

# **Parte IV**

## **Redes de pacotes**



Nos vários capítulos que antecedem esta parte referimo-nos várias vezes a uma infra-estrutura, que designámos como a “rede propriamente dita”, que tem por função receber pacotes de dados dos emissores, encaminhá-los até ao seu destino e finalmente entregá-los aos destinatários finais. Tradicionalmente, esta parte de uma rede de computadores é designado por **nível rede** nos modelos de camadas, ver o Capítulo 4.

Esta infra-estrutura é formada por um conjunto de nós de comutação de pacotes (comutadores de pacotes), ver o Capítulo 3, interligados por canais de dados, ver o Capítulo 2. Os computadores ligam-se entre si ou aos comutadores de pacotes, também através de canais de dados, e trocam pacotes directamente entre si (*end-to-end*) usando os serviços da rede. O coração da Internet é uma grande rede de pacotes, formado por inúmeras redes interligadas, e actuando em coordenação, de tal forma que essa nuvem de redes interligadas actua como uma rede logicamente única.

Compete ao software residente nos computadores aos níveis transporte e aplicação implementar os serviços distribuídos que os utilizadores utilizam, com base nos serviços do nível rede, como está ilustrado na Figura 13.15. Quando um computador cliente interage com um computador servidor para lhe pedir uma página Web, os dois dialogam “directamente” usando de forma transparente os serviços do nível rede.

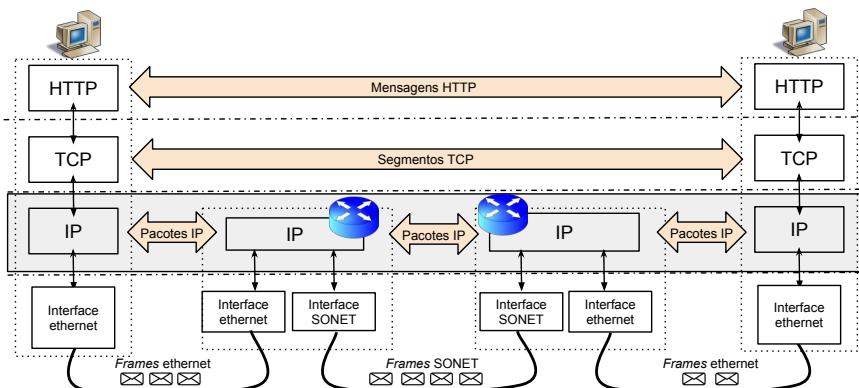


Figura 13.15: O nível rede (a sombreado) utiliza canais de dados e comutadores de pacotes para providenciar os seus serviços aos níveis superiores

Nesta parte do livro, o termo rede de pacotes, ou simplesmente rede, vai ser usado para designar essa infra-estrutura de encaminhamento de pacotes da origem até ao destino. Uma rede pode ser muito simples, baseada num único canal multi-ponto que interliga vários computadores, pode ser mais complexa, sendo constituída por uma malha de canais e nós de comutação de pacotes, ou pode ser ainda mais complexa, sendo formada por uma interligação de redes diferentes, uma rede de redes ou um *internetwork* na terminologia em língua inglesa, e assim sucessivamente. O termo é usado de forma recursiva em diferentes situações, que o leitor terá de distinguir pelo contexto da discussão.

Desenhar uma rede envolve a solução de vários problemas. Primeiro é necessário definir um esquema de endereçamento que permita indicar a origem e o destino dos pacotes. No caso das redes muito simples, isso é simples de resolver pois o único requisito é que os endereços sejam diferentes uns dos outros. Se a rede é mais complexa, o seu funcionamento interno pode impor a necessidade de alguma abstração de detalhes através da utilização de uma hierarquia de endereços. Diferentes partes da rede olharão para diferentes partes do endereço, o que é sempre uma boa medida para promover a escalabilidade. Quando a rede é uma interligação de redes, com uma escala muito

grande como a Internet, o problema do endereçamento pode tornar-se ainda mais complexo.

Um segundo problema que a rede tem de resolver é determinar caminhos para encaminhar pacotes com sucesso até ao destino. Este problema também pode ter soluções simples, mas no caso geral, é bastante mais complexo pois a escolha do caminho tem de satisfazer critérios de correção; *i.e.*, os pacotes chegam aos destino previsto, de eficiência; *i.e.*, de alguma forma tenta-se optimizar o funcionamento do sistema; e tem-se também de encontrar formas de lidar com as falhas das componentes da rede (canais e nós de comutação) e eventualmente também com as perdas de pacotes. Finalmente, a rede tem de ser monitorada, gerida, e continuamente adaptada a novos utilizadores e a novos requisitos.

Uma rede viária é uma boa analogia para o nível rede. No mesmo continente, é possível iniciar uma viagem de automóvel numa rua de uma cidade num dado país, e terminá-la numa aldeia de outro país. Pelo meio o automóvel atravessou ruas numa cidade, entrou em estradas nacionais, passou para auto-estradas, atravessou fronteiras, passou por estradas municipais e finalmente chegou à aldeia. Para chegar com êxito ao destino é necessário conhecer o seu endereço. Para isso usam-se convenções comuns nos diferentes países. Todas as estradas usadas permitiram a circulação daquele automóvel, e finalmente, nos cruzamentos de estradas, houve a capacidade de tomar as decisões correctas.

Do ponto de vista do condutor do automóvel o processo é relativamente simples. No entanto, do ponto de vista da construção e gestão da rede viária o problema é muito mais complexo. Numa pequena cidade ou numa vila, a rede viária é relativamente fácil de construir e gerir. Trata-se de uma pequena rede. Do ponto de vista das grandes cidades e suas interligações, o problema é mais complexo. Por um lado temos redes viárias muito densas e carregadas, e por outro temos interligações dessas redes, que são elas próprias outras redes, ou seja redes de trânsito, pois as mesmas não têm edifícios, origem ou destino de trajectos, apenas existem para que se circule nelas.

Por isso, o estudo do nível rede também é muito multi-facetado pois existem diferentes tipos de redes com diferentes tipos de requisitos, e existe também uma interligação de todas essas redes. Os capítulos que formam esta parte do livro são vários, nomeadamente:

**Capítulo 14** introduz os canais baseados em difusão, ou multi-ponto, e como estes podem ser usados para construir redes simples que permitem a muitos computadores trocarem pacotes directamente entre si usando um único canal. Neste tipo de redes não é necessário fazer o encaminhamento de pacotes recorrendo a nós de comutação de pacotes pois um único canal é suficiente para assegurar a comunicação extremo-a-extremo. Na analogia com as redes viárias estes canais são como ruas de cidades.

**Capítulo 15** discute as formas mais simples de fazer encaminhamento, nomeadamente recorrendo a técnicas de inundação (*flooding*). Apesar de estas técnicas parecerem demasiado ingénugas é possível aproveitar todas as suas vantagens em contextos particulares, onde também é fácil contornar os seus inconvenientes. Em redes com um âmbito geográfico limitado e sem grandes constrangimentos de capacidade, está técnica é popular e revela-se bastante adequada. Na analogia com a rede viária estas redes são como localidades de pequena ou média dimensão.

**Capítulo 16** introduz técnicas, algoritmos e protocolos de encaminhamento para redes mais complexas, mas baseados numa solução de compromisso relativamente simples: os pacote são encaminhados pelo caminho mais curto. Esta maneira de conceber o encaminhamento é um bom compromisso para qualquer tipo de rede, desde que a mesma esteja bem dimensionada para o tráfego injectado e

não existam canais saturados. Na analogia com a rede viária estas redes são como regiões de média dimensão ou como pequenos países.

**Capítulo 17** discute as formas de endereçamento mais adequadas a redes simples, sem necessidade de administração de endereços, assim como as formas de endereçamento que são requeridas em redes de grande dimensão e hierárquicas. Discute também diversas facetas da relação entre o endereçamento e a liberdade de movimentos dos clientes entre redes e fornecedores de serviço. Depois o capítulo introduz o protocolo IP e a forma como este é usado para realizar o encaminhamento dentro e entre as redes da periferia da Internet.

**Capítulo 18** introduz como é realizado o encaminhamento entre as diferentes redes que formam a Internet, *i.e.*, como são escolhidos os caminhos nas auto-estradas de interligação das diferentes redes que a formam. Por um lado, a este nível, os requisitos do encaminhamento são complexos pois é necessário satisfazer critérios de optimalidade mas também critérios relacionados com questões comerciais (quem paga a quem os serviços de transporte de pacotes) e políticos (será que estes pacotes podem atravessar esta rede?). Estes factores requerem algoritmos e um protocolo de encaminhamento especial, o protocolo BGP (*Border Gateway Protocol*). Na analogia com a rede viária estas redes são como as redes de auto-estradas, que eventualmente cruzam diversos países, e interligam regiões densamente povoadas.



## Capítulo 14

### *Redes baseadas em canais de difusão*

*Metcalfe's Law: "The value of a network grows as the square of the number of its users."*

– Autor: Robert Metcalfe, inventor of the Ethernet network

Os canais baseados em difusão (*broadcasting*), ou multi-ponto, permitem que muitos computadores partilhando um meio de comunicação comum possam comunicar directamente uns com os outros; por essa razão permitem realizar redes muito simples como ilustra a Figura 14.1. Com efeito, se  $n > 2$  computadores estiverem directamente ligados por um canal deste tipo,  $n$  computadores podem comunicar directamente sem necessidade de comutadores de pacotes, ver o Capítulo 3. Quando um computador transmite uma mensagem, todos os outros ligados ao canal vêm o seu conteúdo. No entanto, só o destinatário, i.e., o computador com o endereço do destino da mensagem, a guarda. Adicionalmente, estes canais suportam directamente a comunicação de um para todos, isto é, quando um emissor transmite uma mensagem para o endereço de difusão (*broadcasting*), esta é recebida por todos os outros.

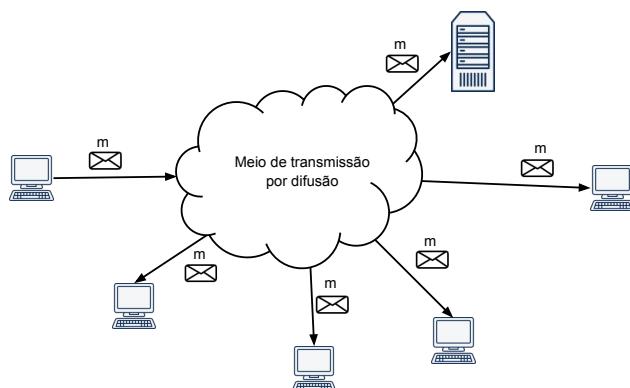


Figura 14.1: Transmissão de uma mensagem por um canal baseado em difusão através de um meio de comunicação partilhado

Como foi referido no Capítulo 2, um canal diz-se *simplex*, se só permite transmitir

informação num só sentido e diz-se *full-duplex* se permite a transmissão em simultâneo em todos os sentidos; um canal ponto-a-ponto *full-duplex* é equivalente a dois canais *simplex*, um em cada sentido. Um canal *half-duplex* é um canal que liga vários interlocutores mas em que só um pode emitir em cada momento. Os canais que iremos estudar neste capítulo são canais *half-duplex*.

