RCI Dynamic Directory

Redes de Computadores e Internet
2º Semestre 2013/2014
Projeto de Laboratório

1. Descrição da aplicação

Pretende-se desenvolver uma aplicação Dynamic Directory (**dd**) que permite manter e consultar uma base-de-dados distribuída de nomes de utilizadores. Esta base-de-dados é consultada quando um utilizador pretende estabelecer uma chamada com outro para com ele trocar mensagens de texto. Os nomes dos utilizadores têm uma hierarquia em dois níveis, nome próprio e apelido, sendo da forma **name.surname**. Existe um Servidor de Apelidos (SA) fornecido pelo corpo docente que responde por todos os apelidos registados. A aplicação **dd** é desenvolvida pelos alunos. Ela contém um Servidor UDP de Nomes Próprios (SNP) que responde por todos os nomes próprios registados que partilham o mesmo apelido do do utilizador. A aplicação **dd** contém também um Servidor TCP de Chamadas (SC) para receber chamadas. O SNP e o SC associados ao utilizador **A.X** são denotados por SNP(**A.X**) e SC(**A.X**), respetivamente.

A próxima secção explicita como se processa a consulta da localização do SC de um nome. As duas secções seguintes explicitam como se mantem a base-de-dados de nomes atualizada com a adesão de um novo utilizador, Secção 1.2, e a saída de um utilizador registado, Secção 1.3.

1.1 Consulta de um nome

O SA tem uma base-de-dados com a localização (endereço IP e porto) de um, *e apenas de um*, SNP associado a cada apelido. Esse SNP é dito *autorizado* para o apelido em causa. O SNP autorizado para o apelido **X** é denotado por SNP(**X**). Cada SNP tem uma base-de-dados com a localização de *todos* os SNPs e de *todos* os SCs que com ele partilham o mesmo apelido. Suponha que o utilizador **A.X** pretende determinar a localização de SC(**B.Y**). Distinguimos dois casos:

- Se **A.X** e **B.Y** têm o mesmo apelido, **Y** = **X**, e **B.Y** estiver registado, então SNP(**A.X**) terá na sua base-dados a localização de SC(**B.Y**).
- Caso contrário, se **A.X** e **B.Y** têm apelidos diferentes, **X** ≠ **Y**, então **A.X** interroga o SA. Se houver pelo menos um nome registado com apelido **Y**, então o SA envia a SNP(**A.X**) a localização do SNP autorizado para **Y**, SNP(**Y**). Seguidamente, **A.X** interroga SNP(**Y**). Se o nome **B.Y** estiver registado em SNP(**Y**), então o SNP(**Y**) envia a SNP(**A.X**) a localização de SC(**B.Y**).

1.2 Adesão de um utilizador

Suponha que o utilizador **A.X** adere à rede. Ele regista-se no SA. Se **A.X** for o primeiro utilizador de apelido **X** a registar-se, então SA toma SNP(**A.X**) como o SNP autorizado para o apelido **X**, SNP(**X**) = SNP(**A.X**), e envia a localização de SNP(**X**) a SNP(**A.X**). Distinguimos dois casos:

- Se SNP(**A.X**) e SNP(**X**) forem o mesmo, SNP(**A.X**) = SNP(**X**), então **A.X** fica a saber que o seu SNP é o SNP autorizado para **X**.
- Caso contrário, se SNP(**A.X**) não for SNP(**X**), SNP(**A.X**) ≠ SNP(**X**), então **A.X** regista-se em SNP(**X**) e dele toma conhecimento de localização de todos os SCs e SNPs de utilizadores com apelido **X**. De seguida, **A.X** regista-se em cada um dos SNPs de utilizadores com apelido **X**.

