

Transferência de Conteúdos em Redes Arbóreas

Redes de Computadores e Internet

2º Semestre 2022/2023

Projeto de Laboratório

1. Introdução

Considera-se uma rede composta por nós com *identificadores únicos* interligados em árvore. Cada nó contém um conjunto de conteúdos com *nomes únicos a cada nó* (conteúdos de nós distintos podem ter o mesmo nome). Um utilizador de um nó origem que pretende ir buscar determinado conteúdo forma uma *mensagem de pesquisa* contendo o identificador do nó destino, o identificador do nó origem, e o nome do conteúdo. Esta mensagem será encaminhada na rede e chegará ao destino que responderá com uma *mensagem de conteúdo*, contendo o identificador do nó que pesquisou o conteúdo, destino da mensagem, o seu identificador, origem da mensagem e o conteúdo propriamente dito (para simplificar basta uma indicação que o conteúdo existe). Se o destino não tiver o conteúdo pesquisado, então ele responde com uma *mensagem de indisponibilidade*.

Cada nó mantém uma *tabela de expedição* com entradas, cada uma das quais associa um nó destino ao vizinho ao longo do único caminho na árvore até lá. Esta tabela é povoada pelas próprias mensagens que atravessam a rede e pode estar incompleta a determinado instante. Um nó que tenha uma mensagem com destino em outro nó consulta a sua tabela de expedição. Três situações podem ocorrer: (1) O nó encontra uma entrada para o nó destino na tabela e o vizinho associado ao nó destino é diferente daquele por onde recebeu a mensagem, ou a mensagem foi originada no próprio nó. Neste caso, o nó *expede* a mensagem para o vizinho lido da tabela de expedição; (2) O nó encontra uma entrada para o nó destino na tabela, mas o vizinho associado a essa entrada é o próprio vizinho por onde a mensagem foi recebida. Então, o nó *filtra* a mensagem, isto é, descarta-a; (3) O nó não encontra uma entrada para o nó destino na tabela. Desta feita, o nó *difunde* a mensagem por todos os seus vizinhos, à exceção daquele por onde recebeu a mensagem, isto é, envia cópias da mensagem a cada um desses vizinhos.

Por outro lado, de cada vez que um nó recebe uma mensagem através de um vizinho, ele introduz uma entrada na sua tabela de expedição com a associação entre o nó origem da mensagem e o vizinho por onde esta foi recebida, para ser usada em sentido inverso, aquando da chegada de uma outra mensagem com destino no nó origem da mensagem original. A este procedimento chama-se *aprendizagem transparente* (ou *aprendizagem automática*!).

1.1 Topologia da rede em árvore

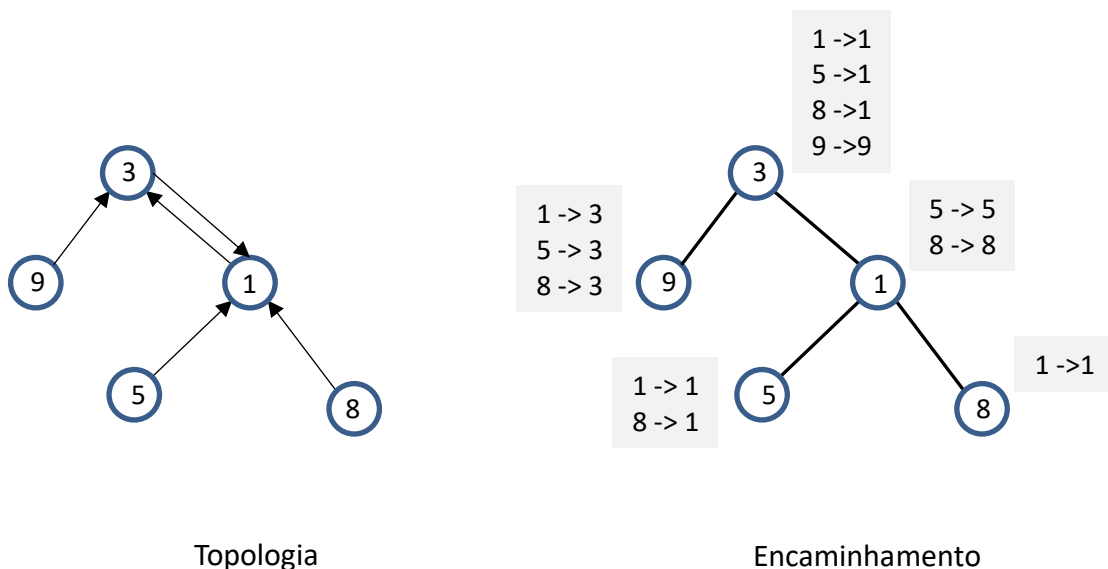
A topologia da rede que interliga os nós será sempre uma árvore. Cada aresta da árvore é substanciada numa sessão TCP, sendo os dois nós que partilham a aresta *vizinhos* um do outro. A manutenção de uma árvore requer procedimentos para a entrada e para a saída de um nó. Para simplificar, supomos que não se dão entradas e/ou saídas simultâneas.