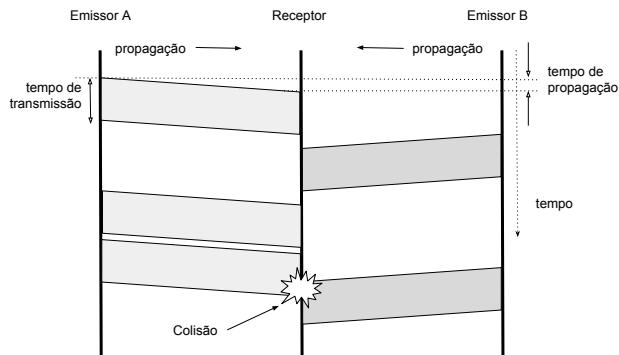


Figura 14.2: Emissores ligados ao mesmo meio de comunicação - se as emissões tiverem lugar em momentos diferentes, não há colisão; senão os *frames* colidem e o receptor recebe-os com erros

Com efeito, ver a Figura 14.2, como há vários emissores que partilham o mesmo meio de comunicação (o *transmission media*), estes têm de transmitir *frames* em momentos diferentes, sob pena de os sinais emitidos por cada um se cruzarem num potencial receptor, que será incapaz de reconhecê-los. Esta mistura dos sinais que suportam a codificação de cada um dos *frames* chama-se uma **colisão** (*collision*) e inutiliza todos os *frames* que colidam.

Assim, estes canais necessitam de mecanismos e protocolos que impeçam acessos simultâneos ao meio de comunicação. Estes são designados protocolos de **controlo de acesso ao meio** e são abreviados pela sigla em língua inglesa **MAC – Medium Access Control**. No essencial, é algo semelhante ao que é necessário usar durante a realização de uma reunião para evitar que os presentes falem todos ao mesmo tempo, *i.e.*, trata-se de um mecanismo de regulação do direito a usar a palavra.

Os meios de transmissão usados pelos canais de difusão são vários. O mais comum é a atmosfera pois, quando não se usam antenas direcionadas, este meio permite que um sinal emitido seja recebido simultaneamente por muitos receptores. Alguns exemplos de canais deste tipo, já referidos no Capítulo 2, são canais de dados via satélite, assim como os banais canais Wi-Fi. No entanto, também existem canais baseados em difusão cujo meio de comunicação é baseado em meios guiados como cabos de cobre ou fibras ópticas.

A probabilidade de existirem colisões é muito dependente de vários factores. Estamos a admitir que os canais que nos interessam são canais de dados e portanto os emissores geram pacotes de forma muito irregular e não necessariamente coordenada, podendo alternar entre períodos sem actividade e períodos com muita actividade. Um *frame* com 1 Kbyte tem cerca de 8.000 bits e é transmitido em aproximadamente 80  $\mu$ s a 100 Mbps e em aproximadamente 8  $\mu$ s a 1 Gbps. Logo, se o tráfego for pouco intenso e todas as interfaces transmitirem de forma independente e não sincronizada (o que nem sempre é verdade), a probabilidade de colisões é baixa. No entanto, se o tráfego for intenso a probabilidade de colisões já é mais elevada.

Mesmo não havendo colisões, qual a probabilidade de os *frames* serem recebidos

com erros? No Capítulo 2 vimos que nos canais guiados, e sobretudo nas fibras ópticas, essa probabilidade é pequena e por isso deixa-se aos níveis superiores a sua correcção (*e.g.*, ao protocolo TCP). No entanto, nos canais sem fios essa probabilidade é muito elevada e provavelmente é melhor tentar corrigi-los logo ao nível de cada canal, quer através de códigos de correcção de erros, quer através de retransmissão. A razão é que a deteção dos erros ao nível extremo a extremo leva muitas vezes centenas de milissegundos enquanto que ao nível do canal pode levar apenas alguns micro segundos.

Em todos estes canais os receptores apercebem-se através dos códigos de controlo de erros da presença de colisões pois recebem sinais de mais do que um emissor misturados. A mesma questão se pode colocar com respeito ao emissor: será que este detecta a colisão? Para ser capaz de o fazer tem de ser capaz de comparar o sinal que emite com o recebido. Se houver diferenças significativas, então houve uma colisão. Nos canais sem fios, o sinal degrada-se exponencialmente com a distância percorrida e os obstáculos atravessados. Nestes canais os emissores não detectam as colisões pois só se “ouvem a eles próprios”. Com efeito, o sinal emitido pode ser centenas de milhares de vezes mais potente que o recebido.

Nos canais de difusão baseados em fios, os emissores conseguem detectar as colisões pois o sinal não se degrada de forma significativa a distâncias curtas (ou mesmo nas longas no caso das fibras ópticas). Assim, nestes canais é possível ao emissor detectar as colisões. No entanto, essa detecção está dependente do tempo de propagação do sinal pois, se o tempo de emissão for inferior ao tempo de propagação, será mais difícil detectar colisões.

Com efeito, se um canal de difusão com fios tiver 100 metros de comprimento, o tempo de propagação é de cerca de  $100 \times 1/2.10^{-8} \approx 0,5 \mu\text{s}$  (micro segundos) visto que o sinal se propaga a cerca de 200.000 Km por segundo num meio de comunicação guiado, ver o Capítulo 2. No entanto, 10 bytes são transmitidos em  $80 \times 10^{-8} \approx 0,08 \mu\text{s}$  a 1 Gbps, o que queria dizer que neste caso a colisão poderia não ser detectada pelo emissor visto que os dois sinais não se cruzariam simultaneamente na sua interface.

Os canais baseados em difusão pela atmosfera são muitos e variados pois são usados para difusão da rádio e televisão, nas redes de telemóveis e para comunicação de dados entre *smartphones*, nas redes Wi-Fi, nas redes de sensores, nos automóveis, *etc.* Existem também canais baseados em difusão que usam cabos coaxiais, fibras ópticas e satélites. A gama de soluções usadas para regular o acesso ao meio de comunicação é muito variada e ultrapassa claramente o âmbito deste livro. Neste capítulo vamos apenas analisar uma classe de canais deste tipo que usam mecanismos baseados em aleatoriedade, e que foram desenvolvidos com o objectivo principal de permitirem a comunicação de dados entre computadores.

O capítulo começa por discutir os requisitos que os protocolos MAC para comunicação de dados têm de satisfazer, delimita o tipo de canais que vamos analisar, e analisa os mecanismos que podem ser usados para coordenar o acesso ao meio de comunicação. Depois desta introdução, são apresentadas duas tecnologias usadas na maioria dos computadores e *smartphones* actuais, as tecnologias Ethernet (normas IEEE 802.3) e Wi-Fi (normas IEEE 802.11).

Os canais baseados em difusão funcionam necessariamente no modo *half-duplex* pois se várias interfaces emitem simultaneamente, sobre um mesmo meio de comunicação partilhado, dá-se uma **colisão** dos sinais que provoca a recepção de *frames* com erros e que por isso ficam inutilizados e têm de ser postos de lado, *i.e.*, ignorados. Para resolver este problema é necessário um mecanismo de **controlo de acesso ao meio**, que é abreviado pela sigla em língua inglesa **MAC** – *Medium Access Control*.

Existem muitas tecnologias de canais *half-duplex* baseados em difusão que utilizam vários tipos de meios de comunicação como a atmosfera ou meios guiados (cabos de

cobre ou de fibra óptica). Nos canais sem fios, ao contrário dos canais baseados em meios guiados, a probabilidade de erros é muito elevada, e pode não ser possível um emissor detectar se o canal está ou não ocupado, ou se houve ou não colisão.

## 14.1 Requisitos e possíveis soluções

Os protocolos MAC devem satisfazer requisitos comuns aos mecanismos flexíveis de multiplexagem de canais de dados de forma flexível. Assim, dado um canal multi-ponto com a capacidade ou débito  $D$ , ao qual estão ligadas várias interfaces de comunicação, então:

1. se só uma interface emissora pretende emitir mensagens, então esta deve poder utilizar a totalidade da capacidade do canal, *i.e.*, o débito  $D$ ;
2. se  $n$  interfaces pretendem emitir mensagens, então cada uma deve poder utilizar em média a fração  $D/n$  da capacidade do canal;
3. o protocolo MAC deve ter um *overhead* baixo, e não consumir ele próprio uma fração significativa da capacidade  $D$ ;
4. o protocolo MAC deve ser simples e escalável, ou seja, adaptar-se facilmente a um número variável de interfaces; e por fim,
5. opcionalmente, o protocolo MAC deve suportar um sistema de prioridades entre as interfaces.

### Classes de mecanismos e de protocolos MAC

Existem duas grandes famílias de mecanismos e protocolos MAC: os centralizados e os distribuídos. Os protocolos centralizados são caracterizados por se basearem num árbitro central que “dá a palavra aos diferentes participantes” (como a mesa de uma assembleia). Os distribuídos baseiam-se em coordenação distribuída, isto é, o canal é ele próprio usado para coordenar (“dar a vez”) aos diferentes participantes, que nesse caso estão numa posição semelhante uns aos outros.

De forma geral, a indústria de computadores, ao contrário da indústria de telecomunicações, optou sobretudo por soluções distribuídas, em que as funções de arbitragem existem em todas as interfaces ligadas ao canal. Inicialmente foram comercializadas duas abordagens distribuídas distintas por esta indústria. Uma mais complexa, baptizada “Token Ring”, baseada num mecanismo de coordenação distribuída que afecta o direito de emitir “à vez” em carrossel (*round robin*), mas que foi comercialmente abandonada. E outra, baseada na selecção aleatória da próxima interface a emitir, que foi e continua a ser muito utilizada.

No resto deste capítulo vamos estudar dois exemplos de canais baseados em protocolos MAC distribuídos com abordagens de afectação aleatória do canal: Ethernet clássica e Wi-Fi. Começaremos, no entanto, por abordar primeiro diversos mecanismos que são usados nesses protocolos. O tratamento apresentado é simplificado e serve apenas para perceber de forma qualitativa os mecanismos que vão ser utilizados nas duas tecnologias apresentadas a seguir. Na Secção 14.4 serão indicadas referências que apresentam tratamentos mais aprofundados e completos do tema.

### Mecanismos usados pelos protocolos MAC aleatórios distribuídos

Os protocolos MAC aleatórios distribuídos residem exclusivamente nas interfaces de ligação ao canal, e implementam um sistema de coordenação dos emissores em competição pelo canal sem recurso a árbitros externos.

Em todas essas abordagens cada interface emissora processa um *frame* de cada vez. Assim, até que cada *frame* seja recebido pelo destinatário, ou abandonado por

sucessivos erros, a interface emissora fará várias tentativas até o processar, e novos *frames* que cheguem à mesma interface para serem emitidos ficarão à espera de vez numa fila de espera.

Numa primeira fase vamos assumir as condições mais desfavoráveis: a taxa de erros é elevada, e o emissor não consegue detectar se o meio está ocupado, nem se houveram colisões. A solução mais simples consiste em logo que a interface recebe um *frame* para transmitir, começar a emitir-lo imediatamente sem esperar (excepto se ainda estiver a processar o *frame* anterior). Coloca-se então a questão de saber se o *frame* chegou ao destinatário, ou se foi destruído por erros ou colisões.

Uma solução simples para ambas as dúvidas é adoptar o mesmo procedimento que o protocolo *stop & wait*, i.e., o destinatário, caso tenha recebido bem o *frame*, envia um ACK ao emissor. Os **frames de ACK**, sendo muito curtos, terão uma probabilidade inferior aos de dados de entrarem em colisão. No entanto, em geral, também se poderão perder por erros ou colisões.

