RCI Secure Chat

Redes de Computadores e Internet
2º Semestre 2015/2016
Projeto de Laboratório

1. Descrição do projeto

Pretende-se desenvolver um sistema de comunicação *RCI secure chat* com o qual utilizadores identificados univocamente por nomes completos, compostos por nome próprio seguido de apelido, trocam mensagens de texto (vulgo *chat*) de forma segura.

O sistema de comunicação compreende duas aplicações: um diretório distribuído com correspondências entre nomes e as suas localizações; uma aplicação para a troca de mensagens de texto entre dois utilizadores precedida de autenticação mútua.

A localização de um nome é da forma *IPaddress;port*, em que *IPaddress* é o endereço IP da máquina onde está alojado o processo servidor associado a esse nome e *port* é o porto do dito processo. À imagem do diretório *Domain Name System* (DNS) da Internet, o diretório do sistema *RCI secure chat* tem as correspondências entre nomes e localizações distribuídas por vários servidores de nomes. Para cada apelido existe um servidor de nomes próprios que contem todas as correspondências entre nomes completos que partilham esse apelido e as suas localizações. Existe um único servidor de apelidos que contem todas as correspondências entre apelidos e a localização dos servidores de nomes próprios a eles associados. O diretório distribuído requer dois protocolos de comunicação: um para o manter atualizado; outro para consultar a localização de um nome completo. Ambos operam sobre UDP, pelo que os portos associados à localização dos servidores de nomes são portos UDP.

O servidor de apelidos é fornecido pelo corpo docente. Cada grupo de alunos tem que desenvolver a aplicação **snp** que cria um servidor de nomes próprios associado a um apelido. O protocolo de atualização é o seguinte. Quando um servidor de nomes próprios, associado ao apelido Z, é lançado, ele regista Z e a sua localização no servidor de apelidos. Deve ser dada a possibilidade de terminação do servidor de nomes próprios a partir da consola. Quando tal acontece, o registo do apelido é apagado do servidor de apelidos. Quando um utilizador, de nome completo A.Z, entra no sistema, ele regista A.Z e a sua localização no servidor de nome próprios Z. O utilizador apaga o seu registo imediatamente antes de sair do sistema.

O protocolo de consulta de um nome completo é o seguinte. Um utilizador, de nome completo A.Z, que queira saber a localização de outro, de nome completo B.Y, delega

essa descoberta no seu servidor de nome próprios Z. Se Y = Z, então o servidor de nomes próprios Z entrega a localização de B.Y a A.Z. Caso contrário, se Y ≠ Z, então o servidor de nomes próprios Z interroga o servidor de apelidos que lhe retorque com a localização do servidor de nomes próprios Y. De seguida, o servidor de nomes próprios Z interroga o servidor de nomes próprios Y, obtendo deste a localização de B.Y. Por fim, o servidor de nomes próprios Z entrega a localização de B.Y a A.Z.

A aplicação para troca de mensagens de texto suporta-se na descoberta da localização de nomes completos providenciada pelo protocolo de consulta do diretório distribuído. As mensagens de texto trocadas entre dois utilizadores estão envoltas em chamadas, cada uma das quais concretizada numa sessão TCP, pelo que os portos constantes da localização de um nome completo são portos TCP. Cada grupo de alunos tem que desenvolver a aplicação **schat** para a troca de mensagens de texto entre dois utilizadores. Quando um utilizador, de nome completo A.Z, pretende trocar mensagens com outro utilizador, de nome completo B.Y, ele começa por solicitar ao servidor de nomes próprios Z que lhe descubra a localização de B.Y. Uma vez na posse da localização de B.Y, A.Z estabelece uma chamada autenticada com B.Y, podendo então trocar mensagens de texto com B.Y.

O protocolo de chamada autenticada entre A.Z e B.Y consiste na seguinte troca de mensagens. A.Z inicia o estabelecimento de uma chamada com B.Y. De seguida, A.Z identifica-se, enviando o seu nome completo a B.Y. B.Y tem agora que autenticar A.Z. Para isso, B.Y envia um desafio a A.Z que consiste num byte por ele escolhido aleatoriamente. A.Z encripta esse byte com a chave secreta partilhada com B.Y e devolve-o a B.Y. Quando B.Y recebe o byte de A.Z, ele verifica se esse byte é realmente o byte enviado como desafio encriptado com a chave secreta partilhada com A.Z. Em caso afirmativo, B.Y autenticou A.Z. Resta agora A.Z autenticar B.Y por um processo simétrico: A.Z envia um desafio a B.Y e valida a resposta encriptada de B.Y. Se a autenticação mútua for concluída com sucesso, então os dois utilizadores podem trocar mensagens de texto. Posteriormente, qualquer dos utilizadores pode tomar a iniciativa de terminar a chamada. Se uma das duas autenticações falhar, então o utilizador que não consegue autenticar a outro termina de imediato a sessão TCP.

