RCI Reliable Message Board

Redes de Computadores e Internet
2º Semestre 2016/2017
Projeto de Laboratório

1. Descrição do projeto

Pretende-se desenvolver um sistema de comunicação *RCI Reliable Message Board* com o qual utilizadores publicam e acedem a mensagens de texto, interagindo através delas. Por uma questão de fiabilidade, as mensagens são guardadas simultaneamente em mais do que um servidor de mensagens O sistema deverá garantir a causalidade das mensagens: uma mensagem que é resposta a outra é visualizada depois desta por todos os utilizadores.

O corpo docente fornece um servidor de identidades, o qual mantem a identidade dos vários servidores de mensagens presentes no sistema. A identidade dum servidor de mensagens é composta pelo seu nome, o seu endereço IPv4, o porto no qual ele atende a pedidos vindos dos terminais e o porto no qual ele atende a pedidos de estabelecimento de sessão vindos de outros servidores de mensagens. Cada grupo de alunos deve desenvolver a aplicação **`rmb`** que permite a publicação e o acesso a mensagens a partir dum terminal. A aplicação suporta dois comandos de utilizador. O comando **`publish`** ***`message`*** faz a aplicação publicar o conteúdo de ***`message`*** num servidor de mensagens (que se encarregará de o replicar pelos outros servidores de mensagens). O comando **`show_latest_messages`** ***`n`*** faz a aplicação baixar as últimas ***`n`*** mensagens dum servidor de mensagens. O servidor de mensagens contactado para publicação de mensagens e acesso a mensagens deverá ser o mesmo, conquanto escolhido arbitrariamente pela aplicação. A aplicação suporta ainda o comando de gestão **`show_servers`** que a faz interrogar o servidor de identidades sobre a identidade dos vários servidores de mensagens presentes no sistema. A comunicação entre um terminal e o servidor de identidades e a comunicação entre um terminal e os servidores de mensagens são realizadas por UDP. Os alunos devem convencer-se que a aplicação **`rmb`** pode ser terminada sem aviso prévio aos servidores.

Cada grupo de alunos deverá também desenvolver a aplicação **`msgserv`** a ser executada em cada um dos servidores de mensagens. Quando a aplicação é invocada, ela: (i) regista o servidor de mensagens no servidor de identidades, sendo este registo refrescado periodicamente durante o tempo de vida da aplicação; (ii) interroga o servidor de mensagens sobre a identidade dos outros servidores de mensagens; e (iii) estabelece uma sessão TCP com cada um destes. Se o servidor de mensagens não for o

único presente no sistema, então ele obtém o conjunto de todas as mensagens já publicadas junto de um qualquer servidor de mensagens, à escolha da aplicação. Periodicamente, o servidor de identidades analisará os registos submetidos pelos servidores de mensagens, apagando os que não são refrescados. Os alunos devem convencer-se que a aplicação **`msgserv`** pode ser terminada sem aviso prévio ao servidor de identidades ou aos terminais.

Os atrasos na comunicação entre duas máquinas são imprevisíveis e variáveis no tempo. Para manter a causalidade das mensagens, cada servidor de mensagens gere um relógio lógico LC. O valor de LC é atribuído a cada mensagem recebida de um terminal, formando o tempo lógico desta, e incluído com a mensagem quando ela é difundida pelos outros servidores. O relógio lógico LC é inicializado a 0 e atualizado da seguinte forma: (i) quando uma mensagem é recebida de um terminal, $LC \leftarrow LC + 1$; (ii) quando uma mensagem com tempo lógico k é recebida de outro servidor, $LC \leftarrow \max(LC, k) + 1$. Os alunos devem convencer-se das duas propriedades seguintes resultantes do uso de relógios lógicos: (i) utilizadores diferentes podem visualizar as mensagens por ordem distinta; (ii) a causalidade das mensagens é respeitada em todas as visualizações dos utilizadores.

Para a gestão de um servidor de mensagens, a aplicação **`msgserv`** providencia dois comandos. O comando **`show_servers`** faz a aplicação mostrar no ecrã a identidade de todos os servidores de mensagens com os quais o servidor tem uma sessão TCP. O comando **`show_messages`** faz a aplicação mostrar no ecrã todas as mensagens que tem guardadas.