Este funcionamento requer um sistema de numeração das interfaces ligadas ao canal. O único requisito desses endereços é serem diferentes uns dos outros, mantendo-se a unicidade mesmo quando diversas interfaces podem ligar-se ou desligar-se do canal dinamicamente e sem aviso. Os *frames* têm endereços de destino e de origem no cabeçalho, ver a Secção 2.5. Um endereço de destino especial, chamado o endereço de *broadcast*, assinala que o *frame* se destina a todas as interfaces.

Quando uma interface recebe um *frame*, só se o endereço de destino é o seu (ou o de *broadcast*) é que esta o memoriza e processa. Se o endereço de destino for o endereço específico da interface, esta envia um ACK. Os *frames de broadcast* não são acked pois isso desencadearia a emissão de uma “tempestade” de ACKs.

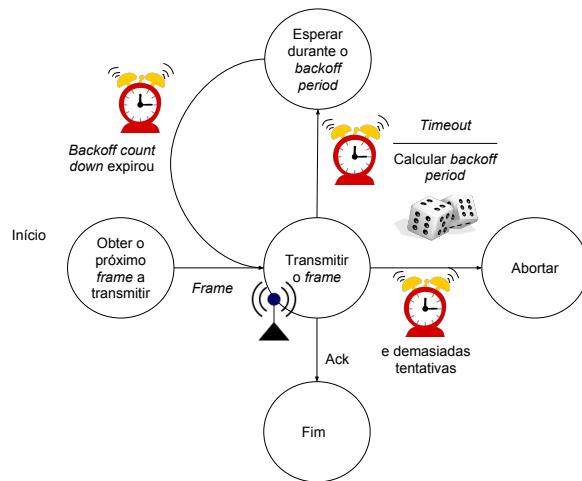


Figura 14.3: Máquina de estados usada no protocolo MAC ALOHA para a emissão de cada *frame*

Assim, tal como no protocolo *stop & wait*, o emissor instala um alarme temporizado (*timeout*) e espera pelo ACK. Se este não vier a tempo, reemite o *frame* até um certo número de tentativas. Se o *frame* se tiver perdido, o reemiti-lo imediatamente seria a receita para o desastre se os vários emissores usassem o mesmo valor de duração do alarme (*timeout*), pois continuariam eternamente a emitir simultaneamente, de forma sincronizada, e provocando novas colisões.

A solução para este desastre foi uma espécie de “ovo de Colombo” e consistiu

em tentar usar compassos de espera diferentes pelos diferentes emissores. Isso pode ser obtido usando um gerador de números aleatórios e adicionando ao *timeout* um **compasso de espera aleatório** suplementar que se chama o *backoff period*. A interface que obtiver o valor de *backoff period* mais baixo, emitirá primeiro que as outras. A Figura 14.3 mostra o diagrama da máquina de estados, ou simplesmente diagrama de estados, do processamento de cada *frame* por uma interface emissora. Este método foi baptizado de “ALOHA puro”<sup>1</sup>.

Este protocolo MAC satisfaz de forma variável os critérios enunciados mais acima. Nomeadamente, se só um emissor pretender emitir, ele tem o canal à disposição e, se por hipótese os erros se dessessem apenas a colisões, o *overhead* será baixo. O esquema é simples e não requer nenhum árbitro centralizado caro. No entanto, é óbvio que quando existem muitos emissores a tentarem emitir, ou bem que o intervalo dos compassos de espera é muito alargado, o que aumenta muito o desperdício da capacidade do canal (*overhead*), ou então a probabilidade de novas colisões será elevada.

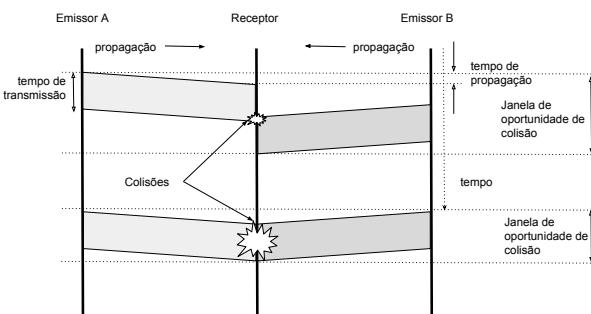


Figura 14.4: Com a versão Slotted ALOHA a oportunidade de colisões é menor

Uma primeira tentativa de melhoramento do MAC ALOHA consistiu em introduzir a noção de *slots de emissão (intervalos fixos de emissão)*: uma interface só pode emitir no início de cada *slot*; desta forma diminui-se a probabilidade de as mensagens poderem colidir como ilustra a Figura 14.4. Em contrapartida, todas as interfaces têm de ser sincronizadas para reconhecerem o início de cada novo *slot* de forma sincronizada. Isso constituiu uma complicação suplementar. Este novo MAC chama-se MAC Slotted ALOHA e a sua máquina de estados está representada na Figura 14.5.

Para que o mecanismo de resolução de colisões seja efectivo, o intervalo temporal no qual é calculado o valor aleatório do *backoff period* é proporcional a um número significativo de vezes do tempo de emissão das mensagens, ou da duração do *slot*. Apesar de o MAC Slotted ALOHA ter melhor comportamento, o resultado em qualquer caso é que o *overhead* de ambas as variantes do MAC ALOHA é claramente superior a 50% da capacidade do canal sempre que há vários emissores, mesmo no caso mais favorável, e tende para próximo de 100% quando o número de emissores simultâneos vai aumentando, pois os emissores passam cada vez mais tempo à espera de poderem emitir ou a provocar colisões. Adicionalmente, a versão Slotted ALOHA introduz desperdícios suplementares se os *frames* tiverem uma dimensão inferior ao *slot*. Por estas razões é necessário ver se é possível melhorar estes protocolos.

<sup>1</sup> O método foi introduzido na primeira rede de dados sem fios publicamente acessível, desenvolvida na Universidade do Hawaii em 1971, que foi baptizada ALOHA (*Additive Links On-line Hawaii Area*).

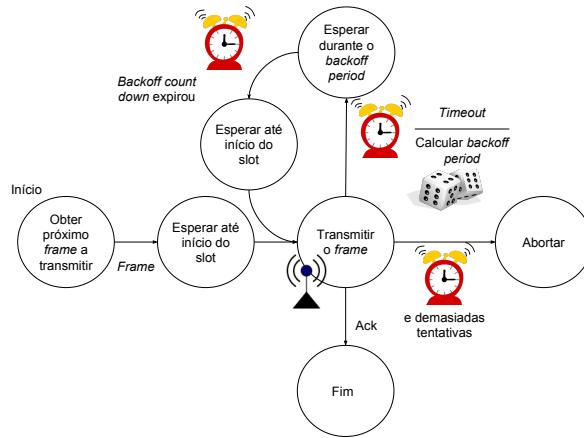


Figura 14.5: Máquina de estados usada no protocolo MAC Slotted ALOHA para a emissão de cada *frame*

### *Carrier Sense Multiple Access (CSMA)*

Este melhoramento consiste em pôr os emissores permanentemente à **escuta de actividade no meio de comunicação (carrier sense)** e impedir-lhos de usarem o canal se este estiver ocupado. Assim, se o sinal da onda portadora (*carrier*) estiver presente, nenhum outro *frame* deve poder ser emitido, o que diminui a probabilidade de colisões.

Mas isso levanta o seguinte problema: que fazer quando a outra interface terminar a sua emissão e o meio de comunicação ficar livre? Há várias hipóteses, a primeira consiste em, assim que o canal ficar livre, todas as interfaces que têm *frames* à espera de serem emitidos, começam imediatamente a fazê-lo. Os *frames* que já foram transmitidos com colisão pelo menos uma vez não interferem obrigatoriamente nesta fase pois podem ainda estar à espera que o seu compasso de espera (*backoff period*) acabe, mas os que já podem ser transmitidos, assim como os novos, já podem colidir.

Assim, quando o meio fica livre, existem duas razões que podem estar na origem de que vários emissores se sincronizem para começarem a emitir ao mesmo tempo. A primeira diz respeito aos novos *frames*, que vão ser emitidos pela primeira vez, e a probabilidade de colisão é grande se houverem vários nessa situação. A segunda diz respeito aos *frames* que já foram emitidos pelo menos uma vez, mas já estão à espera de serem emitidos de novo, pois ainda não foi recebido o seu ACK e o *backoff period* para nova transmissão já se esgotou. Esta segunda origem pode ser menos frequente visto que esse período de espera é aleatório e potencialmente diferente em cada caso.

Existem várias formas de tratar estes problemas, nomeadamente: 1. ignorá-lo e emitir assim que o meio fica livre (designada por *1-persistent*); 2. baseada em testar periodicamente se o meio está livre e transmitir se este está livre, ou voltar a testar daí a um período aleatório senão estiver (designada por *nonpersistent*); e 3. a solução aplicada à versão Slotted ALOHA (designada *p-persistent*) que consiste em transmitir com probabilidade *p* quando, no início de um *slot*, o meio fica livre. A seguir vamos apresentar uma variante concreta que inspirou a usada nos canais Wi-Fi.

### **MAC CSMA com Collision Avoidance (CSMA/CA)**

Este protocolo MAC adopta a solução que consiste em introduzir necessariamente um novo compasso de espera aleatório antes da primeira emissão de cada *frame*. Este MAC foi criado para situações em que o potencial emissor não tem a certeza se o

meio está livre ou não, como é o caso dos canais Wi-Fi por exemplo, como veremos na secção seguinte.

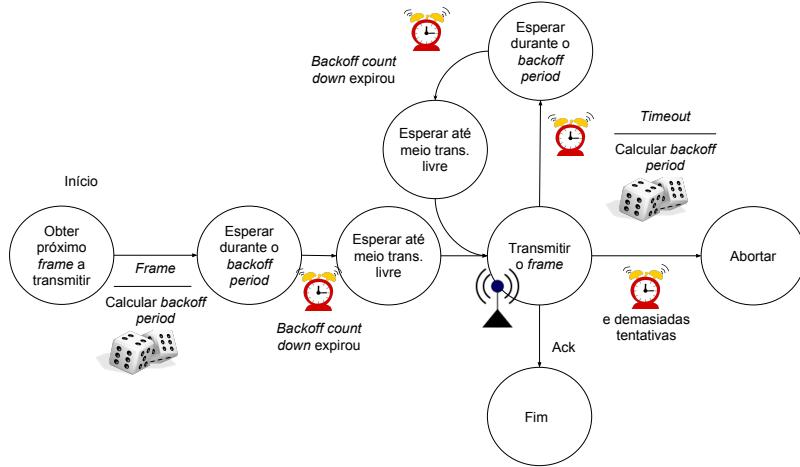


Figura 14.6: Máquina de estados usada no protocolo MAC CSMA/CA para a emissão de cada *frame*

Este período inicial suplementar de *backoff* é introduzido para aumentar a probabilidade de não haverem colisões iniciais, daí a designação *CA* - *Collision Avoidance*. A Figura 14.6 apresenta a respectiva máquina de estados.