A chave secreta é um número entre 0 e 255, representado num byte, que serve para indexar uma permutação dos números de 0 a 255. A chave de encriptação é guardada num ficheiro.

As secções seguintes dão conta dos comandos nas interfaces de utilizador e do formato das mensagens dos protocolos das aplicações **snp** e **schat**.

2. Especificação da aplicação snp

A aplicação **snp** é invocada da seguinte forma.

```
snp –n surname –s snpip -q snpport [-i saip] [-p saport]
```

em que:

- ***surname*** é um apelido.
- ***snpip*** é o endereço IP da máquina que aloja o **snp**.
- ***snpport*** é o porto UDP (servidor) associado ao processo **snp**.
- ***saip*** e ***saport*** são, respetivamente, o endereço IP e o porto UDP do servidor de apelidos. Estes argumentos são opcionais. Por omissão, ***saip*** deve tomar o endereço IP da máquina **`tejo.tecnico.ulisboa.pt`** e ***saport*** deve tomar o valor **`58000`**.

Em resultado da invocação, a aplicação disponibiliza um servidor de nomes próprios associado ao apelido ***surname*** no porto UDP ***snpport***. A especificação da aplicação **snp** compreende uma interface de comando, o protocolo que garante a atualização do diretório distribuído e o protocolo para consulta de um nome completo.

2.1 Interface de comando

A interface de comando aceita as seguintes diretivas.

- **`list`**
  Visualização das correspondências entre nomes completos e suas localizações guardadas no servidor.
- **`exit`**
  Terminação do servidor.

2.2 Protocolo para atualização do diretório distribuído

O protocolo para atualização do diretório distribuído compreende as seguintes mensagens.

- **`SREG`** ***`surname;ip;snpport`***
  Mensagem enviada de um servidor de nomes próprios para o servidor de apelidos para registo do apelido ***surname***, fazendo-o corresponder ao endereço IP ***ip*** e ao porto UDP ***snpport***. Em resposta a esta mensagem, o servidor de nomes próprios espera receber **`OK`**, indicando que o registo foi efetuado, ou **`NOK`**, no caso contrário.
- **`REG`** ***`name.surname;ip;scport`***
  Mensagem enviada de um utilizador para um servidor de nomes próprios para registo do nome completo ***name.surname***, fazendo-o corresponder ao endereço IP ***ip*** e ao porto TCP ***scport***. Em resposta a esta mensagem, o servidor de

nomes próprios responde **OK**, se o registo for efetuado, ou **NOK,** no caso contrário.

- **SUNR** ***surname***
  Mensagem enviada de um servidor de nomes próprios para o servidor de apelidos solicitando a remoção do registo do apelido ***surname***. Em resposta a esta mensagem, o servidor de nomes próprios espera receber **OK**, indicando que a remoção foi efetuada com sucesso, ou **NOK**, no caso contrário.
- **UNR** ***name.surname***
  Mensagem enviada de um utilizador para um servidor de nomes próprios solicitando a remoção do registo do nome completo ***name.surname***. Em resposta a esta mensagem, o servidor de nomes próprios responde **OK**, se o registo existia e foi apagado com sucesso, ou **NOK**, no caso contrário.
- **OK**
  Mensagem de resposta que confirma a realização de um pedido.
- **NOK** ***string***
  Mensagem de resposta que indica que um pedido não foi realizado, acompanhada pelo texto ***string*** que informa a razão da não concretização do pedido.

2.3 Protocolo para consulta de um nome completo

O protocolo de consulta de um nome completo compreende as seguintes mensagens.

