Redes de Computadores

1° Semestre 2014/2015
LEIC Alameda

Programação usando a interface de *Sockets*

"Content Delivery Network"

"A content delivery network (CDN) is an interconnected system of computers on the Internet that provides Web content rapidly to numerous users by duplicating the content on multiple servers and directing the content to users based on proximity."

1. Introdução

Pretende-se desenvolver uma aplicação de distribuição de conteúdos, com vários servidores que replicam os conteúdos, sendo os pedidos dos utilizadores respondidos pelo servidor mais "próximo".
Neste trabalho consideramos que existirá um servidor central, *Central Server* (CS), que será o ponto de contacto dos clientes quer para solicitar conteúdos, quer para disponibilizar novos conteúdos para o sistema.
Assume-se que o utilizador conhece o URL onde está disponível o servidor central.

A interação com o CS permite ao utilizador listar os conteúdos disponíveis e obter a identificação do servidor de conteúdos (SS) mais adequado para o descarregar. A aplicação do utilizador fará então a transferência desse conteúdo do servidor SS para a diretoria local previamente selecionada.
O sistema permite também carregar novos conteúdos para os servidores, através do contacto com o servidor central, que depois replica o conteúdo por todos os servidores SS disponíveis.

Para a implementação, os protocolos da camada de aplicação operam de acordo com o paradigma cliente-servidor e servem-se dos serviços da camada de transporte através da interface de *sockets*, usado os protocolos UDP e TCP.
O *Central Server* aceita pedidos através do protocolo UDP, identifica o servidor de conteúdos, *Storage Server* (SS), mais adequado e responde ao cliente com a indicação de qual o servidor a que deve ser pedido o conteúdo numa ligação em TCP.
O *Central Server* aceita novos conteúdos através de uma ligação TCP estabelecida pelo utilizador, replicando depois esses conteúdos nos servidores SS existentes usando ligações TCP.

2. Especificação do Projecto

2.1 Utilizadores

O programa que implementa o cliente do serviço de acesso a conteúdos deve ser invocado usando o seguinte comando:

`user` *`[-n CSname] [-p CSport]`*,

em que:

`CSname` é o nome da máquina que aloja o servidor central (CS) a contactar. Este argumento é opcional. Em caso de omissão, assume-se que o servidor está a correr na própria máquina.

`CSport` é o porto bem-conhecido no qual o servidor CS aceita pedidos por parte de utilizadores, quer em TCP quer UDP. Este argumento é opcional. Em caso de omissão, assume o valor 58000+NG, onde NG corresponde ao número do grupo.

Logo após a invocação do cliente, este aguarda a indicação da acção a tomar, estando previstas os seguintes comandos:

- *`list`* – na sequência desta instrução a aplicação de cliente deve comunicar com o CS, usando o protocolo UDP, para pedir a lista de conteúdos disponíveis em todos os SS. Essa lista deve ser mostrada ao utilizador, com numeração dos conteúdos para facilidade de selecção. Na sua resposta o CS identifica também qual o endereço IP e porto TCP do servidor de conteúdos (SS) que este cliente deve a contactar para descarregar conteúdos que pretenda obter.
- *`retrieve`* *content_name* – a aplicação de cliente deve comunicar em TCP com o SS (previamente indicado pelo CS em resposta ao comando *`list`*), transferir o conteúdo *content_name* para a directoria local e avisar o utilizador que a transferência foi efectuada.
- *`upload`* *content_name* – a aplicação de cliente deve comunicar com o CS em TCP para fazer a transferência do conteúdo *content_name* que se encontra na directoria local para o CS. O CS deve actualizar a sua lista de conteúdos disponíveis e fazer a sincronização com todos os servidores SS disponíveis.
- *`exit`* – a aplicação de cliente termina a sua execução.

2.2 Servidor Central: *Central Server* (CS)

O servidor central, *Central Server* (CS), é invocado usando o comando:

`CS` *`[-p CSport]`*,

em que:

`CSport` é o porto bem-conhecido no qual os servidores CS aceitam pedidos por parte de utilizadores, quer em UDP quer em TCP. Este argumento é opcional. Em caso de omissão, assume o valor 58000+NG, onde NG corresponde ao número do grupo.

Ao ser invocado o programa do servidor central, este começa por disponibilizar dois servidores, ambos com porto bem-conhecido *CSport*, um suportado em UDP onde escuta pedidos de utilizadores, outro suportado em TCP que pode ser usado para receber novos conteúdos.
O servidor CS funcionando em UDP pode atender instruções de tipo *list* por parte do utilizador. O servidor CS deve devolver ao utilizador a listagem dos conteúdos disponíveis no sistema, bem como o endereço IP e número de porto TCP do servidor SS mais "próximo" do cliente e que deverá posteriormente ser contactado pelo cliente em TCP (na sequência do comando *retrieve*) para efectuar a transferência do conteúdo pretendido. A escolha do servidor SS será feita consultando uma tabela previamente criada.
O servidor CS funcionando em TCP pode atender pedidos do tipo *upload* em que a aplicação de cliente transfere em TCP um novo conteúdo para o CS, que deve actualizar a sua lista de conteúdos e de seguida replicar este conteúdo para todos os servidores SS disponíveis.
O servidor CS tem uma lista dos endereços IP e dos portos TCP onde estão activos servidores SS.