#### Deteção de colisões pelos emissores

Em todos os MACs até agora referidos sempre que uma interface começa a transmitir, não pára a transmissão até ter transmitido integralmente o *frame*. Isso traduz-se num desperdício significativo da capacidade do canal quando ocorrem colisões. A pergunta a fazer é: não seria possível à interface emissora aperceber-se mais cedo da colisão e desta forma libertar o canal imediatamente? Como já foi referido, a grande maioria dos canais sem fios não permitem a deteção de colisões pois frequentemente o sinal emitido é vários milhares de vezes mais forte que o recebido. Adicionalmente várias características dos canais sem fios impedem essa deteção.

No entanto, o mesmo não se passa com os canais com fios. Nestes, dado que o sinal se propaga com pouca resistência pelo meio de comunicação, o emissor, durante a emissão, recebe o sinal emitido eventualmente misturado com o seu, e pode verificar se o sinal emitido e o sinal por ele recebido coincidem ou não. Se a resposta for negativa, é porque houve uma colisão.

#### CSMA/CD - CSMA com Collision Detection

Esta variante de protocolo MAC é especialmente adequada a canais com fios onde é fácil detectar as colisões de forma segura. Por outro lado, neste tipo de canais, se não houver colisão até ao fim da transmissão de um *frame*, então a probabilidade de não existirem erros e o mesmo ser bem recebido é muito elevada. Assim, pode-se igualmente abdicar do uso do ACK e deixar aos níveis superiores a eventual recuperação de erros. A Figura 14.7 apresenta a máquina de estados do protocolo CSMA/CD.

Uma questão suplementar interessante é saber durante quanto tempo ( $\delta t$ ) tem de durar a emissão de uma interface para que essa emissão evite que outras interfaces possam começar a emitir. Ou seja, se uma interface *I* conseguir emitir sozinha durante

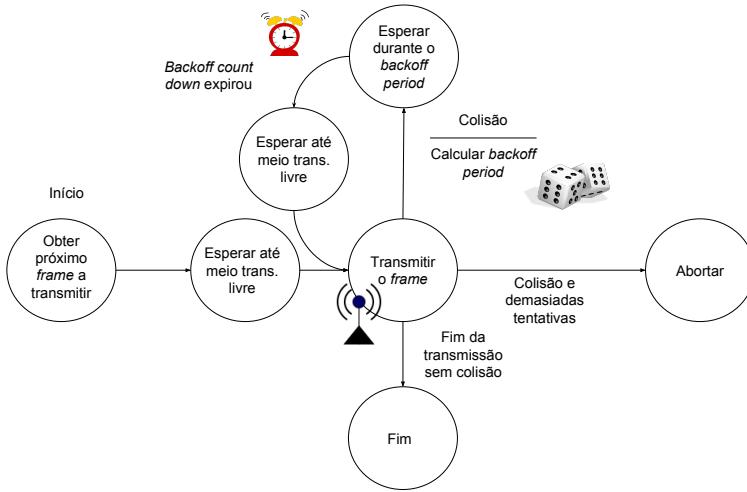


Figura 14.7: Máquina de estados usada no protocolo MAC CSMA/CD para a emissão de cada *frame*

$\delta t$ , então todas as outras interfaces detectam essa emissão e abstêm-se de emitir, pelo que  $I$  como que reservou o canal. O valor de  $\delta t$  está relacionado com o tempo de propagação, como veremos a seguir, e condiciona a dimensão mínima dos *frames*.

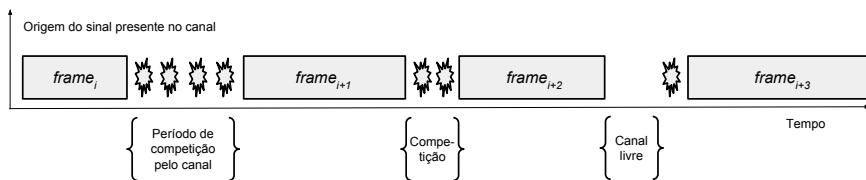


Figura 14.8: Evolução no tempo da utilização de um canal de difusão partilhado pelo protocolo CSMA/CD

Com o protocolo CSMA/CD podemos representar um diagrama temporal do estado da utilização do canal como o da Figura 14.8. Inicialmente o canal está ocupado a transmitir o  $frame_i$ . No fim, como existem diversas interfaces com *frames* para transmitir, segue-se um período de competição pelo canal com várias colisões. No final desse período de competição, uma das interfaces ganha e o  $frame_{i+1}$  é transmitido. Segue-se um novo período de competição, que termina com o início da transmissão do  $frame_{i+2}$ . No final da transmissão deste *frame*, nenhuma interface tem *frames* para transmitir, até que surge de novo uma situação em mais do que uma tem, e estas entram de novo em competição com um novo período de contenção. Este é seguido pela transmissão do  $frame_{i+3}$  pela interface que ganhou a competição.

Naturalmente, dadas as suas características, as variantes CSMA/CA e CSMA/CD desperdiçam uma menor capacidade do canal com a coordenação dos emissores que as variantes ALOHA. Em particular, a variante CSMA/CD sobre canais guiados satisfaz a maioria dos requisitos apresentados no início desta secção com *overhead* inferior ao das outras alternativas, particularmente devido às condições mais favoráveis em que é

utilizada, *i.e.*, deteção da ocupação do meio com maior fiabilidade, detecção de colisões e baixa taxa de erros do canal.

Os diferentes protocolos MAC acima apresentados foram caracterizados apenas de forma qualitativa com o objectivo de por em evidência alguns dos diferentes mecanismos a que os mesmos recorrem. A seguir vamos estudar de forma mais detalhada dois destes protocolos e a forma como esses mecanismos são concretizados em canais específicos.

Os canais baseados em difusão funcionam em modo *half-duplex* e por isso utilizam um protocolo MAC para regular e coordenar o acesso ao meio de comunicação.

Estes protocolos são vários e utilizam diferentes tipos de mecanismos, nomeadamente: ***frames de ACK*** quando a probabilidade de erro e de colisões é mais elevada, compassos de espera aleatórios (***Random Backoff Periods***) em caso de deteção de colisões ou antes de transmitirem um *frame* pela primeira vez, sondagem do meio de comunicação para verificar se este está livre ou ocupado (***CS - Carrier Sense***), e mecanismos de deteção de colisões (***CD - Collision Detection***).

Uma associação específica de diferentes mecanismos é usada em função do meio de comunicação e de outras características necessárias em diferentes contextos. Os protocolos MAC, quando têm por objectivo a interligação de computadores, foram normalizados pela IEEE, na família de protocolos com o prefixo **IEEE 802**.

## 14.2 Canais de dados Ethernet (IEEE 802.3)

Tendo como inspiração o protocolo ALOHA, Robert Metcalfe and David Boggs, na altura a trabalharem no Xerox Parc Research Center<sup>2</sup>, inventaram uma rede, a que chamaram Ethernet [Metcalfe and Boggs, 1976]. Esta rede consistia num único canal de difusão baseado num cabo coaxial partilhado por (até) várias centenas de interfaces. O MAC usado foi o protocolo CSMA/CD.

Estes canais já foram brevemente apresentados na Secção 2.5, onde o leitor poderá consultar o formato dos endereços e dos *frames* usados, e onde também é discutida a dimensão máxima dos mesmos. Resumidamente, um *frame* Ethernet tem o formato indicado na Figura 14.9. Uma das primeiras versões normalizadas e comercializadas destes canais e interfaces foi desenvolvida no seio da IEEE através de um conjunto de normas com o prefixo 802.3, daí esses canais também serem designados canais IEEE 802.3.

Os endereços de nível canal, ou nível MAC (*MAC addresses*), das interfaces Ethernet têm 48 bits e são afectados pelos seus fabricantes de forma que garantem que os mesmos são únicos no mundo pois cada fabricante tem de adquirir intervalos privados de endereços junto da IEEE. Desta forma está garantido que quando quaisquer duas interfaces deste tipo são ligadas ao mesmo meio de difusão, as mesmas terão necessariamente endereços canal distintos<sup>3</sup>.

A concretização do MAC CSMA/CD nas interfaces Ethernet passa por garantir que se o menor *frame* possível possa ser transmitido sem que o seu emissor tenha

<sup>2</sup>O Xerox Parc era um laboratório de investigação da Xerox na Califórnia, em Palo Alto, no Silicon Valley.

<sup>3</sup>A maioria dos *drivers* das interfaces Ethernet nos sistemas de operação actuais permitem alterar esses endereços pelo que nesse caso a unicidade não está garantida, o que pode levantar um problema de segurança.

Figura 14.9: Formato dos *frames* Ethernet

detectado uma colisão, então é porque todas as outras interfaces ligadas ao mesmo canal detectaram esta emissão e abstêm-se de interferir. Interessa que a duração deste *frame* seja a menor que possível dado que um *frame* pode ter uma parte de dados muito pequena. Na Ethernet 802.3 essa dimensão mínima é de 64 bytes ou 512 bits. A seguir mostra-se como este valor foi fixado.

Repare-se que é preciso garantir que a emissão com sucesso (sem colisão) de um *frame* de dimensão mínima adquiriu o canal só para ele através de uma espécie de *lock*. A fixação desta dimensão está relacionada com o tempo de transmissão e o tempo de propagação, *i.e.*, o tempo de transmissão do *frame* mínimo ( $T_t$ ) tem de ser maior ou igual a duas vezes o tempo de propagação ( $T_p$ ). A Figura 14.10 mostra porquê.

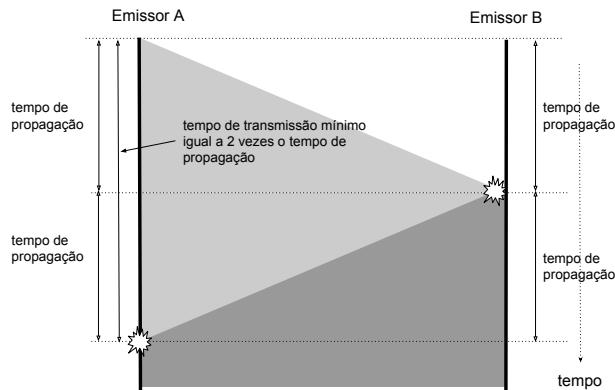


Figura 14.10: O emissor A começa a emitir, mas mesmo antes que o seu sinal chegue a B, este, julgando o meio livre, começa a emitir. B detecta imediatamente a colisão, mas A só detecta a colisão quando o sinal de B se propagar até ele, *i.e.*, ao fim de  $2 \times T_p$ .

A rede Ethernet inicial fixou a dimensão do *frame* mínimo em 64 bytes pois a norma admitia que o meio de propagação do sinal poderia ter no máximo 2.500 metros. Duas vezes o tempo de propagação extremo a extremo máximo corresponde a  $51\mu s$  ( $5.10^3 / 2.10^8$  s). A 10 Mbps, que era o débito dos canais Ethernet iniciais, neste tempo pode-se transmitir 511 bits ou aproximadamente 64 bytes ( $8 \times 64$  bits / $10^7$  bps =  $51\mu s$ ). Ao tempo de transmissão mínimo chamou-se o ***Collision Slot Time*** ou ***Contention Slot Time (CST)*** e corresponde, como já vimos, a  $51,2 \mu s$  a 10 Mbps.

Quando uma interface detecta uma colisão, emite um curto sinal especial chamado *jam signal*, que se destina a assinalar a todas as outras interfaces que aquele *frame*

não é válido.