Orientação da árvore.

Um mecanismo que permite a manutenção de uma árvore começa por considerar uma orientação para as suas arestas. Se a orientação da aresta entre o nó X e o nó Y é no sentido de X para Y, então dizemos que o nó Y é *vizinho externo* do nó X e este é *vizinho interno* do nó Y. As orientações das arestas da árvore satisfazem as duas condições seguintes:

1. Cada nó tem apenas um vizinho externo, podendo ter múltiplos vizinhos internos ou nenhum.
2. Em redes com mais do que um nó, há exatamente dois nós, chamados *âncora*, que são vizinhos externos um do outro.

A figura (lado esquerdo) mostra uma árvore com as arestas orientadas.



O *contacto* de um nó é o par com o seu endereço IP e porto TCP servidor (de escuta). Cada nó mantém os identificadores e contactos dos seus vizinhos internos, do seu vizinho externo e de um nó de recuperação (*backup*), sendo este nó o vizinho externo do vizinho externo. Em particular, os âncora têm-se a si próprios como nós de recuperação.

Entrada de um nó.

Há um servidor de nós (fornecido pelo corpo docente) com um diretório onde estão registados todos os nós pertencentes à rede, na forma do seu identificador e contacto. Um nó entrante na rede consulta o servidor de nós que lhe fornece a lista dos nós da rede. O nó entrante escolhe um desses nós, liga-se a ele numa sessão TCP, estabelece-o como seu

vizinho externo, envia-lhe uma *mensagem de entrada* com o seu identificador e contacto e recebe dele uma *mensagem de vizinho externo* com o identificador e contacto deste, o qual virá a ser o vizinho de recuperação do nó entrante. Se o vizinho externo do nó entrante era o único na rede, então, antes de responder com a mensagem de vizinho externo, ele promove o nó entrante a vizinho externo, formando os dois nós as âncoras da rede. Por último, o nó entrante regista-se no servidor de nós. A comunicação entre um nó e o servidor de nós ocorre por UDP.

Saída de um nó.

Um nó que queira sair da rede apaga o seu contacto no servidor de nós e termina as sessões TCP que tinha com todos os seus vizinhos. Os vizinhos internos do nó de saída ficam sem vizinho externo e a rede fica separada. Um nó que tenha perdido o seu vizinho externo e cujo vizinho de recuperação não seja ele próprio, liga-se ao seu vizinho de recuperação, envia-lhe uma mensagem de entrada e obtém deste uma mensagem de vizinho externo com a qual obtém novo nó de recuperação. Contrariamente, se o vizinho de recuperação do nó é ele próprio, então o nó promove qualquer um dos seus vizinhos internos a vizinho externo antes de enviar uma mensagem de vizinho externo.

1.2 Pesquisa e encaminhamento de conteúdos

Como já foi referido, o utilizador associado a cada nó cria um conjunto de conteúdos com nomes únicos a esse nó. A transação de um conteúdo envolve três tipos de mensagens, de pesquisa, de conteúdo e de indisponibilidade, aqui referidas apenas por mensagens. Todas estas mensagens têm os identificadores do nós origem e destino da mensagem. O nó destino constante das mensagens é usado para as expedir e filtrar, enquanto o nó origem constante das mensagens é usado para preenchimento das tabelas de expedição.

