Projeto

Dado um conjunto de especificações pretendidas para um serviço de rede pretende-se:

- Definição de protocolos de aplicação criados.
- Implementação da aplicação desenvolvida em C ou Java.
- Produto final cujo funcionamento deverá ser demonstrado na apresentação do projeto.

1. Resumo do projeto

- Pretende-se o desenvolvimento de uma aplicação servidora HTTP capaz de operar em *cluster*.

- Cada instância da aplicação servidora opera num nó de uma mesma rede e disponibiliza via método ***GET*** um conjunto de ficheiros que possui localmente armazenados.

- Usando *broadcast* em UDP cada instância informa as restantes sobre a lista de ficheiros que disponibiliza.

- Como resultado cada instância disponibiliza uma página WEB com uma lista de nomes de ficheiros e respetivas ligações de acesso via HTTP.

- Na lista de ficheiros estarão incluídos ficheiros disponibilizados pela própria instância e ficheiros disponibilizados por outras instâncias do *cluster*.

- Em qualquer das instâncias do *cluster* a lista de ficheiros será a mesma, mas as ligações de acesso HTTP nem sempre correspondem ao endereço do nó da instância que disponibiliza a página uma vez que diferentes ficheiros estarão acessíveis em instâncias diferentes.

2. Especificações detalhadas

- O servidor HTTP integrado na aplicação apenas necessita de suportar o método GET com conteúdo do tipo ***text/html***.

- Cada servidor do *cluster* disponibiliza determinados ficheiros HTML via HTTP num número de porto fixo.

- Os clientes podem aceder via HTTP a qualquer um dos servidores obtendo uma página inicial (***GET /***) com uma lista de nomes de ficheiros e respetivas ligações HTTP para todos os ficheiros disponíveis no *cluster*.

- Cada servidor do *cluster* mantém uma lista dinâmica dos ficheiros disponíveis, atualizada de 30 em 30 segundos.

- Todos os servidores HTTP que constituem o *cluster* estarão na mesma rede IPv4, por isso podem contactar-se diretamente por *broadcast* na rede local (‘‘255.255.255.255’’).

- Para atualizar a lista de ficheiros, de 30 em 30 segundos cada um dos servidores emite uma mensagem UDP em *broadcast* dirigida a um número de porto fixo para detetar outros elementos do *cluster*. Obtendo em resposta uma lista dos ficheiros disponibilizados por cada um deles.

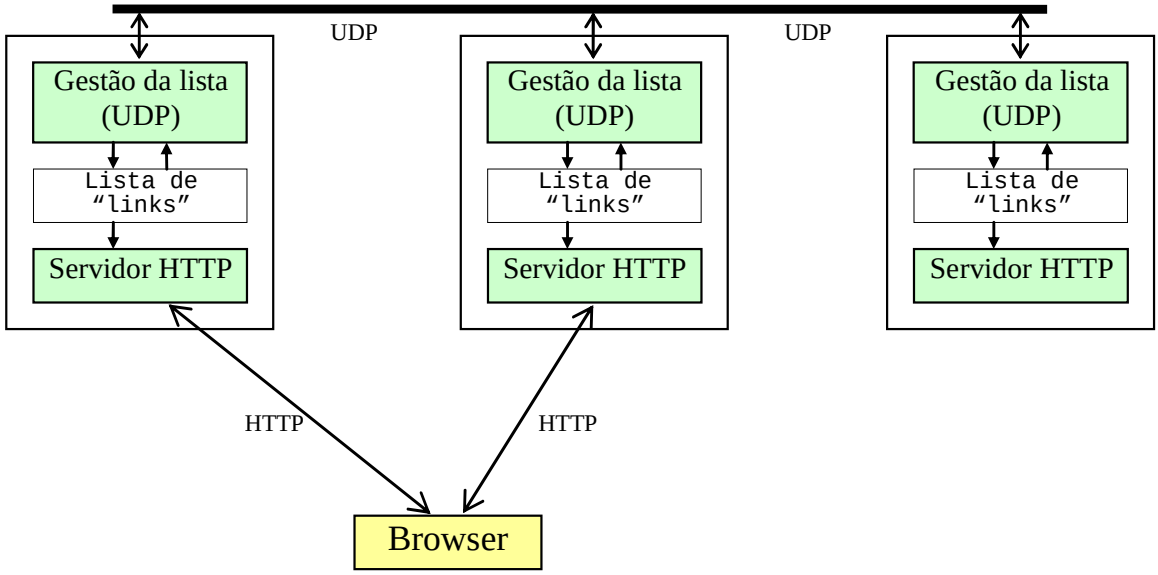
- A lista de ficheiros fornecida via UDP em resposta ao *broadcast* deve ter comprimento limitado para evitar que o tamanho do *datagrama* UDP ultrapasse os 512 bytes.

- Os servidores anteriormente detetados que não respondem ao *broadcast* são eliminados da lista.

- É admissível que existam nomes de ficheiros repetidos em diferentes servidores do *cluster*.

- A página inicial (***GET /***), além dos nomes dos ficheiros disponíveis e respetivas ligações HTTP, deve apresentar informação relativamente ao **número de servidores existentes no cluster e respetivos endereços IPv4**.

- Diagrama da arquitetura:



3. **Número de porto TCP e UDP a utilizar**

- Cada aplicação terá necessidade de usar um número de porto fixo TCP (para o serviço HTTP) e um número de porto fixo UDP (para a gestão das listas).

- Para evitar conflitos, são definidas as seguintes gamas de números de porto (UDP e TCP) a usar em cada turma:

  - 2DA - 30000 a 30099
  - 2DB - 30100 a 30199

- 2DC - 30200 a 30299
- 2DD - 30300 a 30399
- 2DE - 30400 a 30499
- 2DF - 30500 a 30599
- 2DG - 30600 a 30699
- 2DH - 30700 a 30799
- 2DI - 30800 a 30899
- 2DJ - 30900 a 30999
- 2DK - 40000 a 40099
- 2DL - 40100 a 40199
- 2NA - 40200 a 40299
- 2NB - 40300 a 40399

- Desta forma pretende-se evitar interferências entre os projetos. Cada grupo necessita apenas de um número de porto, devendo usar o **mesmo número para TCP (HTTP) e para UDP**.

- Dentro de cada turma os grupos deverão estabelecer entre si o número de porto que vão usar dentro da gama correspondente.

4. Alguns fatores de valorização previstos

- Suportar outros conteúdos no método GET além do tipo ***text/html***.

- Refrescamento automático pelo *browser* da página inicial carregada.

- Detetar e eliminar da lista nomes de ficheiros repetidos.

- Permitir que os elementos do *cluster* possam não se encontrar todos na mesma rede IPv4.

- Apresentar na página inicial nomes de máquinas em lugar de endereços IPv4 (incluindo nas ligações HTTP).

- Permitir listas de ficheiros de comprimento ilimitado na resposta à mensagem UDP em *broadcast*.