Com a evolução da Ethernet foi possível aumentar o débito do canal para 100 Mbps. Neste caso o *frame* mínimo passaria a ter 640 bytes o que seria demasiado grande e levaria a um grande desperdício pois muitos *frames* são inferiores àquela dimensão. A alternativa consistiu em impor uma dimensão máxima do meio de comunicação de 250 metros. Assim, o *Collision Slot Time* (CST) a 100 Mbps passou a ser de 5,12  $\mu$ s e nesse período é possível emitir 512 bits ou 64 bytes a 100 Mbps. Desta forma a dimensão mínima dos *frames* continuou a ser independente do débito dos canais Ethernet.

Um outro aspecto muito relevante na implementação do MAC CSMA/CD tem a ver com a forma como o *backoff period* é calculado. Este período é proporcional a  $n \times$  CST, onde  $n$  é calculado de forma aleatória através de um método que consiste em ir alargando o intervalo em que se calcula cada número aleatório, conforme o número de tentativas de resolução da colisão já realizadas anteriormente sem êxito. Após a primeira colisão o intervalo é [0, 1], após a segunda colisão passa para [0, 3], ..., após  $k$  colisões passa para  $[0, 2^k - 1]$ . Este método designa-se um ***binary exponential backoff*** e destina-se a acomodar uma forma mais inteligente de resolver as colisões.

Inicialmente, há a esperança de que só duas interfaces estejam em competição e por isso tenta-se resolver a colisão usando compassos de espera de 0 ou  $1 \times$  CST (uma espécie de “moeda ao ar”). Caso suceda uma nova colisão, alarga-se o intervalo em que se tenta resolver a colisão usando compassos de espera de 0, 1, 2 ou  $3 \times$  CST, e assim sucessivamente. Com  $k = 10$  o intervalo é agora de 0 a  $1023 \times$  CST, o maior intervalo possível indicado pela norma. A norma também especifica que não se devem tentar mais do que 15 vezes (ou 16 se contarmos a transmissão inicial). Após este limite considera-se impossível transmitir qualquer *frame* e este é abandonado. Ao intervalo no qual se calcula o compasso de espera chama-se a *collision window*. A Listagem 14.1 apresenta o pseudo código do protocolo CSMA/CD.

Listing 14.1: Pseudo-código do algoritmo CSMA/CD executado pelas interfaces IEEE 802.3

```

1 CST = 0.000051; // 10 Mbps
2
3 boolean CSMA_CD_TransmitFrame (frame) {
4     k = 1;                                // transmission attempts
5     while ( true ) {
6         wait_for_free_media();             // carrier sense
7         transmit ( frame );            // try transmission
8         if ( success ) return true;    // end with success
9         transmit ( jam_signal );      // else collision
10        if ( k == 16 ) return false;   // unsuccess, abort transmission
11        else k++;                   // one more try
12        cw = min( 2^k - 1, 1023);    // collision window <= 1023
13        n = random ( 0 .. cw );
14        wait ( n*CST );
15    }
16 }
```

Na primeira versão da Ethernet o meio de comunicação através do qual a difusão era realizada era um cabo coaxial, semelhante aos cabos coaxiais (ver o Capítulo 2) utilizados ainda recentemente pelas televisões por cabo, ao qual eram ligados (através de “baixadas”) os diferentes computadores. Quando uma interface transmitia, o sinal espalhava-se pelo cabo coaxial e atingia todas as outras interfaces. A Figura 14.11 ilustra este tipo de instalação da rede Ethernet tal como esta foi usada até ao início dos anos de 1990.

Mais tarde passaram-se a usar cabos mais flexíveis, designados UTP, e com fichas RJ45 (ver o Capítulo 2), que interligavam todos os equipamentos a um dispositivo central, chamado *hub ethernet*, que funcionava como um repetidor do sinal. Quando

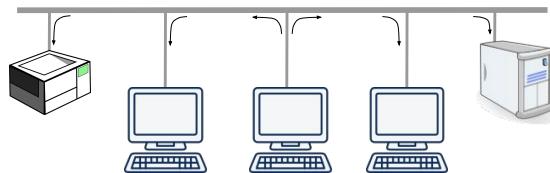


Figura 14.11: Um conjunto de computadores interligados a 10 Mbps por uma rede Ethernet 802.3 baseada num cabo coaxial partilhado entre os diferentes computadores

uma porta do *hub* recebia sinal, este amplificava-o e emitia-o para todas as outras portas. Ou seja, o meio de comunicação passou a ser constituído por uma estrela de cabos UTP com um amplificador no meio. A Figura 14.12 ilustra este tipo de instalação da rede Ethernet que passou a ser popular a partir dos anos de 1990, tendo substituído completamente as ligações baseadas em cabos coaxiais.

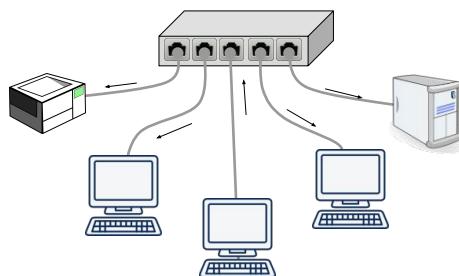


Figura 14.12: Um conjunto de computadores interligados por uma rede Ethernet 802.3 baseada em cabos UTP e um repetidor de sinal (*hub* Ethernet)

Todas estas configurações funcionavam em modo *half-duplex* através de um meio de comunicação em difusão partilhado, e realizavam o controlo de acesso ao meios através do protocolo MAC CSMA/CD, tal como acabámos de descrever.

O problema da deteção das colisões levantou questões delicadas quando foi possível construir interfaces de comunicação Ethernet com débitos de 1, 10 e 100 Gbps, como é corrente hoje em dia. Para a transmissão do sinal nos débitos mais elevados é necessário recorrer a fibras ópticas. Mas o mecanismo de detecção de colisões levaria a que a 1 Gbps o menor *frame* passasse para 640 bytes ou 4096 bits de comprimento, ou a distância entre os dois equipamentos mais afastados não poderia exceder 25 metros. A 10 Gbps, o mesmo raciocínio levaria a que se adoptasse a distância máxima de 2,5 metros *etc.*

A Tabela 14.1 mostra alguns parâmetros característicos do protocolo MAC Ethernet em função dos diferentes débitos. A tabela omite que nos débitos mais elevados podem usar-se *jumbo frames*, ver a Secção 2.5, uma opção que tem de ser explicitamente parametrizada em todos os equipamentos envolvidos.

Felizmente foi possível encontrar uma alternativa para o problema da dimensão mínima dos *frames*. Como os cabos UTP e as fibras ópticas suportam 2 canais *simplex* a transmitirem simultaneamente em cada sentido, foi possível passar a usar canais baseados nesses meios de comunicação em modo *full-duplex*, mas só ponto-a-ponto. Uma rede Ethernet passou então a poder usar simultaneamente canais *half-duplex*

Tabela 14.1: Parâmetros característicos do protocolo MAC Ethernet CSMA/CD IEEE 802.3 em função do débito das interfaces

Parâmetro	10 / 100 Mbps	1 Gbps	10 / 100 Gbps
Backoff slot time	512 bit times	4096 bit times	não se aplica
Inter Frame Space (IFS)	96 bits	96 bits	96 bits
# tentativas máximas	16	16	não se aplica
Jam size	32 bits	32 bits	não se aplica
Minimum frame size	64 bytes	640 bytes	64 bytes
Maximum frame size	1518 bytes	1518 bytes	1518 bytes

multi-ponto e *full-duplex* ponto-a-ponto. Os primeiros só podem transmitir um *frame* de cada vez entre  $n \geq 2$  interfaces. Os segundos podem transmitir simultaneamente nos dois sentidos mas apenas entre duas interfaces, *i.e.*, ponto-a-ponto. Para fazer uma rede com canais ponto-a-ponto, os *hubs* Ethernet tiveram de ser substituídos por verdadeiros comutadores de pacotes a funcionarem no modo *store & forward*, que estudaremos no Capítulo 15, e que se chamam *switches* Ethernet.

Nos débitos superiores a 1 Gbps, os canais só podem funcionar em modo ponto-a-ponto, sem protocolo MAC CSMA/CD. No débito 1 Gbps, quase sempre os canais também só funcionam em modo ponto-a-ponto, mas a norma ainda admite uma forma especial de funcionamento com o MAC CSMA/CD que é pouco usada, mas que exige *frames* de comprimento mínimo de 4096 bits ou 512 bytes, contendo eventualmente vários pacotes IP, ver Tabela 14.1.

Com a descida de preço dos equipamentos, hoje em dia a grande maioria das redes Ethernet funcionam predominantemente com canais *full-duplex* e os equipamentos com essas interfaces são geralmente interligados por *switches* Ethernet. Os canais *half-duplex* podem funcionar a 10 e a 100 Mbps (ou até 1 Gbps em casos particulares). Os canais com débitos maiores que 1 Gbps são todos ponto-a-ponto e não estão dependentes de nenhum protocolo MAC.

A Figura 14.13 mostra uma configuração típica, caracterizada pela coexistência de canais *half* e *full-duplex*. Duas sub-redes, baseadas cada uma num canal *half-duplex* materializado por um *hub* Ethernet, são interligadas por dois *switches* Ethernet ligados por um canal *full-duplex*. Este canal, é formado por dois canais *simplex*, cada um dedicado a uma direção de transmissão, e por isso não necessitam de qualquer protocolo MAC. Sempre que a interface necessitar de transmitir por um desses canais, pode fazê-lo imediatamente pois não existem competidores pelo meio de comunicação. Por essa razão, os canais Ethernet ponto-a-ponto *full-duplex* não têm limites de dimensão, a não ser os relacionados com a propagação do sinal. Usando fibras ópticas podem ter até centenas de Kms.

Esta interligação e coexistência é possível pois as interfaces modernas IEEE 802.3 baseadas em cabos de cobre conseguem funcionar nos dois modos. Em modo *half-duplex* executam o protocolo MAC CSMA/CD e partilham o meio de comunicação com outras interfaces a funcionarem no mesmo modo. Por isso se diz que esse meio assim partilhado é um domínio de colisão único.

Quando a interface está a funcionar em modo *full-duplex*, o canal só pode ligar a outra interface e não é executado nenhum protocolo MAC. Esses canais não têm colisões, são *collision free*. Os *switches* Ethernet têm geralmente interfaces que podem funcionar nos dois modos. As interfaces Ethernet actuais (*e.g.*, com conectores RJ45) nos computadores também podem funcionar nos dois modos. Na figura, os dois *switches* Ethernet têm portas *half-duplex* ligadas aos *hubs* e portas *full-duplex* que os ligam um ao outro. A configuração apresentada mostra aquilo que se costuma designar por

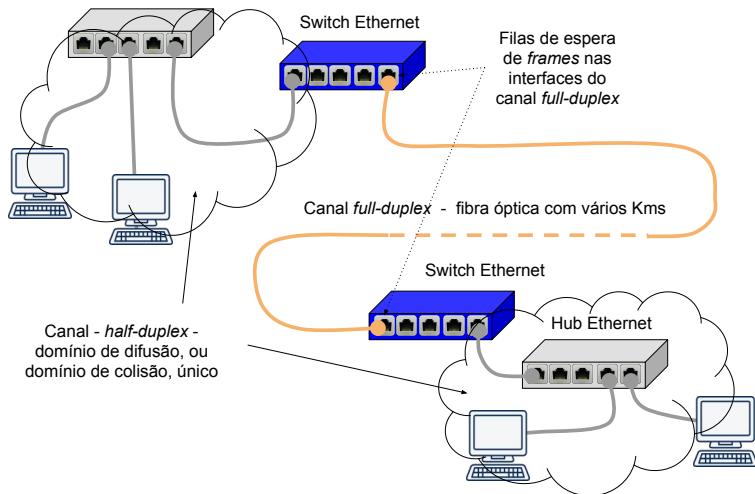


Figura 14.13: Coexistência na mesma rede Ethernet 802.3 de *hubs* e *switches* Ethernet e de canais *half* e *full-duplex*

dois domínios de colisão independentes.