Importa ainda esclarecer como se atualizam as tabelas de expedição quando um nó entra ou sai da rede. A entrada de um nó na rede não requer quaisquer ações adicionais. Quando este nó enviar uma mensagem, ela será difundida na rede e as tabelas de expedição passarão a contar com uma entrada para esse nó.

Por outro lado, quando um nó sai da rede, as tabelas de expedição têm de ser explicitamente atualizadas. Seja X o nó que sai da rede. Há dois aspetos a considerar. Primeiro, qualquer vizinho de X tem de apagar da sua tabela de expedição as entradas que têm X por vizinho. Segundo, todos os antigos vizinhos de X ficam encarregados de apagar as entradas em que X é nó destino das suas tabelas de expedição, se as houver, e dos outros nós nas sub-árvores definidas depois da retirada de X. Para este efeito, um vizinho de X apaga a entrada com destino X da sua tabela de expedição, se houver, e envia aos seus vizinhos (anteriores ao restauro da árvore) uma *mensagem de retirada*. Um nó que receba uma destas mensagens apaga a entrada com destino X da sua tabela de expedição, se houver, e envia a mensagem por todos os seus vizinhos à exceção daquele por onde recebeu a mensagem.

2. Especificação

Cada grupo de dois alunos deve concretizar a aplicação **cot** correspondente a um nó e composta pelos elementos seguintes:

- Comando de invocação da aplicação;
- Interface de utilizador;
- Protocolo de diretório;
- Protocolo de topologia;
- Protocolo de encaminhamento;
- Protocolo de pesquisa.

Nota: nos comandos e mensagens que se apresentarão de seguida, o espaço em branco e/ou o carácter **<LF>** (*line feed*) são separadores entre campos de mensagens, sendo que o carácter **<LF>** é também um separador entre mensagens enviadas sobre uma mesma sessão TCP. Por consequência, os nomes escolhidos pelo utilizador não devem conter espaços em branco ou o carácter **<LF>**: deverão ser sequências de caracteres alfanuméricos.

2.1 Comando de invocação da aplicação

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

cot IP TCP regIP regUDP

em que **IP** e **TCP** são o contacto do nó criado pela aplicação: **IP** é o endereço IP da máquina que aloja a aplicação e **TCP** é o porto TCP servidor da aplicação. Os parâmetros **regIP** e **regUDP** são o contacto, endereço IP e porto UDP, respetivamente, do servidor de nós fornecido pelo corpo docente que, por omissão, tomam os valores **193.136.138.142** e **59000**.

2.2 Interface de utilizador

A interface de utilizador consiste nos comandos seguintes.

- **join net id**
Entrada de um nó na rede **net** com identificador **id**. Se o identificador **id** já estiver a ser usado por outro nó, então a aplicação escolhe um identificador único, avisando o utilizador dessa alteração. Os valores de **net** são representados por três dígitos, podendo variar entre **000** e **999**; os valores de **id** são representados por dois dígitos, podendo variar entre **00** e **99**.
- **djoin net id bootid bootIP bootTCP**
Entrada de um nó na rede **net** com identificador **id**, que se sabe ser único na rede. É passado à aplicação o identificador e o contacto de um nó da rede, através dos

parâmetros **bootid**, **bootIP** e **bootTCP**, ao qual o nó se deverá ligar sem interrogar o servidor de nós. Se for **bootid** igual **id** a então cria-se a rede só com um nó.

- **create name**

É criado um conteúdo de nome **name**. Os valores de **name** são representados por sequências alfanuméricas com um máximo de 100 carateres.

- **delete name**

É apagado o conteúdo de nome **name**.

- **get dest name**

Pesquisa do conteúdo com o nome **name** localizado no nó **dest**.

- **show topology (st)**

Mostra os identificadores e os contactos dos vizinhos internos, do vizinho externo e do vizinho de recuperação.

- **show names (sn)**

Mostra os nomes dos conteúdos presentes no nó.

- **show routing (sr)**

Mostra a tabela de expedição do nó.

- **leave**

Saída do nó da rede.

- **exit**

Fecho da aplicação.

2.3 Protocolo de registo

Na comunicação com o servidor de nós são usadas as mensagens seguintes sobre UDP.

- **NODES net**

Um nó pede ao servidor de nós informação sobre os nós existentes na rede **net**, consistindo nos seus identificadores e contactos.