- **SQRY** ***surname***
  Mensagem enviada de um servidor de nomes próprios para o servidor de apelidos solicitando a localização do servidor de nomes próprios associado a ***surname***.
- **SRPL** ***surname;snpip;snpport***
  Mensagem enviada do servidor de apelidos para um servidor de nomes próprios informando que o servidor de nomes próprios ***surname*** está localizado no endereço IP ***snpip*** e porto UDP ***snpport***. A mensagem **SRPL** (sem parâmetros) indica que o último apelido que foi demandado não está registado no servidor de apelidos.
- **QRY** ***name.surname***
  Mensagem enviada de um utilizador para o seu servidor de nomes próprios, ou de um servidor de nomes próprios para outro, solicitando a localização do nome completo ***name.surname***.
- **RPL** ***name.surname;scip;scport***
  Mensagem enviada de um servidor de nomes próprios para um dos seus utilizadores, ou para outro servidor de nomes próprios, informando que o nome completo ***name.surname*** está localizado no endereço IP ***scip*** e porto TCP ***scport***. A mensagem **RPL** (sem parâmetros) indica que o último nome completo que foi demandado não foi descoberto pelo servidor de nomes próprios.

3. Especificação da aplicação schat

A aplicação **schat** é invocada da seguinte forma.

```
schat –n name.surname –i ip -p scport -s snpip -q snpport
```

em que:

- ***name.surname*** é o nome completo do utilizador, constituído pelo nome próprio ***name*** e o apelido ***surname***.
- ***ip*** é o endereço IP da máquina que aloja o **schat**.
- ***scport*** é o porto TCP (servidor) associado ao processo **schat**.
- ***snpip*** e ***snpport*** são o endereço IP e o porto UDP do servidor de nomes próprios associado ao apelido ***surname***.

Em resultado da invocação, a aplicação disponibiliza um servidor TCP para a receção de chamadas no porto ***scport***. A especificação da aplicação **schat** compreende uma interface de utilizador e o protocolo de chamadas autenticadas.

3.1 Interface de utilizador

A interface de utilizador aceita os seguintes comandos:

- **join**
  Registo no diretório.
- **leave**
  Remoção do registo no diretório.
- **find** ***name.surname***
  Consulta da localização do nome completo ***name.surname***. (Comando para teste e depuração.)
- **connect** ***name.surname keyfile***
  Estabelecimento de uma chamada com o utilizador de ***name.surname*** usando a chave secreta armazenada no ficheiro ***keyfile*** para autenticação.
- **message** ***string***
  Envio da mensagem de texto ***string***.
- **disconnect**
  Terminação da chamada em curso.
- **exit**
  Terminação da aplicação.

3.2 Protocolo de chamada autenticada

O protocolo de chamada autenticada faz uso das seguintes mensagens.

- **NAME** ***name.surname***

Mensagem enviada do originador da chamada ao destinatário desta identificando-o. Esta mensagem termina com o carácter '**`\n`**'.

- **`AUTH`** ***`byte`***
  Mensagem usada para o envio do byte ***`byte`***, representando um desafio ou uma resposta encriptada a um desafio.

4. Desenvolvimento

Cada grupo de alunos deve adquirir a destreza necessária sobre programação em redes para realizar o sistema de comunicação *RCI secure chat*.

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

i. Implemente o protocolo de atualização no **snp**. Realize o comando **`exit`** do **snp**.
ii. Implemente o protocolo de atualização no **`schat`**. Realize os comandos **`join`**, **`leave`** e **`exit`**.
iii. Realize o comando **`list`** do **snp**.
iv. Implemente o protocolo de consulta de um nome completo. Realize o comando **`find`**.
v. Implemente o estabelecimento de chamadas sem autenticação. Realize os comandos **`connect`** e **`disconnect`**.
vi. Altere a implementação anterior de forma a incluir a autenticação mútua dos dois intervenientes na chamada.

Comente e teste o seu código **enquanto** o desenvolve. Na avaliação, o projeto será compilado e executado pelo corpo docente **apenas** no ambiente de desenvolvimento disponível no laboratório.

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()`;
- Geração aleatória de números: `rand()`.

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

- Condições de erro nas chamadas de sistema.

- Mensagens do protocolo com formatação errónea;
- Sessão TCP terminada de forma imprevista;

5. 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`

6. Entrega do Projeto

O código a entregar deve ser guardado num arquivo `zip` contendo o código fonte do **snp** e do **schat** bem como a respetiva `makefile`. O arquivo deverá ser entregue por e-mail ao vosso docente de laboratório. Ele deve estar preparado para ser aberto para o diretório corrente podendo ser compilado sem erros com o comado `make`. O nome do arquivo é da seguinte forma: `proj<número_do_grupo>.zip`, em que `<número_do_grupo>` é o número do vosso grupo (exemplo, `proj07.zip`). A data de entrega é domingo, dia 3 de Abril, às 23:59.