1.3 Abandono de um utilizador

Suponha que **A.X** abandona a rede. Distinguimos dois casos:

- Se **A.X** é o único utilizador registado com apelido **X**, então necessariamente SNP(**A.X**) é o SNP autorizado para **X**, SNP(**A.X**) = SNP(**X**). **A.X** tem apenas que apagar o seu registo em SA.
- Caso contrário, se há mais do que um utilizador registado com apelido **X**, então **A.X** apaga o seu registo de cada um dos SNP dos utilizadores com apelido **X**. Adicionalmente, se SNP(**A.X**) é o SNP autorizado para o apelido **X**, SNP(**A.X**) = SNP(**X**), então ele nomeia um outro utilizador **B.X** para ser aquele cujo SNP é autorizado, SNP(**B.X**) = SNP(**X**). Concretamente, **A.X** participa a SA e a SNP(**B.X**) que SNP(**X**) = SNP(**B.X**).

2. Especificação

Cada grupo de dois alunos deve concretizar a aplicação **dd** compreendendo os elementos seguintes:

- Uma interface de utilizador.
- Um protocolo de adesão e abandono da rede.
- Um protocolo de consulta da localização de um servidor de chamadas.
- Um protocolo de chamada.

2.1 Invocação dd

A aplicação **dd** é invocada com o comando

```
dd name.surname IP [-t talkport] [-d dnsport] [-i saIP] [-p saport]
```

em que

- **`name.surname`** é o nome do utilizador associado ao **`dd`**, constituído por nome próprio (**`name`**) e apelido (**`surname`**). Este parâmetro é obrigatório.
- **`IP`** é o endereço IP da máquina que aloja o **`dd`**. Este parâmetro é obrigatório.
- ***`saIP`*** e ***`saport`*** são o endereço IP e o porto UDP bem-conhecido do servidor de apelidos. Por omissão, ***`saIP`*** é o endereço IP da máquina **`tejo.ist.utl.pt`** e ***`saport`*** toma o valor **`58000`**.
- ***`dnsport`*** é o porto UDP bem-conhecido do servidor de nomes próprio. Deverá existir um valor por omissão.
- ***`talkport`*** é o porto TCP bem-conhecido do servidor de chamadas. Deverá existir um valor por omissão.

Em resultado da invocação, a aplicação disponibiliza um servidor TCP no porto ***`talkport`*** para receção de chamadas e um servidor UDP no porto ***`dnsport`*** para receção de pedidos à base-de-dados de nomes.

2.2 A interface de utilizador

A interface de utilizador aceita os comandos seguintes:

- **`join`**
  O utilizador pretende registar-se na base-de-dados.
- **`leave`**
  O utilizador pretende apagar o seu registo da base-de-dados.
- **`find`** ***`name.surname`***
  O utilizador pretende a localização do servidor de chamadas com nome ***`name.surname`***.
- **`connect`** ***`name.surname`***
  O utilizador pretende estabelecer uma chamada com o utilizador ***`name.surname`***.
- **`disconnect`**
  O utilizador pretende desligar-se da chamada na qual está a participar.
- **`message`** ***`string`***
  O utilizador pretende enviar a mensagem ***`string`*** ao seu interlocutor em chamada.
- **`exit`**
  O utilizador fecha a aplicação.

2.3 Protocolo de adesão e abandono da rede

O protocolo de adesão e abandono da rede compreende as seguintes mensagens:

- **`REG`** ***`name.surname;ip;talkport;dnsport`***
  A aplicação regista o nome ***`name.surname`*** no SA ou num SNP, oferecendo a localização do seu SC, (***`ip`***, ***`talkport`***), e do seu SNP, (***`ip`***, ***`dnsport`***).
- **`DNS`** ***`name.surname;authip;authdnsport`***

O SA ou um SNP indica a localização, (***authip***, ***authdnsport***), do SNP autorizado para o apelido ***surname***.

- **LST**
  name.surname;ip1;talkport1;dnsport1
  …
  name.surname;ipN;talkportN;dnsportN

  Em resposta ao pedido de registo de um utilizador de apelido ***surname***, o SNP autorizado entrega uma lista composta por várias linhas, cada uma das quais terminada com o caráter '\n'. A primeira linha tem a sequência **LST**. Cada uma das linhas seguintes identifica um nome constante da base-de-dados do SNP, com a localização do respetivo SC e SNP. A lista termina numa linha em vazia. Se o utilizador ***name.surname***, já estiver registado, então o SNP não envia nenhum nome.
- **UNR** ***name.surname***
  A aplicação notifica o SA ou um SNP que o utilizador abandonou a rede.
- **OK**
  Um SNP envia confirmação de que recebeu um pedido.