2. Especificação da aplicação rmb

A aplicação **`rmb`** é invocada da seguinte forma.

`rmb [-i siip] [-p sipt]`

em que:

- ***`siip`*** é o endereço IP do servidor de identidades fornecido pelo corpo docente. Este argumento é opcional. Por omissão, ***`siip`*** deve tomar o endereço IP da máquina **`tejo.tecnico.ulisboa.pt`**.
- ***`sipt`*** é o porto UDP do servidor de identidades fornecido pelo corpo docente. Este argumento é opcional. Por omissão, ***`sipt`*** deve tomar o valor **`59000`**.

A especificação da aplicação **rmb** compreende uma interface de utilizador, o protocolo de comunicação com o servidor de identidades e o protocolo de comunicação com os servidores de mensagens.

2.1 Interface de utilizador

A interface de utilizador aceita os comandos seguintes.

- **`show_servers`**
  Obtenção das identidades de todos os servidores de mensagens registados.
- **`publish`** ***`message`***
  Publicação da mensagem de texto ***`message`*** nos servidores de mensagens. Uma mensagem de texto tem no máximo 140 carateres.
- **`show_latest_messages`** ***`n`***
  Baixamento das últimas ***`n`*** mensagens guardadas nos servidores de mensagens.
- **`exit`**
  Terminação da aplicação.

2.2 Protocolo de comunicação entre terminais e o servidor de identidades

O protocolo de comunicação entre terminais e o servidor de identidades compreende as duas mensagens protocolares seguintes.

- **`GET_SERVERS`**
  Mensagem enviada dum terminal para o servidor de identidades solicitando a identidade de todos os servidores de mensagens registados.
- **`SERVERS\n`**`(`***`name;ip;upt;tpt`***`\n`**`)*`**
  Mensagem enviada do servidor de identidades para um terminal com a lista das identidades de todos os servidores de mensagens registados. A identidade de um servidor de mensagens é composta pelo seu nome, ***`name`***, o seu endereço IP, ***`ip`***, o porto UDP no qual ele atende a pedidos vindos dos terminais, ***`upt`***, e o porto TCP no qual ele atende a pedidos de estabelecimento de sessão vindos de outros servidores de mensagens, ***`tpt`***. As identidades são terminadas pelo carácter **`\n`**. Um terminal não faz uso do porto no qual o servidor de mensagens atende a pedidos de estabelecimento de sessão vindos de outros servidores de mensagens.

2.3 Protocolo de comunicação entre terminais e servidores de mensagens

O protocolo de comunicação entre terminais e servidores de mensagens compreende as duas mensagens protocolares seguintes.

- **`PUBLISH`** ***`message`***
  Mensagem enviada dum terminal para um servidor de mensagens para publicação da mensagem de texto ***`message`***.
- **`GET_MESSAGES`** ***`n`***
  Mensagem enviada dum terminal para um servidor de mensagens solicitando as últimas ***`n`*** mensagens nele guardadas de acordo com os tempos lógicos.

- **MESSAGES\n(*message*\n)***
  Mensagem enviada dum servidor de mensagens para um terminal com uma lista de mensagens, ordenadas pelos tempos lógicos.

3. Especificação da aplicação msgserv

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

msgserv –n *name* –j *ip* -u *upt* –t *tpt* [-i *siip*] [-p *sipt*] [–m *m*] [–r *r*]

em que:

- ***name*** é o nome do servidor de mensagens.
- ***ip*** é o endereço IP do servidor de mensagens.
- ***upt*** é o porto UDP no qual o servidor de mensagens atende a pedidos vindos dos terminais.
- ***tpt*** é o porto TCP no qual o servidor de mensagens atende a pedidos de sessão vindos de outros servidores de mensagens.
- ***siip*** é o endereço IP do servidor de identidades fornecido pelo corpo docente. Este argumento é opcional. Por omissão, ***siip*** deve tomar o endereço IP da máquina **tejo.tecnico.ulisboa.pt**.
- ***sipt*** é o porto UDP do servidor de identidades fornecido pelo corpo docente. Este argumento é opcional. Por omissão, ***sipt*** deve tomar o valor **59000**.
- ***m*** é o número máximo de mensagens guardadas no servidor de mensagens. Este argumento é opcional. Por omissão, deve tomar o valor 200.
- ***r*** é o intervalo de tempo entre registos do servidor no servidor de identidades de mensagens, medido em segundos. Este argumento é opcional. Por omissão, deve tomar o valor 10.

