RCI Service On a Ring

Redes de Computadores e Internet
2º Semestre 2017/2018
Projeto de Laboratório
Versão 2.0

1. Descrição do projeto

Pretende-se desenvolver um sistema de disponibilização de um serviço fornecido igualmente por qualquer um de um conjunto possível de servidores (a natureza do serviço é indefinida). Os servidores têm identificadores únicos, representados por números inteiros positivos, e estão interligados num anel orientado. Cada servidor sabe qual é o identificador, o endereço IP e o porto TCP do seu sucessor no anel, comunicando com ele sobre uma sessão TCP.

Há um servidor central (provido pelo corpo docente) que a cada momento tem registo de um servidor de despacho, disponível para fornecer o serviço. Um cliente que pretenda requisitar o serviço interroga o servidor central sobre qual o servidor de despacho, seu identificador, endereço IP e porto UDP. De seguida, o cliente requisita o serviço ao servidor de despacho, ficando este indisponível. Mais tarde, o cliente há de libertar o servidor, devolvendo-o ao estado disponível. A comunicação do cliente quer com o servidor central quer com o servidor de despacho ocorre por UDP.

Quando o servidor de despacho passa do estado disponível para o indisponível em virtude de atender a um cliente, ele apaga o seu registo no servidor central e inicia uma busca em torno do anel por um novo servidor de despacho, o qual atenderá o próximo cliente. Na concretização dessa busca, o servidor que passou a indisponível faz circular em torno do anel uma mensagem, dita testemunho tipo *S*. O primeiro servidor disponível a receber o testemunho tipo *S* assume-se como novo servidor de despacho registando-se no servidor central. Ele altera o tipo do testemunho de *S* para *T* e prossegue com a sua circulação em torno do anel. O servidor que lançou o testemunho tipo *S* em torno do anel receberá de volta um testemunho tipo *T*, se o serviço foi transferido, ou um testemunho tipo *S*, se nenhum servidor do anel estava disponível quando o testemunho passou por ele. Neste último caso, dizemos que o anel está indisponível. O servidor lança então um testemunho tipo *I* em torno do anel para que a informação de indisponibilidade do anel seja partilhada por todos os servidores.

Se o anel está indisponível e todos os servidores sabem disso, como é que o anel retoma o serviço quando um dos servidores fica disponível? Quando um servidor com conhecimento de que o anel está indisponível fica disponível, ele lança um testemunho

tipo *D* à volta do anel, notificando todos os servidores que o anel passa a disponível. O testemunho tipo *D* contem o identificador do servidor que o lançou no anel. Se o testemunho do tipo *D* viajar ao longo de todo o anel retornando ao servidor do qual emanou, então o servidor regista-se no servidor central na qualidade de servidor de despacho. No entanto, é possível que dois (ou mais) servidores fiquem disponíveis concorrentemente, antes que os testemunhos do tipo *D* por eles lançados alcancem o outro. Só um desses servidores será eleito servidor de despacho e concomitantemente registado no servidor central. Arbitramos que esse servidor é o que tem um identificador menor. Mas como é que os servidores disponíveis sabem qual deles tem identificador menor? Um servidor que lançou um testemunho do tipo *D* e vem a receber um testemunho tipo *D* só o propaga em torno do anel se o identificador contido no testemunho for inferior ao seu. Desta feita, apenas o servidor disponível com o identificador mais baixo receberá de volta o testemunho do tipo *D* que lançou, registando-se como servidor de despacho no servidor central.

Há ainda duas questões, próximas uma da outra, a resolver. Como é que um servidor entra no anel? Como sai do anel? Para efeitos de manutenção do anel, existe um servidor de arranque registado no servidor central. Quando um servidor novo se quer juntar ao anel, ele pede ao servidor central o identificador, endereço IP e porto TCP do servidor de arranque. Pedido e resposta ocorrem sobre UDP. De seguida, o servidor novo liga-se ao servidor de arranque, estabelecendo-o como seu sucessor. É agora necessário que o predecessor do servidor de arranque se ligue ao servidor novo e o estabeleça como seu sucessor. Para isso, o servidor de arranque lança um testemunho tipo *N* ao longo do anel, o qual contem o seu identificador e o identificador, endereço IP e porto TCP do servidor novo. O testemunho tipo *N* viajará até ao predecessor do servidor de arranque, o qual trocará de sucessor, do servidor de arranque para o servidor novo.

Quando um servidor, que não o de arranque, quer sair do anel, ele terá que primeiro assegurar a ligação entre o seu predecessor e o seu sucessor. Para esse efeito, ele faz circular um testemunho tipo *O* em torno do anel com o seu identificador e o identificador, endereço IP e porto TCP do seu sucessor. Quando o testemunho tipo *O* chega ao predecessor do servidor que está de saída, este predecessor liga-se ao sucessor identificado no testemunho e estabelece-o como seu sucessor. Ele ainda entrega o testemunho tipo *O* ao servidor de saída para que este saia com a certeza de que o anel permanece fechado. Se se trata do servidor de arranque que vai sair, então ele apaga o seu registo no servidor central, transferindo essa função para o seu sucessor, o qual se registará posteriormente. Seguidamente, o servidor procede como descrito anteriormente. Os alunos devem convencer-se que os procedimentos anteriormente descritos funcionam bem nos casos limite de apenas um ou dois servidores no anel.

