Rede Baseada em Nomes

Redes de Computadores e Internet
2º Semestre 2020/2021
Projeto de Laboratório

1. Introdução

Uma *rede baseada em nomes* é uma rede na qual cada nó detém um conjunto de objetos nomeados e sobre ela vai buscar objetos nomeados detidos por outros nós. O nome de um objeto é composto pelo identificador único do nó na rede que o detém seguido de um sub-nome único localmente atribuído por esse nó. Cada nó tem uma cache com capacidade para armazenar um pequeno conjunto de objetos.

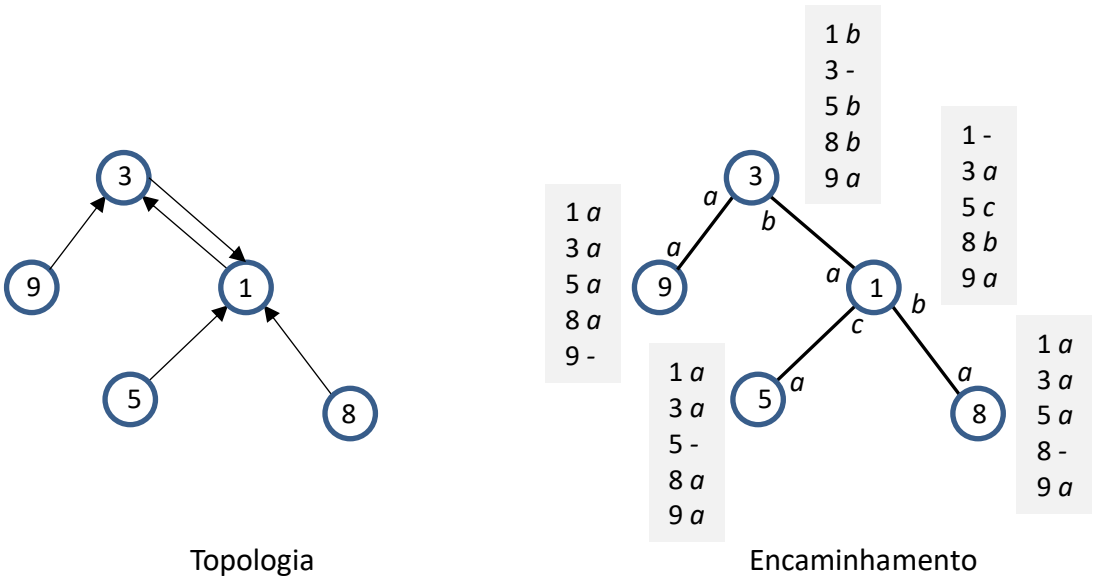
Quando um utilizador de um nó origem pretende ir buscar determinado objeto, o nó envia para a rede uma mensagem de *interesse* contendo o nome desse objeto. A mensagem de interesse será encaminhada pela rede em direção ao nó destino que detém o objeto. O nó destino ou qualquer nó intermédio que tenha o objeto armazenado em cache responde à mensagem de interesse com uma mensagem de *objeto* contendo o nome e o objeto procurado. O nó origem e os nós intermédios que transitam a mensagem de objeto no seu percurso em direção ao nó origem armazenam esta mensagem na sua cache.

1.1 Topologia da rede em árvore

Para simplificar o encaminhamento, a topologia da rede 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 da árvore requer um procedimento para a entrada de um nó e um procedimento para a saída de um nó. Estes procedimentos pressupõem uma orientação para as arestas da árvore. 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.
2. Em redes com mais do que um nó, há exatamente dois nós que são vizinhos externos um do outro.

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



Cada nó mantém o contacto do seu vizinho externo e o contacto de um nó de recuperação (*backup*), sendo este nó o vizinho externo do vizinho externo. Um contacto é um par com o endereço IP e o porto TCP servidor de um nó.

Entrada de um nó.