- **NODESLIST net<LF>**

id1 IP1 TCP1<LF>

id2 IP2 TCP2<LF>

...

O servidor de nós envia uma lista com uma linha por nó na rede **net**. Cada linha dessa lista contém o identificador e contacto de um nó.

- **REG net id IP TCP**

Um nó regista-se na rede **net**.

- **OKREG**

O servidor de nós confirma o registo de um nó.

- **UNREG net id**

O nó retira o seu registo da rede **net**.

- **OKUNREG**

O servidor de nós confirma a retirada do registo de um nó.

2.4 Protocolo de topologia

As mensagens de entrada e de vizinho externo têm, respetivamente, os formatos seguintes.

- **NEW *id IP TCP*<LF>**
Um nó dá a conhecer a outro o seu identificador e contacto.
- **EXTERN *id IP TCP*<LF>**
Um nó dá a conhecer a outro o identificador e o contacto do seu vizinho externo.

2.5 Protocolo de encaminhamento

A mensagem de retirada tem o formato seguinte.

- **WITHDRAW *id*<LF>**
Retirada da tabela de expedição do nó com identificador *id*.

2.6 Protocolo de pesquisa de objeto

As mensagens de pesquisa, de conteúdo e de indisponibilidade têm, respetivamente, o formato seguinte.

- **QUERY *dest orig name*<LF>**
Pesquisa com origem no nó *orig* do conteúdo com nome *name* localizado no nó *dest*.
- **CONTENT *dest orig name*<LF>**
Envio do nó origem *orig* ao nó destino *dest* do conteúdo com nome *name*.
- **NOCONTENT *dest orig name*<LF>**
Envio do nó origem *orig* ao nó destino *dest* da informação que o conteúdo com nome *name* não se encontra neste último nó.

3. Desenvolvimento

Cada grupo de alunos deve adquirir a destreza necessária sobre programação em redes para realizar a aplicação proposta. Sugerem-se os passos seguintes para a concretização da aplicação:

- i. Criação de uma rede com um só nó, comunicação com o servidor de nós, comandos *join* e *leave*;
- ii. Criação de uma rede com vários nós, comandos *djoin*, *join* e *show topology*;
- iii. Manutenção de uma rede com vários nós, comandos *join*, *leave* e *show topology*;
- iv. Criação de conteúdos e sua pesquisa, comandos *create*, *get* e *show routing*;
- v. Manutenção das tabelas de expedição, comandos *leave*, *show routing* e *get*.

São dadas seis semanas para concretizar o projeto, contadas a partir do início académico das aulas. Como referência, as partes i e ii devem estar concluídas em três semanas, no máximo, havendo uma verificação desta conclusão por parte do corpo docente.

O código da aplicação deverá ser comentado e testado à medida que é desenvolvido. Ele fará uso de chamadas de sistema de programação em redes que em C serão as seguintes:

- Leitura de informação do utilizador para a aplicação: `fgets()`;
- Decomposição de *strings* em tipos de dados e vice-versa: `sscanf()`, `sprintf()`;
- Gestão de um cliente UDP: `socket()`, `close()`;
- Gestão de um servidor UDP: `socket()`, `bind()`, `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 não devem terminar abruptamente nas seguintes situações de falha:

- Mensagens de protocolo inesperadas ou mal formatadas;
- Sessão TCP fechada abruptamente;
- Condições de erro nas chamadas de sistema.

4. Bibliografia

- José Sanguino, A Quick Guide to Networking Software, 5ª edição, 2020.
- W. Richard Stevens, Unix Network Programming: Networking APIs: Sockets and XTI (Volume 1), 2ª edição, Prentice-Hall PTR, 1998, capítulo 5.
- Manual on-line, comando `man`.

5. Entrega do Projecto

O código fonte da aplicação **cot** deve ser guardado num arquivo **zip** juntamente com a respetiva **makefile**. O arquivo deve estar preparado para ser aberto no diretório corrente e compilado com o comando **make**. O arquivo submetido deve ter o seguinte formato: **proj<número_do_grupo>.zip** (ex: **proj07.zip**). O arquivo deverá ser submetido no sistema fénix até sexta-feira, dia 31 de março às 23h59.