Um outro aspecto interessante é analisar onde existem filas de espera. As portas dos *hubs* não têm filas de espera de *frames* porque estes são apenas meios de comunicação do sinal. Num *hub* só pode circular um *frame* de cada vez. Já as interfaces dos computadores e as interfaces dos *switches* têm associadas filas de espera para ser possível o funcionamento característico da multiplexagem estatística.

As tecnologias designadas como Ethernet nasceram nos anos de 70 do século passado, e funcionaram pela primeira vez em laboratório a 3 Mbps. Como foi apresentado acima, hoje em dia esses canais funcionam tipicamente de 100 Mbps a 100 Gbps e é provável que o débito possa vir a aumentar nos próximos anos. Todas as variantes partilham, grosso modo, o formato de *frame* e o sistema de endereçamento, os quais suportam a total interoperabilidade entre todas as versões desses canais. No que diz respeito ao protocolo MAC CSMA/CD, o mesmo é usado apenas nos canais de menor débito e a curta distância, dadas as características do protocolo MAC. Finalmente, como suportes de comunicação do sinal, podem ser usados cabos coaxiais (menos frequente hoje em dia), cabos *twisted pair* ou fibras ópticas.

As tecnologias colectivamente designadas como “Ethernet” são um exemplo notável do êxito de uma família de canais que evoluiu, e continua a evoluir, durante mais de 40 anos. O seu êxito está provavelmente relacionado com a simplicidade (*KISS - Keep it Simple, Stupid*), a compatibilidade mantida durante a evolução, a sua perfeita integração com o mundo IP, e o preço. Este último tem descido de forma consistente porque a simplicidade, os meios de comunicação usados, e um volume de produção de muitos milhões, constituem uma combinação imbatível.

Veremos a seguir que existem canais compatíveis com os canais 802.3 mas que usam como meio de propagação do sinal a atmosfera.

As normas IEEE 802.3 destinam-se a canais guiados em que é possível detectar as colisões, a fiabilidade da comunicação é elevada quando não existem colisões, e é possível reservar o meio através de um pequeno *frame* desde que a distância máxima entre interfaces seja relativamente pequena. Este conjunto de razões permitem a utilização de um **protocolo MAC do tipo CSMA/CD sem utilização de ACK**.

Uma outra faceta interessante da implementação na Ethernet do protocolo MAC CSMA/CD consistiu na adopção do método *binary exponential backoff*, que adapta a duração dos compassos de espera aleatórios em função da maior ou menor probabilidade de colisão.

Modernamente, esta família de normas abrange um conjunto de tecnologias que comportam funcionamento em *half-* e *full-duplex*. Neste último caso, não existe necessidade de utilizar um protocolo MAC, nem limitações de distância, porque o canal funciona em modo ponto-a-ponto entre apenas duas interfaces.

### 14.3 Canais de dados Wi-Fi (IEEE 802.11)

A utilização de canais sem fios é muito interessante pela comodidade que proporcionam e tornam-se imprescindíveis quando se usam equipamentos móveis. Dado o êxito da Ethernet nos escritórios e nas casas particulares, a utilização de um canal semelhante mas sem fios sempre foi muito atractiva.

Todos os canais sem fios utilizam ondas portadoras electromagnéticas (ver a Secção 2.3) que se propagam na atmosfera. Dependendo das frequências usadas e da potência da emissão, estas ondas podem atingir de alguns metros a milhares de quilómetros. Como a propagação se realiza geralmente de forma não guiada (com antenas omnidireccionais), é fácil os diferentes emissores interferirem uns com os outros. Para evitar essa situação, os canais sem fios usados pela rádio, a televisão, as redes de telemóveis e outras redes de voz móveis utilizam frequências portadoras reservadas ou licenciadas. Essas licenças são concedidas, contra pagamento, por uma autoridade de gestão do espaço radio-eléctrico<sup>4</sup>.

Para evitar as complicações e os custos do licenciamento, as normas IEEE 802.11 foram concebidas com base na utilização de intervalos de frequência deixadas livres pela regulação do espaço radio-eléctrico. Estas gamas de frequência têm a designação genérica de bandas ISM (*Industrial, Scientific and Medical bands*). Os intervalos de frequências mais usados são de 2,4 a 2,5 GHz e de 5 a 6 GHz<sup>5</sup>, que constituem uma fracção reduzida do espaço radio-eléctrico disponível e que se caracterizam por serem muito atenuadas por obstáculos. Só são permitidas baixas potências de emissão para evitar interferência entre diferentes canais e de forma a tentar confinar o respectivo âmbito a espaços privados (escritórios, casas, etc.).

As interfaces IEEE 802.11 e o software que as controlam, nesta secção designadas por equipamentos terminais, ou simplesmente **terminais**, exploram automaticamente a gama de frequências disponíveis e permitem a coexistência de vários canais de dados no mesmo espaço, através de multiplexagem em frequência (ver a Secção 3.1). Dependendo das larguras de banda de frequências usadas na multiplexagem em frequência,

<sup>4</sup> Cada país tem a sua autoridade de regulação específica. Em Portugal esse papel cabe à ANACOM, a Autoridade Nacional das COMunicações.

<sup>5</sup> As gamas reais de frequências disponíveis variam de país para país e é provável que mais gamas de frequências sejam libertadas no futuro, à medida que a pressão para a utilização de Wi-Fi suba, e que a televisão tradicional liberte frequências.

varia o débito e o número de sub-canais utilizáveis em cada uma das gamas de frequências. Para diminuir a interferência e também poupar energia nos terminais móveis, as potências de emissão são muito reduzidas, de tal forma que um canal IEEE 802.11 tem um âmbito de 20 ou 30 metros, ou menos, em espaço urbano, e de algumas centenas de metros ao ar livre (ou mais mas só se se utilizarem antenas especiais direcionadas).

Vários factores contribuem para uma elevada taxa de erros nestes canais, nomeadamente: a dificuldade das ondas com as frequências usadas atravessarem obstáculos (*e.g.*, paredes), fenómenos de reflexão que fazem com que o receptor receba várias ondas ligeiramente desfasadas, a baixa potência de emissão, a elevada resistência da atmosfera à propagação do sinal que provoca uma queda exponencial da sua potência durante a propagação, as interferências de outros equipamentos (*e.g.*, os fornos micro-ondas domésticos usam as mesmas frequências com alta potência) e a coexistência de múltiplos canais na mesma ou em frequências próximas que interferem uns com os outros.

A primeira versão da norma IEEE 802.11 data de 1997 e introduziu um canal multi-ponto sem fios a funcionar a 1 ou 2 Mbps. Mais tarde constitui-se uma aliança de fabricantes de interfaces IEEE 802.11 que tomou o nome Wi-Fi<sup>6</sup> e que está na origem do nome corrente desta tecnologia.

Apesar do baixo débito inicial, o êxito foi significativo e seguiram-se versões a 11 e 54 Mbps (IEEE 802.11a/b e IEEE 802.11g). Hoje em dia os canais Wi-Fi banalizaram-se completamente, o que tem motivado sucessivas evoluções da norma, existindo versões actuais (*e.g.*, IEEE 802.11n introduzida em 2009 e IEEE 802.11ac introduzida em 2014) que admitem débitos de centenas de Mbps ou mesmo mais em casos particulares. Veremos a seguir que dadas as características do meio de propagação, dos protocolos MAC usados e do engarrafamento do espaço urbano com vários canais Wi-Fi em competição directa, o débito extremo a extremo experimentado pelos terminais é muito mais baixo.

### Características do canal e do protocolo MAC adoptado

Devido às suas características, os canais IEEE 802.11 usam um protocolo MAC do tipo CSMA/CA. Com efeito, dadas as características do meio de transmissão, não é possível detectar colisões. A principal razão consiste no facto de que no meio usado a potência do sinal decresce rapidamente com a distância e, quando uma interface emite sinal, um sinal concorrente seria recebido com uma potência comparativamente tão baixa, que este não interferiria a ponto de ser interpretado como uma colisão. A Figura 14.14 ilustra a situação. Assim, a interface de um terminal só pode estar em dois modos alternativos: emissão ou receção.

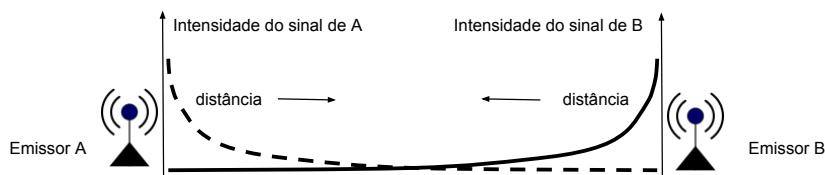


Figura 14.14: Os terminais A e B estão suficientemente afastados para não receberem os respectivos sinais com intensidade suficiente para detectarem a colisão

Por outro lado, o emissor não consegue detectar com exatidão se o meio está livre ou não, ou se houve ou não colisões. A Figura 14.15 ilustra uma situação designada

<sup>6</sup> Com origem na mote de marketing *Wireless Fidelity*™.

por “terminal escondido” (*hidden terminal*), em que um obstáculo esconde um terminal de outro (A e B não recebem as respectivas emissões) mas um terceiro (C) recebe de ambos. Este fenómeno impede a deteção da ocupação do meio com fiabilidade e também esconde colisões.

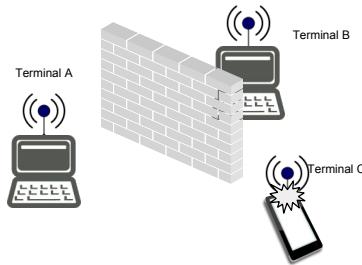


Figura 14.15: Terminal escondido. A não detecta a emissão de B e vice-versa. No entanto, C recebe ambos os sinais.

Estas não são as únicas razões que impedem a deteção de colisões ou da ocupação do canal com fiabilidade. Existem outras situações em que um terminal não detecta outro que está activo, e situações em que um terminal “acha” que outro está activo, no entanto isso não o impedia de emitir. Esta última situação é designada por “terminal exposto” (*exposed terminal*). A figura Figura 14.16 ilustra ambas as situações. Na mesma, os círculos representam o alcance do sinal emitido por cada terminal.

Devido a este conjunto de razões o protocolo MAC tem de usar ACKs para ter a certeza de uma correcta recepção e tentar compensar as eventuais colisões visto que não as consegue detectar. Por outro lado, se uma estação detectar que o meio está ocupado, não tem outra hipótese senão diferir a emissão, mesmo na situação de terminal exposto. Finalmente, quando uma estação começa a emitir, só termina quando completar a transmissão do frame.