Há um servidor de nós (fornecido pelo corpo docente) onde estão registados os contactos de cada nó pertencente à rede. Um nó entrante na rede consulta o servidor de nós que lhe fornece uma lista de contactos. O nó entrante escolhe um dos contactos listados, liga-se ao nó correspondente numa sessão TCP, estabelece este nó como seu vizinho externo, envia-lhe uma mensagem de *presença* com o contacto do nó entrante e recebe dele uma mensagem de *vizinho* com o contacto do seu vizinho externo que virá a ser o vizinho de recuperação do nó entrante. Se o nó ao qual o nó entrante se liga era o único nó na rede, então este nó também estabelece o nó entrante como seu vizinho externo. 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. Em geral, a rede ficará desconexa. 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 o seu contacto e obtém deste um contacto para um novo vizinho de recuperação. Se o vizinho de recuperação do nó é ele próprio, então o nó promove qualquer um dos seus vizinhos internos a vizinho externo.

1.2 Encaminhamento

O encaminhamento é responsável por direcionar mensagens de interesse em direção aos destinos que detêm os objetos procurados. Como a topologia da rede é uma árvore, existe apenas um caminho de um nó para outro. Cada nó deverá manter uma tabela de expedição com entradas que associam cada nó destino à sessão TCP com o vizinho ao longo do caminho entre o nó e o nó destino. A figura (lado direito) mostra as tabelas de expedição dos vários nós, com as sessões TCP identificadas por letras.

Para manter as tabelas de expedição atualizadas face a alterações na topologia da rede, estabelecimento e retirada de arestas, são trocadas mensagens de *anúncio* e mensagens de *retirada* cujo conteúdo é apenas o identificador de um nó.

Estabelecimento de uma aresta.

Quando um nó entra na rede ou quando recupera da saída de outro nó, são criadas novas arestas, substanciadas em sessões TCP. Sempre que um nó adquire um novo vizinho, ele envia-lhe uma mensagem de anúncio por cada nó listado na sua tabela de expedição. Quando um nó recebe uma mensagem de anúncio vinda de um vizinho, ele atualiza a entrada na tabela de expedição que diz respeito ao nó identificado no anúncio, associando-o à sessão TCP que o liga ao vizinho. As mensagens de anúncio são difundidas na rede. Desta feita, se, e só se, o nó não tinha conhecimento prévio do nó anunciado, então ele envia uma réplica da mensagem de anúncio a cada um dos seus outros vizinhos.

Retirada de uma aresta.

Quando um nó sai da rede, as sessões TCP que ele mantinha com os seus vizinhos são terminadas, correspondendo à retirada das arestas que o uniam aos vizinhos. Quando um nó deteta a terminação de uma sessão TCP, ele retira da sua tabela de expedição todas as entradas que estam associadas à sessão TCP terminada e envia aos seus vizinhos uma mensagem de retirada por cada um dos nós identificados nessas entradas. Tal como no caso de uma mensagem de anúncio, uma mensagem de retirada é difundida na rede. Quando um nó recebe uma mensagem de retirada vinda de um vizinho com o identificador de um nó listado na sua tabela de expedição, ele retira a entrada correspondente da tabela e envia uma réplica da mensagem de retirada a cada um dos seus outros vizinhos.

1.3 Pesquisa de nomes e gestão da cache

Um nó que pretenda ir buscar um objeto, extrai do seu nome o identificador do nó destino que detém o objeto, consulta esse identificador na sua tabela de expedição, envia uma mensagem de interesse sobre a sessão TCP lida da tabela e fica à espera de receber informação sobre o objeto. As mensagens de interesse contêm o nome do objeto.

Um nó que receba uma mensagem de interesse sobre uma sessão TCP tida com um vizinho responde da forma seguinte.

1. Se o nó tem conhecimento do objeto, então ele envia uma mensagem de objeto de volta ao vizinho. Para simplificar, as mensagens de objeto contêm apenas os seus nomes e não os objetos propriamente ditos.
2. Se o nó não tem conhecimento do objeto e não é o destino do objeto, então ele retransmite a mensagem de objeto sobre a sessão TCP com o vizinho ao longo do caminho até ao destino e fica à espera de receber informação sobre o objeto.
3. Se o nó não tem conhecimento sobre o objeto, mas é o seu destino, então ele envia uma mensagem de *objeto ausente* de volta ao vizinho.