2.4 Protocolo de consulta da localização de um servidor de chamadas

O protocolo de consulta de um servidor de chamadas envolve as mensagens seguintes:

- **QRY** ***name.surname***
  Um SNP interroga o SA ou outro SNP sobre a localização do servidor de chamadas de ***name.surname***.
- **FW** ***name.surname;authip;authdnsport***
  O SA responde com a localização, (***authip***, ***authdnsport***), do SNP autorizado para o apelido ***surname***. Se não houver nenhum nome registado com apelido ***surname***, então a resposta é apenas **FW**.
- **RPL** ***name.surname;ip;talkport***
  Um SNP responde com a localização, (***ip***, ***talkport***), do SC de ***name.surname***. Se ***name.surname*** não está registado, então a resposta é apenas **RPL**.

2.5 Protocolo de chamada

Um utilizador inicia uma chamada com outro estabelecendo uma sessão TCP para o servidor de chamadas deste último. A chamada pode ser terminada por qualquer dos utilizadores fechando a sessão TCP. Durante a chamada as mensagens são enviadas com o comando seguinte:

- **MSS** ***name.surname;string***
  O utilizador de nome ***name.surname*** envia uma mensagem ao utilizador com o qual está conectado.

3. Desenvolvimento

Cada grupo de alunos deve adquirir a destreza necessária sobre programação em redes para realizar a aplicação proposta.

Para o desenvolvimento global do projeto, sugerem-se os passos seguintes:

i. Assuma que todos os utilizadores têm apelidos distintos. Realize o cliente que regista o utilizador no SA alojado na máquina **`tejo.ist.utl.pt`**. Comandos **`join`**, **`leave`** e **`exit`**.
ii. Assuma que todos os utilizadores têm apelidos distintos. Realize o protocolo de consulta da localização de um SC. Comando **`find`**.
iii. Assuma que todos os utilizadores têm apelidos distintos. Concretize uma chamada. Comandos **`connect`**, **`message`** e **`disconnect`**.
iv. Concretize a adesão para o caso em que mais do um utilizador pode ter o mesmo apelido.
v. Concretize o abandono de um utilizador.

Comente e teste o seu código à medida que o desenvolve. Note que o projecto será compilado e executado pelo corpo docente **apenas** no ambiente de desenvolvimento disponível no laboratório, constituído pelos elementos seguintes:

- Compilador: `gcc` versão 4.4.3
- Depurador: `ddd` versão 3.3.11
- `glibc`: versão 2.11.1

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

- Leitura de informação do utilizador para a aplicação: `fgets()`;
- Decomposição de strings em tipos de dados e vice-versa: `sprintf()`, `sscanf()`;
- 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()`;
- Multiplexagem de informação: `select()`.

Quer os clientes quer os servidores devem terminar graciosamente, pelo menos nas seguintes situações de falha:

- Mensagens do protocolo erradas vindas da entidade par correspondente;
- Sessão TCP do cliente ou do servidor fechada de forma imprevista;
- Condições de erro nas chamadas de sistema.

4. Bibliografia

- José Sanguino, A Quick Guide to Networking Software, 2013
- 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
- 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`

5. Entrega do Projeto

O código a entregar deve ser guardado num arquivo `zip` contendo o código fonte da **`dd`** e a respetiva `makefile`. A entrega do trabalho é feita por e-mail ao seu docente de laboratório. O arquivo deve estar preparado para ser aberto para o diretório corrente e será compilado com o comando `make`. O arquivo submetido deve ter o seguinte formato: `proj<número_do_grupo>.zip (ex: proj07.zip)`. A data de entrega é 30/03/2014.