Em resultado da invocação, a aplicação disponibiliza um servidor UDP no porto ***upt*** para atender aos pedidos vindos dos terminais e um servidor TCP no porto ***tpt*** para atender a pedidos de estabelecimento de sessão vindos de outros servidores de mensagens. A especificação da aplicação **msgserv** compreende uma interface de comando, o protocolo de comunicação com o servidor de identidades, o protocolo de comunicação com os terminais, já descrito anteriormente, e o protocolo de comunicação com os outros servidores de mensagens.

3.1 Interface de utilizador

A interface de utilizador aceita os seguintes comandos.

- **join**
  Registo do servidor de mensagens no servidor de identidades.
- **show_servers**

Listagem da identidade de todos os servidores de mensagens com os quais este servidor tem estabelecida uma sessão TCP.

- **`show_messages`**
  Listagem de todas as mensagens guardadas no servidor, ordenadas pelos tempos lógicos.
- **`exit`**
  Terminação da aplicação.

3.2 Protocolo de comunicação entre servidores de mensagens e o servidor de identidades

O protocolo de comunicação entre os servidores de mensagens e o servidor de identidades faz uso das mesmas mensagens protocolares da comunicação entre terminais e servidor de identidades, nomeadamente **`GET_SERVERS`** e **`SERVERS`**, adicionando a seguinte mensagem protocolar.

- **`REG`** ***`name;ip;upt;tpt`***
  Mensagem com a qual um servidor de mensagens se regista no servidor de identidades. A identidade de um servidor de mensagens é composta pelo seu nome, ***`name`***, o seu endereço IP, ***`ip`***, o porto UDP no qual ele atende a pedidos vindos dos terminais, ***`upt`***, e o porto TCP no qual ele atende a pedidos de estabelecimento de sessão vindos de outros servidores de mensagens, ***`tpt`***.

3.3 Protocolo de comunicação entre servidores de mensagens

O protocolo de comunicação entre dois servidores de mensagens compreende as seguintes mensagens protocolares.

- **`SGET_MESSAGES\n`**
  Mensagem enviada dum servidor de mensagens para outro solicitando-lhe todas as mensagens conhecidas.
- **`SMESSAGES\n`**`(`***`clock;message`***`\n)*\n`
  Mensagem enviada dum servidor de mensagens para outro com uma lista de mensagens. Cada mensagem ***`message`*** é precedida pelo seu tempo lógico, ***`clock`***.

4. Desenvolvimento

Cada grupo de alunos deve adquirir a destreza necessária sobre programação em redes para realizar o sistema de comunicação *RCI Reliable Message Board*

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

i. Considere uma versão leve da aplicação **`msgserv`** que assume a existência de apenas um servidor de mensagens. Implemente o protocolo de comunicação entre os servidores de mensagens e o servidor de identidades.

ii. Considere a aplicação **`rmb`**. Implemente o protocolo de comunicação entre os terminais e o servidor de identidades.
iii. Implemente o protocolo de comunicação entre os terminais e os servidores de mensagens.
iv. Considere a aplicação **`msgserv`** completa. Implemente o protocolo de comunicação entre servidores de mensagens.

Comente e teste o seu código **enquanto** o desenvolve. Faça uso dum depurador de erros e dum analisador de pacotes. 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()`;
- Controlo UDP: `socket()`, `bind()`, `close()`;
- Comunicação por UDP: `sendto()`, `recvfrom()`;
- Controlo TCP: `socket()`, `bind()`, `listen()`, `connect()`, `accept()`, `close()`;
- Comunicação por TCP: `write()`, `read()`;
- Multiplexagem de informação e temporizadores: `select()`
- Medição do tempo: `time()`.

Tanto os clientes como os servidores devem terminar graciosamente pelo menos nas seguintes situações de falha:

- Condições de erro nas chamadas de sistema.
- Mensagens do protocolo com formatação errada;
- 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 os códigos fonte do **`rmb`** e do **`msgserv`** 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 e ser compilado sem erros com o comando `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 sexta-feira, dia 7 de Abril, às 23:59.