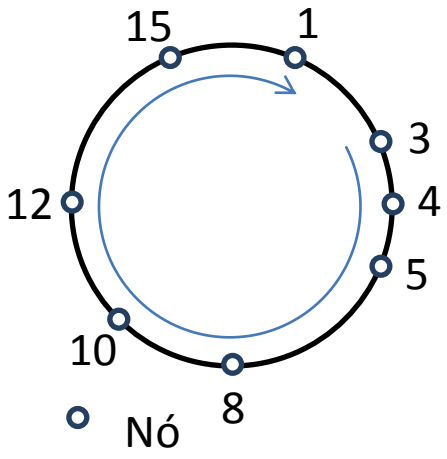
Base-de-dados distribuída

Redes de Computadores
2º Semestre 2014/2015
Projeto de Laboratório

1. Descrição da aplicação

Pretende-se distribuir um conjunto de objetos por um conjunto de nós. Cada objeto é indexado por um identificador. Os identificadores são números entre 0 e 2^h-1, para um valor acordado de *h*, entendidos em anel, isto é, ao identificador 2^h-1 segue-se o identificador 0. Cada nó tem um identificador único. O nó *i* conhece o identificador *succi* do nó que o sucede no anel e o identificador *predi* do nó que o precede no anel. Ele tem uma sessão TCP estabelecida com *succi* e outra com *predi*. Em resultado destas sessões TCP, os nós formam um anel de sessões TCP. O nó *i* é responsável por todos os identificadores compreendidos entre *predi* (exclusivamente) e o dele próprio (inclusivamente). Qualquer nó *i* pode iniciar a pesquisa de um identificador *k* ao longo do anel. Essa pesquisa propaga-se ao longo do anel até chegar ao nó *j* responsável por *k*. O nó *j* responde com o trio (*j*, *j*.IP, *j*.TCP), em que *j*.IP é o endereço IP do nó e *j*.TCP é o porto do seu servidor TCP, que é propagado de volta, em sentido inverso, ao longo do anel até chegar ao nó *i*. Ao par (*j*.IP, *j*.TCP) chamamos a localização de *j*.



Por exemplo, na figura, *h* = 4, pelo que temos 16 identificadores em anel. O sucessor do nó 12 é o nó 15 e predecessor do nó 12 é o nó 10. O nó 15 é responsável pelos identificadores 13, 14, 15; o nó 1 pelos identificadores 0 e 1. Uma pesquisa pelo identificador 14 iniciada no nó 4 irá viajar ao longo do anel, no sentido dos ponteiros do relógio, até chegar ao nó 15 que responderá com o seu endereço IP e o seu porto TCP. A resposta viaja em sentido contrário ao dos ponteiros do relógio até chegar ao nó 4.

Para além das possibilidades de pesquisa, há que fazer a gestão do anel, isto é, garantir que os nós mantêm um anel de sessões TCP quando um nó sai ou um novo nó adere ao anel. Quando um nó *i* sai ele avisa o seu predecessor *predi* para se ligar ao seu sucessor *succi*, fechando o anel sem *i*. Um nó que queira aderir ao anel tem que, antes de tudo, saber o identificador e a localização de pelo menos um nó do anel, dito nó de arranque. Existe um servidor UDP de arranque, fornecido pelo corpo docente, onde se atualiza o nó de arranque. Um nó que queira aderir ao anel consulta o servidor de arranque e obtém o identificador *n* do nó de arranque bem como a sua localização. O nó escolhe um identificador $i \neq n$ para ele próprio e liga-se ao nó *n*. O nó pede ao nó *n* para iniciar uma pesquisa do identificador *i*. Do resultado dessa pesquisa o nó *i* aprende se o identificador escolhido já está em uso ou não. Se estiver, o nó terá que tentar outro identificador. Se o identificador *i* não estiver em uso, o nó aprende o identificador *j* e a localização (*j*.IP, *j*.TCP) daquele nó no anel que virá a ser o seu sucessor. O nó *i* estabelece uma sessão TCP com o nó *j* e o nó *j* avisa o seu predecessor *predj* para se ligar ao nó *i*, fechando o anel já com o nó *i*. Na concretização do projeto, assumimos $h = 6$, ou seja, 64 identificadores em cada anel.

2. Especificação

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

- Uma interface de utilizador
- O protocolo de pesquisa de um identificador
- Os protocolos de gestão do anel

2.1 Invocação da aplicação

A aplicação **`ddt`** é invocada com o comando

`ddt [-t ringport] [-i bootIP] [-p bootport]`

em que

- ***`bootIP`*** e ***`bootport`*** são o endereço IP e o porto UDP do servidor de arranque. Por omissão, ***`bootIP`*** é o endereço IP da máquina **`tejo.ist.utl.pt`** e ***`bootport`*** toma o valor **`58000`**.
- ***`ringport`*** é o porto do servidor TCP usado para estabelecer as sessões TCP no anel.

Em resultado da invocação, a aplicação disponibiliza uma interface de utilizador e um servidor TCP no porto ***`ringport`***.

2.2 Interface de utilizador

A interface de utilizador prevê os comandos seguintes.