Os servidores CS ecoam para o ecrã o tipo de pedido recebido e a identificação da máquina e porto originários desse pedido.
Cada pedido recebido de um utilizador é imediatamente respondido.

2.3 Servidor de Conteúdos: *Storage Server* (SS)

Um servidor de conteúdos, *Storage Server* (SS), é invocado com o comando:

`SS [-p SSport]`,

em que

SSport é o porto bem-conhecido no qual um servidor de conteúdos aceita pedidos provenientes dos utilizadores. Este argumento é opcional. Em caso de omissão, assume o valor 59000.

Logo após a invocação de um servidor SS, este disponibiliza na máquina em que é lançado um servidor suportado em TCP, com porto bem-conhecido *SSport*.
O servidor SS aceita pedidos de utilizadores para lhes enviar um determinado ficheiro. Nesse caso o servidor ecoa para o ecrã a identificação do conteúdo solicitado e a identificação da máquina e porto originários do pedido.

Este servidor pode também aceitar ficheiros que lhe sejam enviados pelo servidor central. Após receber um novo ficheiro deve ecoar para o ecrã a identificação do conteúdo e que a sua origem é o servidor central.

3. Especificação dos Protocolos de Comunicação

3.1 Protocolo *utilizador – servidor CS* em UDP

Quando executado o programa de utilizador (*user*), a interação inicial com o servidor de conteúdos usa a camada de transporte UDP, na sequência da instrução `list`.
O protocolo da camada de aplicação entre o utilizador e a componente **UDP** do servidor CS contempla os seguintes pedidos e respostas:

a) `LST`
Na sequência do comando `list` o utilizador faz um pedido ao servidor CS de listagem dos conteúdos presentemente disponíveis em todos os servidores SS.

b) `AWL IPSS portSS` n_T *F1 F2 … Fn*$_T$
Em resposta a um pedido `LST` o servidor CS responde em UDP indicando o endereço IP (`IPSS`) e o número do porto TCP (`portSS`) onde o *Storage Server* (SS) que deve ser contactado está disponível, o número (n_T) e a lista dos conteúdos disponíveis (`F1 F2 …` Fn_T), sendo `Fn` o nome do ficheiro número `n`. A resposta é enviada pelo servidor CS logo após a receção do pedido.
Se o pedido `LST` não puder ser atendido (por exemplo ficheiro vazio) a resposta será `EOF`. Se o pedido `LST` estiver mal formulado a resposta será: `ERR`.

Para simplificar pode considerar que n_T deverá ser no máximo 30, e que o nome de cada ficheiro (`Fn`) deverá conter um máximo de 20 caracteres.

Cada mensagem de pedido ou resposta termina sempre com o carácter "`\n`".

3.2 Protocolo *utilizador – servidor CS* em TCP

Quando o utilizador pretende carregar um novo ficheiro para o sistema, usando a instrução `upload`, a sua aplicação deve iniciar uma ligação TCP com o servidor central. Deve começar por verificar se esse ficheiro já existe no servidor, ou se pode fazer o correspondente *upload*.
O protocolo da camada de aplicação entre o utilizador e a componente **TCP** do servidor CS contempla os seguintes pedidos e respostas:

a) `UPR` *Fn*
O utilizador faz o pedido de verificação se já existe um ficheiro de nome *Fn* no servidor.

b) `AWR` *status*
Em resposta a um pedido `UPR` o servidor CS caso já exista um ficheiro com esse nome responde com `dup` no campo `status`. Se o pedido ainda não existir no servidor responde com `new` no campo `status`. Se o pedido estiver mal formulado responde com `ERR` no campo `status`.

c) `UPC` *size data*
O utilizador faz o envio do ficheiro previamente indicado no pedido `UPR`, com dimensão `size`, em Bytes, e seguido dos dados correspondentes.

d) `AWC` *`status`*
Em resposta a um pedido `UPC` o servidor CS responde indicando se o pedido estiver bem formulado com `ok` no campo `status`. Se o pedido não puder ser processado responde com `ERR` no campo `status`.

Cada pedido ou resposta termina com o carácter "`\n`".