Um nó que esteja à espera de informação sobre um objeto e recebe a correspondente mensagem de objeto, guarda-a na sua cache. Se se tratar de um nó intermédio no caminho entre origem e destino, então ele retransmite a mensagem de objeto sobre a sessão TCP que o une ao vizinho de onde recebeu a mensagem de interesse inicial e dá a espera pelo objeto por concluída. O procedimento é semelhante para a receção de uma mensagem de objeto ausente exceto que estas mensagens não são guardadas em cache.

Uma política comum de gestão de cache é chamada menos-usado-recentemente (*Least Recently Used*, LRU), na qual uma mensagem de objeto nova expulsa de uma cache cheia a mensagem de objeto nela introduzida há mais tempo.

2. Especificação

Cada grupo de dois alunos deve concretizar a aplicação **`ndn`** 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 caráter ***<LF>*** (*line feed*) são separadores entre campos de mensagens, sendo que o caráter ***<LF>*** é também um separador entre mensagens enviadas sobre uma mesma sessão TCP. Por consequência, os nomes e sub-nomes escolhidos pelo utilizador não devem conter espaços em branco ou o caráter ***<LF>***: deverão ser sequências de carateres alfanuméricos.

2.1 Comando de invocação da aplicação

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

ndn *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***.
- **join *net id bootIP bootTCP***
  Entrada de um nó na rede ***net*** com identificador ***id***. É passado o contacto de um nó da rede, através dos parâmetros ***bootIP*** e ***bootTCP***, ao qual o nó se deverá ligar sem interrogar o servidor de nós.
- **create *subname***
  É criado um objeto cujo nome será da forma ***id.subname***, em que ***id*** é o identificador do nó.
- **get *name***
  Inicia-se a pesquisa do objeto com o nome ***name***. Este nome será da forma ***id.subname***, em que ***id*** é o identificador de um nó e ***subname*** é o sub-nome do objeto atribuído pelo nó com identificador ***id***.
- **show topology (st)**
  Mostra os contactos do vizinho externo e do vizinho de recuperação.
- **show routing (sr)**
  Mostra a tabela de expedição do nó.
- **show cache (sc)**
  Mostra os nomes dos objetos guardados na cache.
- **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.

- **REG *net IP TCP***
  Um nó regista o seu contacto no servidor de nós e associa-o à rede ***net***.
- **OKREG**
  O servidor de nós confirma o registo de um contacto.

- **UNREG** ***net IP TCP***
  O nó retira o seu contacto associado à rede ***net***.
- **OKUNREG**
  O servidor de nós confirma a retirada de um contacto.
- **NODES** ***net***
  Um nó pede ao servidor de nós os contactos associados à rede ***net***.
- **NODESLIST** ***net<LF>***
  IP1 TCP1<LF>
  IP2 TCP2<LF>
  …
  O servidor de nós envia uma lista com os contactos associados à rede ***net***.

2.4 Protocolo de topologia

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

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

2.5 Protocolo de encaminhamento

As mensagens de anúncio e retirada têm, respetivamente, os formatos seguintes.

- **ADVERTISE** ***id<LF>***
  Anúncio do nó com identificador ***id***.
- **WITHDRAW** ***id<LF>***
  Retirada do nó com identificador ***id***.

2.6 Protocolo de pesquisa de objeto

As mensagens de interesse, objeto e objeto-ausente têm, respetivamente, o formato seguinte.

- **INTEREST** ***name<LF>***
  Pesquisa do objeto com nome ***name***.
- **DATA** ***name<LF>***
  Entrega do objeto com nome ***name***.
- **NODATA** ***name<LF>***
  Informação de que o objeto com nome ***name*** não existe.

A cache de cada nó tem capacidade para um máximo de dois objetos.

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. Comunicação com o servidor de nós, comados ***join*** e ***leave*** (parcial);
ii. Criação e manutenção de uma rede, comandos ***join***, ***leave*** (parcial) e ***show topology***;
iii. Criação e manutenção de tabelas de expedição, comandos ***join***, ***leave*** e ***show routing***;
iv. Criação de objetos e pesquisa de objetos sem armazenamento em cache, comandos ***create*** e ***get***;
v. Pesquisa de objetos com armazenamento em cache, comandos ***get*** e ***show cache***;

Como referência, as partes i e ii devem estar concluídas em três semanas, no máximo.

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 **`ndn`** 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 30 de abril às 23h59.