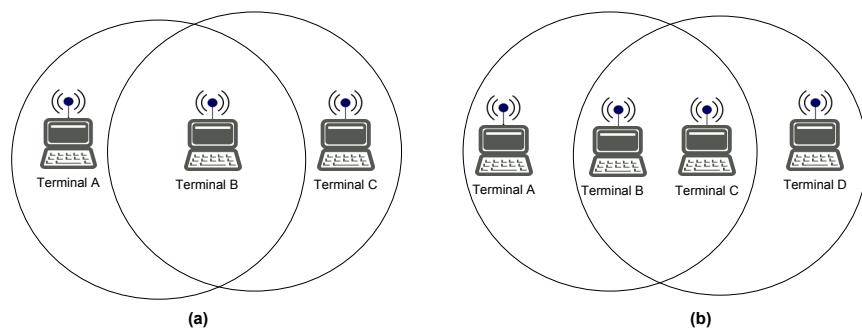


Figura 14.16: Em (a) os terminais A e C estão suficientemente afastados para não receberem os respectivos sinais com intensidade suficiente para detectarem a colisão que ocorre em B. Em (b) apesar de B e C estarem expostos aos respectivos sinais, B poderia emitir para A ao mesmo tempo que C emitiria para D.

### Funcionamento de uma rede Wi-Fi

A norma IEEE 802.11 especifica que o canal pode funcionar em vários modos distintos. No primeiro, designado **modo ad hoc**, os terminais comunicam directamente uns com os outros, ver a Figura 14.17 parte (a). No segundo, designado **modo infra-estrutura**, os terminais comunicam indirectamente e um terminal especial, designado **AP - Access Point**, que funciona como comutador de pacotes (por *store & forward* de frames) intermediário de todas as comunicações entre os terminais que lhe estão ligados, ver a Figura 14.17 parte (b).

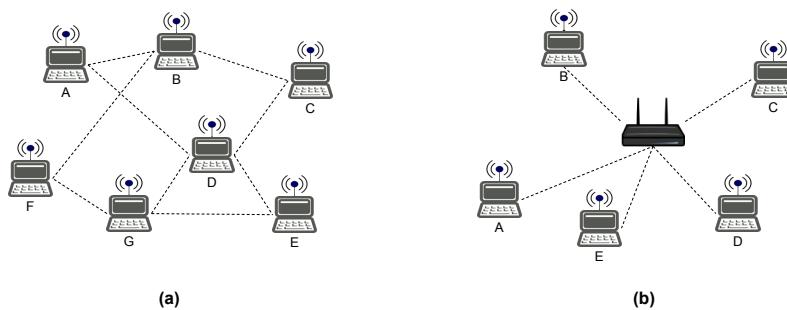


Figura 14.17: Modo ad hoc (a) e modo infra-estrutura (b). Neste último um ponto de acesso serve como intermediário das comunicações entre as diversas estações

Dois ou mais APs podem também ser interligados por *switches Ethernet* que asseguram a comunicação entre os terminais ligados a diferentes APs, assim como eventualmente com outras redes, ver a Figura 14.18.

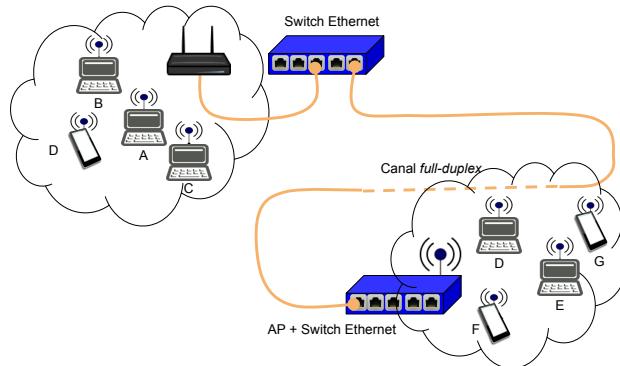


Figura 14.18: Sistema com vários APs interligados por uma rede baseada em *switches Ethernet*. Os APs funcionam também como *switches* e um dos *switches* é simultaneamente também um AP.

Na comunicação em modo ad hoc, quando a origem e os destino têm de passar por terminais intermédios, os diferentes terminais têm de funcionar como comutadores de pacotes. Por exemplo, na Figura 14.17, parte (a), o terminal A para enviar frames a E terá de passar por D. Sem essa hipótese, o âmbito nesse modo seria muito mais

restrito. Este tipo de redes, que também se costumam chamar redes sem fios em malha (*mesh*), exigem funcionalidades especiais de encaminhamento nos terminais e só são usadas quando o modo infra-estrutura, baseado em APs, não pode ser usado (*e.g.*, redes cooperativas, em zonas rurais, em zonas de desastre, *etc.*).

O modo infra-estrutura, com APs, é o mais popular e fácil de montar pois não exige nenhuma funcionalidade ou parametrização adicional dos terminais. Para que diversos APs possam coexistir no mesmo espaço físico, cada AP e os terminais que lhe estão associados pertencem a um **SSID - Service Set Identifier** ou identificador de rede diferente.

O AP difunde periodicamente o seu SSID, o seu endereço MAC e diversas características do canal através de *frames* especiais chamados **beacons frames**. São os diferentes SSIDs captados que os terminais mostram aos utilizadores quando assinalam que há várias redes Wi-Fi disponíveis.

O terminal para poder comunicar através de um canal identificado por um SSID, tem de se associar ao AP do mesmo através de um protocolo especial de associação. Quando a rede é fechada, é também executado um protocolo de autenticação e de estabelecimento de chaves criptográficas para tornar as comunicações seguras. Se um terminal detecta vários APs do mesmo SSID, escolhe automaticamente associar-se àquele com melhor qualidade de sinal. Se a qualidade do sinal se alterar (*e.g.*, por o terminal se deslocar), este pode decidir mudar dinamicamente o AP a que está associado.

O AP também informa periodicamente os terminais que lhe estão associados sobre qual a melhor potência de emissão que devem adoptar. Adicionalmente, os APs actuais escolhem automaticamente a gama de frequências em que operam de forma a diminuir a probabilidade de colisão com outros APs na proximidade. Os terminais, antes de se associarem, pesquisam os **beacons frames** presentes em diferentes frequências até se ligarem ao AP seleccionado.

Numa configuração baseada em infra-estrutura, os APs assumem diversos papéis. Assim, apesar de alguns terminais conseguirem comunicar directamente uns com os outros, um AP estende mais facilmente âmbito do canal. O AP também permite aos terminais libertarem-se mais facilmente dos *frames* que querem emitir e podem por isso poupar mais energia. Esta poupança é vital pois os terminais são geralmente móveis e alimentados a bateria. Quando se usam diversos APs distintos no mesmo espaço físico, é possível fazer coexistir mais facilmente vários canais distintos. Finalmente, os APs desempenham um papel importante na segurança do canal.

Na ausência dos APs os terminais têm de funcionar como comutadores de pacotes e fazerem encaminhamento de pacotes. Para além disso, têm de usar protocolos mais complexos para garantirem a coordenação das frequências, a autenticação e a segurança das comunicações. A faceta semi-centralizada do modo infra-estrutura apresenta diversas vantagens e é geralmente a preferida sempre que é possível utilizar este modo.

### Funcionamento do protocolo MAC

Um terminal que pretende transmitir um *frame* começa por esperar durante um período designado IFS (**IFS - Inter-Frame Space**) seguido de um *random backoff period* inicial (excepto se o terminal não usou recentemente o canal e este está livre durante todo o IFS). Quando esse período se esgota e o meio está livre, inicia a transmissão até ao fim do *frame*. Em seguida espera pelo ACK e caso não o receba, faz nova tentativa após novo IFS seguido de um novo *random backoff period*. Por omissão, *i.e.*, sem prioridades, ver a seguir, o valor do IFS é igual a DIFS (**DIFS - Distributed Inter-Frame Space**). De forma mais sistemática:

1. Caso o terminal não tenha tido recentemente actividade e o meio está livre pelo menos durante DIFS, passar imediatamente à etapa 3.

2. Calcular um *random backoff period* multiplicando por um número aleatório o período *slot time* e esperar que o contador venha a 0. O contador só é decrementado quando o meio está livre e depois de passar DIFS a seguir ao fim da última ocupação por um outro *frame* de um competidor.
3. Emitir o *frame* sem parar até ao fim (visto que não consegue detectar colisões).
4. Esperar pelo ACK.
5. Caso receba o ACK no período previsto, a transmissão foi concluída com sucesso. Senão recomeça o processo na etapa 2, mas usando um compasso de espera calculado de forma aleatória num intervalo multiplicado por dois (*binary exponential backoff*). O número máximo de repetições é variável mas está normalmente fixado em 4.

Uma das maneiras de introduzir prioridades no acesso ao meio consiste em usar diferentes compassos de espera, *i.e.*, diferentes valores de IFS. Assim, o receptor, se recebeu com êxito o *frame*, espera o intervalo de tempo SIFS (**SIFS** - *Short Inter-Frame Space*), menor que DIFS, e transmite imediatamente o ACK. Esta forma de proceder, ilustrada na Figura 14.19, assegura que um ACK é sempre transmitido antes de qualquer outro *frame* a menos de problemas relacionados com um “terminal escondido”.

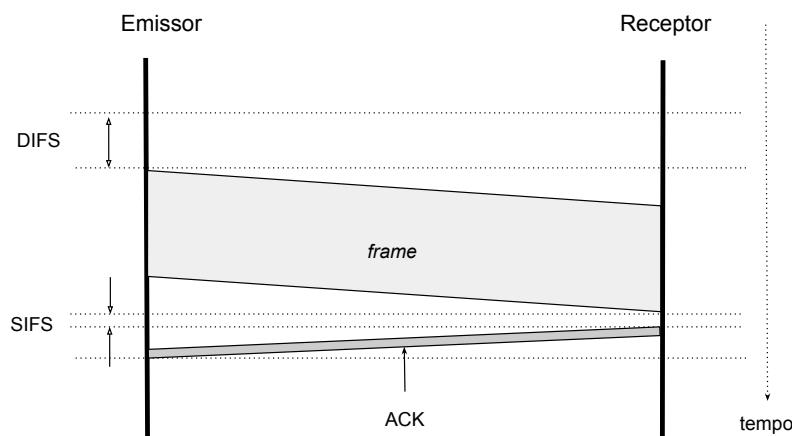


Figura 14.19: Funcionamento do MAC IEEE 802.11 e os respectivos compassos de espera DIFS e SIFS

O MAC IEEE 802.11 introduz um mecanismo suplementar que se chama **NAV** (*Network Allocation Vector*). Este mecanismo comprehende dois aspectos. O primeiro consiste em introduzir em todos os *frames* de dados um campo com a indicação do tempo necessário para a sua transmissão e recepção do respectivo ACK. Este campo está no cabeçalho e permite a qualquer outro terminal saber quanto tempo o canal vai estar ocupado por aquela transmissão concorrente. Se o *frame* não lhe for destinado, o terminal “pode dormir” (desligar a interface e o processamento de sinal do rádio) até ao fim do período NAV.

Adicionalmente, o NAV pode ser complementado por um mecanismo cujo efeito é a tentativa de reserva do canal durante esse período. O mecanismo funciona de tal forma que diminui a probabilidade de colisões em casos de “terminais escondidos”. Quando um terminal pretende enviar um *frame* extenso, emite preliminarmente um

*frame* especial designado RTS (*Request To Send*) em que especifica o NAV de que necessita, isto é o tempo para transmitir o *frame* e receber o seu ACK. O terminal de destino do futuro *frame*, se receber o RTS, deve responder com um *frame* CTS (*Clear To Send*) que inclui também um NAV com o tempo que levará a transmissão solicitada a terminar e a ser *acked*.

