Redes de Computadores

1º Semestre 2013/2014

Programação usando a interface de *Sockets*

"Storage Server"

1. Introdução

Pretende-se desenvolver uma aplicação de armazenamento e acesso a conteúdos composta por dois níveis de servidores. Um utilizador começará por interagir com um dos vários servidores de primeiro nível, *List Server* (LS), onde se pode inteirar dos conteúdos disponíveis. Cada LS conterá informação sobre um dado assunto, por exemplo, notícias nacionais, notícias de desporto, imagens de natureza, etc. Assume-se que o utilizador conhece os URL onde estão disponíveis servidores LS.
A interação com um LS permite listar os conteúdos disponíveis e, ao escolher um deles, obter a identificação do servidor de conteúdos, *Storage Server* (SS), onde o conteúdo desejado está disponível. A aplicação do utilizador fará então a transferência desse conteúdo para a diretoria previamente selecionada.
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*.
A comunicação entre um utilizador e um servidor LS, deve ser suportada na camada de transporte UDP, devendo a interface de *sockets* ser programada para esse efeito.
A transferência de conteúdos entre um servidor SS e a aplicação do utilizador deve ser suportada na camada de transporte TCP.

2. Especificação

2.1 Utilizadores

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

`user [-n LSname] [-p LSport],`

em que

- `-LSname` é o nome da máquina que aloja o servidor LS que se pretende contactar. Este argumento é opcional. Em caso de omissão, assume-se que o servidor está a correr na própria máquina.
- `-LSport` é o porto bem-conhecido no qual o servidor LS que se pretende contactar aceita pedidos por parte de utilizadores. 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 comunica (usando o protocolo UDP) com o servidor de primeiro nível *List Server* (LS) selecionado e aguarda a resposta deste com a identificação do assunto a que este LS se dedica e a listagem dos conteúdos disponíveis.
Depois de obter a resposta inicial do servidor LS o utilizador deve indicar o número do conteúdo que pretende obter (ou indicar o número 0 para terminar a aplicação).
A aplicação do cliente deve então comunicar (usando o protocolo UDP) novamente com o servidor LS indicando qual o conteúdo pretendido e aguardar a resposta com a identificação do *Storage Server* (SS) onde o conteúdo desejado está disponível.
Em seguida deve comunicar (usando o protocolo TCP) com o SS identificado para requisitar a transferência do conteúdo desejado para uma diretoria previamente selecionada.

2.2 Servidor de Primeiro Nível: *List Server* (LS)

O servidor de primeiro nível, *List Server* (LS), é invocado usando o comando:

`LS [-p LSport],`

em que

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

Logo após o lançamento de um servidor LS, este começa a sua operação disponibilizando um servidor suportado em UDP, com porto bem-conhecido `LSport`, onde escuta pedidos de utilizadores. Os pedidos podem ser de dois tipos: (i) listagem dos conteúdos disponíveis; (ii) identificação do *Storage Server* (SS) onde o conteúdo desejado está disponível.
Cada pedido recebido de um utilizador é imediatamente respondido.
O servidor LS ecoa para o ecrã o tipo de pedido recebido e a identificação da máquina e porto originários desse pedido.

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

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

`SS [-p SSport]`,

em que

–SSport é o porto bem-conhecido no qual o 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*, para aceitação de pedidos de conteúdos provenientes dos utilizadores.
O servidor SS ecoa para o ecrã a identificação do conteúdo solicitado e a identificação da máquina e porto originários do pedido.

2.4 Protocolo utilizador – servidor LS

Quando executado o programa de utilizador (*user*), inicia-se a interação com o servidor de primeiro nível, *List Server* (LS), cujo nome da máquina e porto bem-conhecido são especificados como argumentos. A interação usa a camada de transporte UDP.
O protocolo da camada de aplicação entre o utilizador e o servidor LS contempla os seguintes comandos e respostas:

a) `RQT`

O utilizador faz um pedido de listagem dos conteúdos disponíveis ao servidor LS. Este comando é enviado pelo processo cliente logo após a invocação do programa de utilizador (*user*).

b) `AWT` *subject* n_T *T1 T2 … Tn*$_T$

Em resposta a um pedido `RQT` o servidor LS responde indicando o assunto (`subject`) a que este LS se dedica, o número (n_T) e a lista dos conteúdos disponíveis (`T1 T2 …` Tn_T), sendo `Tn` uma palavra descrevendo o tópico número `n`. A resposta é enviada logo após a receção do pedido.
Se o pedido `RQT` estiver mal formulado a resposta será: `ERR`.

Em que (`subject`) deverá ser representado com um máximo de 30 caracteres, (n_T) deverá ser no máximo 30, e cada palavra (`Tn`) deverá conter um máximo de 20 caracteres para descrever o tópico número `n`.

c) `RQC` *n*

O utilizador pede ao servidor LS a identificação do *Storage Server* (SS) onde o conteúdo desejado (identificado pelo número *n*) está disponível.
Este comando é enviado logo após a escolha do utilizador.

d) `AWC` *`Tn IP port`*

Em resposta a um pedido `RQC n` o servidor LS responde indicando o tópico solicitado (*`Tn`*) e o endereço IP e o número do porto TCP onde o *Storage Server* (SS) com o conteúdo desejado está disponível. A resposta é enviada logo após a recepção do pedido.
Se o pedido `RQC` estiver mal formulado a resposta será: `ERR`.

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

2.5 Protocolo utilizador – servidor SS

O programa do utilizador, após obter a identificação do *Storage Server* (SS) onde o conteúdo desejado está disponível, comunica (usando o protocolo TCP) com o SS para requisitar a transferência desse conteúdo.
O protocolo de comunicação entre o utilizador e o servidor SS contempla os seguintes comandos e respostas:

a) `REQ` *`Tn`*

O utilizador faz um pedido do conteúdo *`Tn`* 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 `status = ok` seguido da dimensão (`size`) do campo de dados e dos próprios dados. Se o pedido não puder ser processado responde com `status = nok`.

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

3. Desenvolvimento

3.1 Ambiente de desenvolvimento e teste

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

3.2 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()`.
- Considere a possibilidade de usar um servidor de conteúdos concorrente (usando a chamada de sistema `fork()`).

3.3 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.

4 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

5 Entrega do Projecto

5.1 Código

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

5.2 Submissão

A entrega do trabalho é feita por e-mail ao docente de laboratório, **até dia 11 de Outubro de 2013, 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.