O projeto compreende duas aplicações. A aplicação **`reqserv`** para requisição de serviço pelos clientes (Secção 2) e a aplicação **`service`** para disponibilização do

serviço pelo conjunto de servidores (Secção 3). Os serviços possíveis são representados por números inteiros positivos.

2. Especificação da aplicação reqserv

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

`reqserv [-i` ***`csip`*****`] [-p`** ***`cspt`*****`]`**

em que:

- ***`csip`*** é o endereço IP do servidor central provido pelo corpo docente. Este argumento é opcional. Por omissão, a aplicação deve tomar o endereço IP da máquina **`tejo.tecnico.ulisboa.pt`**.
- ***`cspt`*** é o porto UDP do servidor central provido pelo corpo docente. Este argumento é opcional. Por omissão, a aplicação deve tomar o valor **`59000`**.

A especificação da aplicação **`reqserv`** compreende uma interface de utilizador, o protocolo de comunicação com o servidor central e o protocolo de comunicação com os servidores que fornecem o serviço.

2.1 Interface de utilizador

A interface de utilizador aceita os comandos seguintes.

- **`request_service`** ***`x`*** (abreviatura **`rs`** ***`x`***)
  Solicitação de um serviço ***`x`***.
- **`terminate_service`** (abreviatura **`ts`**)
  Conclusão do serviço em curso.
- **`exit`**
  Terminação da aplicação.

2.2 Protocolo de comunicação entre clientes e o servidor central

O protocolo de comunicação entre clientes e o servidor central compreende as duas mensagens protocolares seguintes.

- **`GET_DS_SERVER`** ***`x`***
  Mensagem com a qual um cliente solicita ao servidor central a identidade do servidor de despacho do serviço ***`x`***, seu endereço IP e porto UDP.
- **`OK`** ***`id;ip;upt`***
  Mensagem de resposta do servidor central a um cliente. O campo ***`id`*** é o identificador do servidor de despacho do serviço ***`x`***. Os campos ***`ip`*** e ***`upt`*** indicam o endereço IP e o porto UDP do servidor de despacho, respetivamente. No caso de o servidor central não ter registo de um servidor de serviço, todos os campos vêm a zero.

2.3 Protocolo de comunicação entre clientes e servidores

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

- **`MY_SERVICE ON | OFF`**
  Mensagem com a qual um cliente requisita um serviço (**`ON`**) ou dá o serviço em curso por concluído (**`OFF`**).
- **`YOUR_SERVICE ON | OFF`**
  Mensagem com a qual um servidor confirma uma mensagem recebida de um cliente, espelhando a informação de serviço que este lhe enviou.

3. Especificação da aplicação service

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

`service -n` ***`id`*** **`-j`** ***`ip`*** **`-u`** ***`upt`*** **`-t`** ***`tpt`*** **`[-i`** ***`csip`*** **`] [-p`** ***`cspt`*** **`]**

em que:

- ***`id`*** é o identificador do servidor.
- ***`ip`*** é o endereço IP do servidor.
- ***`upt`*** é o porto UDP no qual o servidor atende a pedidos vindos dos clientes.
- ***`tpt`*** é o porto TCP no qual o servidor atende a pedidos de ligação vindos de outros servidores.
- ***`csip`*** é o endereço IP do servidor central provido pelo corpo docente. Este argumento é opcional. Por omissão, a aplicação deve tomar o endereço IP da máquina **`tejo.tecnico.ulisboa.pt`**.
- ***`cspt`*** é o porto UDP do servidor central provido pelo corpo docente. Este argumento é opcional. Por omissão, a aplicação deve tomar o porto **`59000`**.

Em resultado da invocação, a aplicação disponibiliza um servidor UDP no porto ***`upt`*** para atender aos pedidos vindos dos clientes e um servidor TCP no porto ***`tpt`*** para atender a pedidos de ligação vindos de outros servidores. A especificação da aplicação **`service`** compreende uma interface de comando, o protocolo de comunicação com o servidor central, o protocolo de comunicação com os clientes, já descrito anteriormente, e o protocolo de comunicação com os outros servidores.

3.1 Interface de utilizador

A interface de utilizador aceita os seguintes comandos.

- **`join`** ***`x`***
  Incorporação do servidor no anel do serviço ***`x`***. Por omissão, o servidor entra no anel disponível para o serviço.

- **`show_state`**
  Solicitação do estado do servidor, o qual é composto por: informação sobre a sua disponibilidade; informação sobre a disponibilidade do anel; identificador do seu sucessor.
- **`leave`**
  Saída do servidor do anel.
- **`exit`**
  Terminação da aplicação.