- **`join` *`x i`***
  O utilizador pretende que o nó se junte ao anel *x* tomando *i* como identificador nesse anel. O nó só pode pertencer a um anel de cada vez.
- **`join` *`x i succi succi.IP succi.TCP`***
  O utilizador pretende que o nó se junte ao anel ***x*** tomando ***i*** como identificador nesse anel e *succi* como o identificador do seu sucessor, localizado em (***succi.IP succi.TCP***). Este comando pressupõe que o utilizador sabe que o identificador *i* é único no anel *x* e que *succi* é o sucessor de *i*. Ele serve para construir um anel sem efetuar pesquisas de identificador.
- **`leave`**
  O utilizador pretende que o nó abandone o anel a que pertence.
- **`show`**
  Mostra ao utilizador o número do anel, o identificador do nó nesse anel, bem como os identificadores do seu sucessor e do seu predecessor.
- **`search` *`k`***
  O utilizador pretende saber o identificador e a localização do nó responsável pelo identificador *k*.
- **`exit`**
  O utilizador fecha a aplicação.

2.3 Pesquisa de um identificador

Na pesquisa de um identificador, consideramos três acontecimentos possíveis.

- O utilizador do nó *i* pede para pesquisar o identificador *k*. Se *i* é responsável por *k*, então ele devolve o trio (*i*, *i*.IP, *i*.TCP) ao utilizador. Se *i* não é responsável por *k*, então ele envia um mensagem <**`QRY` *`i k`*** > a *succi*.
- O nó *i* recebe uma mensagem <**`QRY` *`j k`*** >. Se *i* é responsável por *k*, então ele envia a mensagem <**`RSP` *`j k i i.IP i.TCP`***> a *predi*. Se *i* não é responsável por *k*, então ele envia <**`QRY` *`j k`*** > a *succi*.
- O nó *i* recebe a mensagem <**`RSP` *`j k l l.IP l.TCP`***>. O caso *j* = *i* significa que o nó *j* que iniciou a pesquisa recebe a resposta de volta. Então, ele devolve o trio (*l*, *l*.IP, *l*.TCP) ao utilizador. Se *j* ≠ i, então o nó *i* envia <**`RSP` *`j k l l.IP l.TCP`***> a *predi*.

2.4 Adesão de um nó

Cada anel é conhecido por um número (sem correlação com os identificadores de cada anel). Os docentes fornecem um servidor de arranque UDP com associações entre número do anel e o identificador e a localização do nó de arranque desse anel. Quando

um nó quer aderir ao anel *x* ele começa por escolher um identificador potencial *i* nesse anel.

- O nó *i* inquire o servidor de arranque com a mensagem <**BQRY *x***>. Se o anel *x* ainda não tiver nenhum nó, então o servidor de arranque responde com <**EMPTY**>. Subsequentemente, o nó *i* assume-se como nó de arranque enviando <**REG *x i i*.IP *i*.TCP**> ao servidor de arranque, respondendo este com <**OK**>. Se o anel *x* já tiver pelo menos um nó, então o servidor de arranque responde com <**BRSP *x j j*.IP *j*.TCP**> em que *j* é o nó de arranque do anel *x*. Se *j* = *i*, então o utilizador tem que escolher outro identificador para o nó. Caso contrário, ele avança para o passo seguinte.
- O nó *i* estabelece uma sessão TCP com o nó de arranque *j* e envia-lhe uma mensagem <**ID *i***>. O nó de arranque faz uma pesquisa <**QRY *j i***> pelo anel no fim da qual recebe <**RSP *j i l l*.IP *l*.TCP**>, enviando de seguida a mensagem <**SUCC *l l*.IP *l*.TCP**> ao nó *i*. O nó *i* termina a sessão TCP com *j*. Se *l* = *i*, então o utilizador tem que escolher outro identificador para o nó. Caso contrário, ele avança para o passo seguinte.
- O nó *i* estabelece uma sessão TCP como o nó *l* e envia-lhe <**NEW *i i*.IP *i*.TCP**>. Isto significa que *l* será o sucessor de *i* e *i* será o predecessor de *l*. O nó *l* envia uma mensagem <**CON *i i*.IP *i*.TCP**> ao seu antigo predecessor *m* fechando de seguida a sessão TCP que mantinha com ele. O nó *m* restabelece o anel enviando a mensagem <**NEW *m m*.IP *m*.TCP**> ao nó *i*.

2.5 Saída de um nó

A saída do nó *i* do anel *x* envolve os passos seguintes.

- Se *i* for o único nó de *x* ele envia <**UNR *x***> ao servidor de arranque, anulando o anel *x*. O servidor de arranque responde com <**OK**>. Se *i* não for o único nó de *x* mas for o nó de arranque de *x*, então envia <**REG *x succi succi*.IP *succi*.TCP**> ao servidor de arranque que lhe responde com <**OK**>. Envia ainda <**BOOT**> a *succi* notificando-o de que ele passa a ser o nó de arranque.
- O nó *i* fecha a sessão TCP com *succi* e envia <**CON *succi succi.IP succi*.TCP**> a *predi*. Subsequentemente, o nó *predi* reestabelece o anel enviando < **NEW *predi predi.IP predi*.TCP**> a *succi*.

3. Desenvolvimento

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

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

i. Realize o cliente que interage com o servidor de arranque alojado na máquina **`tejo.tecnico.ulisboa.pt`**. Comandos **`join x i`** (único nó), **`show`**, **`leave`** (único nó) e **`exit`**.
ii. Realize o protocolo de inserção de um nó no anel sabendo o seu identificador, bem como o identificador do seu sucessor no anel e a localização deste. Comando **`join x i succi succi.IP succi.TCP`**.
iii. Realize a pesquisa de um identificador no anel. Comado **`search k`**.
iv. Realize o protocolo de adesão de um nó ao anel. Comando **`join x i`**.
v. Realize o protocolo de abandono de um nó do anel. Comando **`leave`**.

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 Projecto

O código a entregar deve ser guardado num arquivo `zip` contendo o código fonte da **`ddt`** 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 directório 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 29/03/2015.