3.3 Protocolo *utilizador – servidor SS*

Quando o utilizador pretende descarregar um ficheiro, na sequência da instrução `retrieve`, a aplicação deve iniciar uma ligação TCP com o servidor SS identificado previamente (na sequência da consulta ao servidor CS desencadeada pelo comando `list`).
O protocolo da camada de aplicação, suportado em TCP, entre o utilizador e o servidor SS contempla os seguintes pedidos e respostas:

a) `REQ` *`Fn`*
O utilizador faz o pedido do ficheiro de nome *`Fn`* ao SS.

b) `REP` *`status size data`*
Em resposta a um pedido `REQ` o servidor SS responde indicando se o pedido estiver bem formulado com `ok` no campo `status`, seguido da dimensão (`size`) do campo de dados e dos próprios dados. Se o pedido não puder ser processado responde com `ERR` no campo `status`.

Cada pedido ou resposta termina com o carácter "`\n`".

3.4 Protocolo *servidor CS – servidor SS*

Na sequência do carregamento de um novo ficheiro para o servidor CS (por utilização da instrução `upload`), o servidor CS deve iniciar uma ligação TCP com cada um dos servidores SS registados para lhe enviar uma cópia do novo ficheiro.
O protocolo da camada de aplicação, suportado em TCP, entre o servidor CS e cada servidor SS contempla os seguintes pedidos e respostas:

a) `UPS` *`Fn size data`*

O servidor CS faz o envio do ficheiro de nome *`Fn`*, com dimensão `size`, em Bytes, e seguido dos dados correspondentes.

b) `AWS` *`status`*

Em resposta a um pedido `UPS` o servidor SS responde indicando se o pedido estiver bem formulado com `ok` no campo `status`. Se o pedido não puder ser processado responde com `ERR` no campo `status`.

Cada pedido ou resposta termina com o carácter "`\n`".

4. Desenvolvimento

4.1 Ambiente de desenvolvimento e teste

Deve garantir que o seu código compila e executa corretamente no ambiente de desenvolvimento disponível no laboratório LT5.

4.2 Programação

Baseie a operação do seu programa no seguinte conjunto de chamadas de sistema:

- Nome da máquina: `gethostname()`.
- Endereço IP de uma máquina remota a partir do seu nome: `gethostbyname()`.
- Gestão de um servidor UDP: `socket()`, `bind()`, `close()`.
- Gestão de um cliente UDP: `socket()`, `close()`.
- Comunicação UDP: `sendto()`, `recvfrom()`.
- Gestão de um cliente TCP: `socket()`, `connect()`, `close()`.
- Gestão de um servidor TCP: `socket()`, `bind()`, `listen()`, `accept()`, `close()`.
- Comunicação TCP: `write()`, `read()`.
- Utilização de servidores concorrentes usando a chamada de sistema `fork()`.

4.3 Notas de implementação

O código desenvolvido deve estar convenientemente estruturado e comentado.
As chamadas de sistema `read()` e `write()` podem ler e escrever, respetivamente, um numero de bytes inferior ao que lhes foi solicitado – deve garantir que ainda assim a sua implementação funciona corretamente.
Quer o processo cliente quer o processo servidor devem terminar graciosamente pelo menos nas seguintes situações de falha:

- mensagens do protocolo erradas vindas da entidade par correspondente;
- condições de erro das chamadas de sistema.

5 Bibliografia

- W. Richard Stevens, Unix Network Programming: Networking APIs: Sockets and XTI (Volume 1), 2a edição, Prentice-Hall PTR, 1998, ISBN 0-13-490012-X, capítulo 5.
- D. E. Comer, Computer Networks and Internets, 2a edição, Prentice Hall, Inc,
- 1999, ISBN 0-13-084222-2, capítulo 24.
- Michael J. Donahoo, Kenneth L. Calvert, TCP/IP Sockets in C: Practical Guide for Programmers, Morgan Kaufmann, ISBN 1558608265, 2000
- Manual on-line, comando man
- Code Complete - http://www.cc2e.com/
- http://developerweb.net/viewforum.php?id=70

6 Entrega do Projecto

6.1 Código

O código a entregar é composto pelos ficheiros fonte dos programas implementando o *utilizador*, o *servidor CS* e o *servidor SS*, bem como a correspondente *Makefile*.

6.2 Submissão

A entrega do trabalho é feita por e-mail ao docente de laboratório, **até dia 10 de Outubro de 2014, às 20h**.
Deve criar um único ficheiro de arquivo `zip` com todos os ficheiros fonte e outros ficheiros necessários à execução das aplicações. O arquivo deve estar preparado para ser aberto para o diretório corrente e compilado com o comando `make`. O nome do ficheiro submetido deve ter o seguinte formato: `proj<número do grupo>.zip`

6 Dúvidas

Encoraja-se o esclarecimento de dúvidas junto do docente nos horários previstos para esse efeito.