3.2 Protocolo de comunicação entre servidores e o servidor central

O protocolo de comunicação entre servidores e o servidor central faz uso das mensagens protocolares seguintes.

- **`SET_DS`** ***`x;id;ip;upt`***
  Mensagem com a qual o servidor ***`id`*** se afirma servidor de despacho para o serviço ***`x`***. O endereço IP e porto UDP do serviço são indicados nos campos ***`ip`*** e ***`upt`***, respetivamente.
- **`WITHDRAW_DS`** ***`x;id`***
  Mensagem com a qual o servidor ***`id`*** se afirma indisponível para fornecer o serviço ***`x`***.
- **`SET_START`** ***`x;id;ip;tpt`***
  Mensagem com a qual o servidor ***`id`*** se afirma servidor de arranque para o serviço ***`x`***. O endereço IP e porto TCP são indicados nos campos ***`ip`*** e ***`tpt`***, respetivamente.
- **`WITHDRAW_START`** ***`x;id`***
  Mensagem com a qual o servidor ***`id`*** se retira como servidor de arranque.
- **`GET_START`** ***`x;id`***
  Mensagem com a qual o servidor ***`id`*** questiona o servidor central sobre o servidor de arranque do serviço ***`x`***.
- **`OK`** ***`id;id2;ip;tpt`***
  Mensagem de resposta do servidor central a qualquer uma das mensagens anteriores. O campo ***`id`*** espelha o identificador do servidor que interpelou o servidor central. No caso da resposta à mensagem **`GET_START`** ***`x;id`***, o campo ***`id2`*** é o identificador do servidor de arranque e os campos ***`ip`*** e ***`tpt`*** indicam o seu endereço IP e o porto TCP, respetivamente. Se não houver servidor de arranque, então os campos ***`id2`***, ***`ip`*** e ***`tpt`*** vêm a zero. No caso o servidor central não conseguir atender ao pedido de um servidor, todos os campos vêm a zero.

3.3 Protocolo de comunicação entre servidores

O protocolo de comunicação entre servidores do anel compreende testemunhos parametrizados pelo seu tipo, uma mensagem para a adesão de um novo servidor ao anel e outra para delegar o arranque do anel no sucessor de um servidor. Relativamente

à disponibilização do serviço, os testemunhos são da forma seguinte (**\n** é o terminador de mensagens TCP).

- **TOKEN *id;type\n***
  Testemunho lançado no anel pelo servidor ***id***. O campo ***type*** pode tomar um dos valores seguintes:
  S–testemunho de procura de um servidor de despacho.
  T–testemunho de servidor de despacho encontrado.
  I–testemunho de indicação de anel indisponível.
  D–testemunho de retoma do serviço após uma situação de anel indisponível.

Relativamente à manutenção do anel as mensagens protocolares são as seguintes.

- **NEW *id;ip;tpt\n***
  Mensagem com a qual o servidor novo ***id*** se apresenta ao servidor de arranque. Os campos ***ip*** e ***tpt*** indicam o endereço IP e o porto TCP do servidor novo, respetivamente.
- **TOKEN *id;type;id2;ip;tpt\n***
  Testemunho lançado no anel pelo servidor ***id*** com vista à entrada ou saída de um servidor. O campo ***type*** pode tomar um de dois valores:
  N–testemunho lançado pelo servidor de arranque ***id*** relativo à entrada do servidor ***id2***. Os campos ***ip*** e ***tpt*** indicam o endereço IP e o porto TCP de ***id2***, respetivamente.
  O–testemunho lançado pelo servidor ***id*** relativamente à sua saída do anel. O servidor ***id2*** é o sucessor do servidor ***id***. Os campos ***ip*** e ***tpt*** indicam o endereço IP e o porto TCP de ***id2***, respetivamente.
- **NEW_START*\n***
  Mensagem com a qual o servidor de arranque delega a função de arranque no seu sucessor.

4. Desenvolvimento

Um protocolo pode e deve ser conceptualizado por uma máquina de estados. Uma máquina de estados é caracterizada por um conjunto de estados, um dos quais é o estado inicial, e um conjunto de transições de um estado para outro. As transições são causadas por acontecimentos, tais como a receção de uma mensagem ou a receção de um comando exterior, e geram acontecimentos, tais como o envio de mensagens ou a resposta a um comando exterior. Aconselha-se os alunos a desenhar a máquina de estados das aplicações do projeto antes de começar a programar.

A programação das aplicações do projeto será baseada 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()`.

O código deve ser comentado e testado à medida que é desenvolvido. O projeto 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.

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

- Receção de mensagens mal formatadas.
- Terminação imprevista de sessões TCP.
- Erros na invocação de chamadas de sistema.

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.
- Manual on-line, comando `man`.

6. Entrega do Projecto

O código a entregar deve ser guardado num arquivo `zip` contendo o código fonte das aplicações **`reqserv`** e **`service`** e a respetiva `makefile`. A entrega do projeto é feita por e-mail ao seu docente de laboratório. O arquivo deve estar preparado para ser aberto para o diretório corrente e 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 6/04/2018.