Os *frames* RTS e CTS são muito pequenos e portanto a probabilidade de provocarem colisões é menor. No entanto, caso tenham sido trocados com êxito, avisam todos os terminais que os receberem que o meio vai estar ocupado pelo emissor, pois este acabou, com a anuência do receptor, de reservar o canal. Este tipo de mecanismo só deve ser usado com *frames* a partir de uma dimensão significativa, pois com *frames* muito pequenos o *overhead* extra que introduz pode não compensar os ganhos obtidos. A dimensão do menor *frame* cuja transmissão já requer a utilização do mecanismo RTS / CTS é um parâmetro com um valor por omissão de poucas centenas de bytes (e.g., 400 bytes). A Figura 14.20 ilustra o funcionamento do mecanismo.

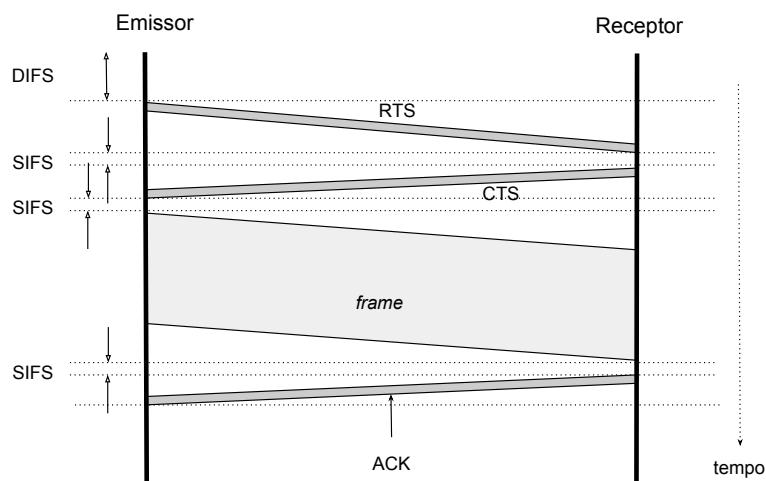


Figura 14.20: Funcionamento do MAC IEEE 802.11 utilizando o mecanismo RTS / CTS

É fácil reconhecer que num quadro de “terminal escondido”, como o da Figura 14.15, este mecanismo ajuda a resolver o problema. Com efeito, se o terminal A enviar um RTS a C, e C responder com um CTS, B não recebe o RTS mas recebe o CTS. Infelizmente, o mesmo não se passa no quadro de um “terminal exposto”. Devido ao *overhead* suplementar e a não ser uma “panaceia para todos os males”, o mecanismo só é usado para *frames* maiores que uma certa dimensão mínima, fixada como um parâmetro do terminal.

O funcionamento distribuído que acabámos de descrever designa-se por modo **DCF** - (*Distributed Coordination Function*). O MAC IEEE 802.11 introduziu também um modo de funcionamento, designado **PCF** - (*Point Coordination Function*), em que o AP funciona como árbitro centralizado da afectação do canal. No entanto, este modo não é utilizado pois é incapaz de funcionar correctamente quando vários APs interferem uns com os outros, o que mostra que a centralização tem os seus limites quando o coordenador não é único.

As normas IEEE 802.11a/b/g incluem vários mecanismos suplementares. Um deles é o da redução automática do débito para compensar uma situação em que o canal tem

muitos erros, ou em que um terminal está muito afastado do AP. Assim, ao invés de transmitir a 54 Mbps (característicos da versão 802.11g), o terminal pode transmitir a 1, 2, 5,5 ou 11 Mbps (característicos da versão 802.11b). A versão 802.11a difere das outras duas por usar a gama de frequências na banda dos 5 GHz e não pode coexistir com as mesmas.

Curiosamente, com adaptação do débito, se cada terminal só poder enviar um *frame* de cada vez, o terminal de maior débito passa a poder usar apenas uma fração minúscula da capacidade do canal. Com efeito, o tempo de transmissão de *frames* a baixo débito ocupa muito mais tempo o canal. Por exemplo, se dois terminais emitirem *frames* de 1 Kbyte, um a 1 Mbps ( $T_t \approx 8\text{ ms}$ ) e outro a 54 Mbps ( $T_t \approx 150\text{ }\mu\text{s}$ ), o débito médio do terminal rápido (excluindo *overheads*) é pouco superior a 1 Mbps também. Por este motivo a norma evoluiu para permitir aos terminais mais rápidos enviarem mais do que um *frame* de cada vez para dividir mais equitativamente o tempo entre os diferentes competidores. Este aspecto será de novo retomado a seguir.

Existem ainda outros mecanismos complementares entre os quais a possibilidade de o AP memorizar durante algum tempo os *frames* que tem para enviar para os diferentes terminais, e avisá-los quando envia os *beacon frames* (que são transmitidos de 100 em 100 ms) se tem *frames* para lhes enviar ou não. Isso permite aos terminais “dormirem” até à emissão do próximo *beacon* caso não tenham tráfego para enviar ou receber no momento.

Em 2005 foi também introduzida uma extensão, designada IEEE 802.11e, que permite utilizar um sistema de prioridades. Esse mecanismo é implementado usando um conjunto alargado de intervalos IFS, que um terminal tem de esperar antes de poder transmitir quando o meio fica livre. Esse conjunto de intervalos de tempo alargado é apresentado na Tabela 14.2

Tabela 14.2: Intervalos de separação entre *frames* da norma IEEE 802.11e, listados por ordem crescente de duração.

Sigla	Nome	Antecede um
SIFS	Short Inter-frame Space	<i>frame</i> de controlo
AIFS1	Arbitration Inter-frame Space 1	<i>frame</i> de alta prioridade
DIFS	Distributed Inter-frame Space	<i>frame</i> de prioridade normal
AIFS4	Arbitration Inter-frame Space 4	<i>frame</i> de baixa prioridade
EIFS	Extended Inter-frame Space	<i>frame</i> emitido após um recebido com erros e NAV distorcido

### Formato dos *frames* IEEE 802.11a/b/g

O formato dos *frames* é o apresentado na Figura 14.21. Olhando para o mesmo é fácil de identificar algumas diferenças com respeito aos *frames* no formato IEEE 802.3. Primeiro, o protocolo MAC é mais complexo, e por isso existe um primeiro campo com informação de controlo que permite indicar se o *frame* é de controlo (ACK, RTS, CTS, *beacon*, *association request*, *association response*, etc.) ou simplesmente um *frame* de dados. A este campo segue-se o campo **Duração** que contém a informação NAV, isto é a duração em micro segundos em que emissor vai (ou pretende no caso de um RTS) ocupar o canal.

O campo dos **Dados** (o *payload*) contém os dados propriamente ditos. Apesar de a norma admitir que os *frames* possam ser maiores, o limite imposto pela Ethernet (1500 bytes) é o dominante. Este campo inclui também alguma informação referente ao tipo de informação transportada (*pacote IP*, *outros pacotes*, etc.). A seguir ao campo de Dados vem o campo **CRC**, idêntico ao usado na norma IEEE 802.3.

O campo **Sequência** é necessário para realizar de forma completa o protocolo *stop & wait*. Com efeito, se o ACK se perder, correr-se-ia o risco de introduzir duplicados. Este campo serve para detectar os mesmos e para tornar fiável o protocolo com várias transmissões.

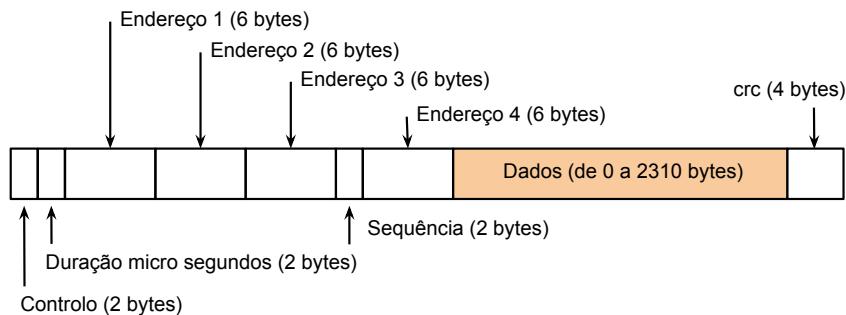


Figura 14.21: Formato dos *frames* IEEE 802.11

A parte menos comum do formato do *frame* tem a ver com a utilização de quatro endereços canal. Um primeiro aspecto a ter presente é que todos os equipamentos ligados a um canal IEEE 802.11 têm de ter endereços MAC, quer os terminais, quer os APs. De facto, só desta forma um AP, ou um terminal, pode decidir, comparando com o seu endereço, se o *frame* lhe é destinado ou não. Os endereços são em tudo compatíveis e similares aos endereços IEEE 802.3.

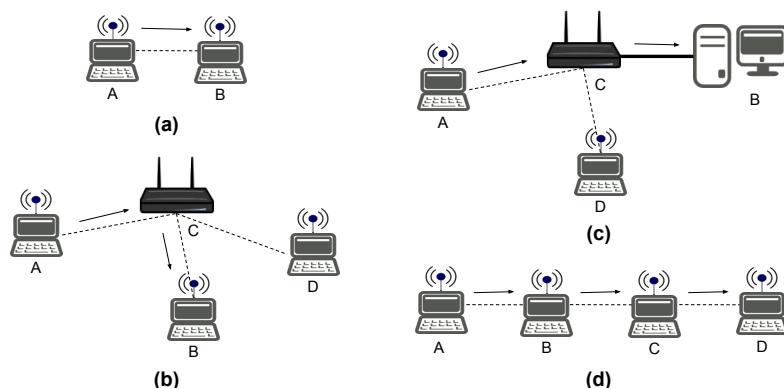


Figura 14.22: Vários cenários de comunicação entre terminais que ilustram a necessidade de usar mais do que dois endereços no cabeçalho dos *frames* IEEE 802.11 (situações (b), (c) e (d))

Quando duas estações estão directamente ligadas ao mesmo meio em modo ad hoc, só são necessários os endereços origem e destino do *frame*, como é normal. Ver a Figura 14.22 (a). No entanto, quando dois terminais estão ligados através de um AP, o endereço origem é o do emissor, o endereço de destino é o do AP, e o endereço de destino final (um terceiro endereço) é o do outro terminal. Ver a Figura 14.22 (b). A

mesma situação tem lugar se o destino final é um outro equipamento ligado ao AP. Ver a Figura 14.22 (c). Finalmente, em modo ad hoc, como é ilustrado na Figura 14.22 (d), um *frame* enviado de A para D através de B e C necessita de 4 endereços quando transita entre B e C: os origem e destino final e os necessários para transitar de B para C.

Assim, o **Endereço 1** corresponde ao receptor real do *frame*, o **Endereço 2** corresponde ao emissor real do *frame*, o **Endereço 3** corresponde ao receptor final do *frame*, e o **Endereço 4** corresponde ao emissor original do *frame*. Logo, no exemplo (d) acima, quando o *frame* transita de B para C, **Endereço 1 = C**, **Endereço 2 = B**, **Endereço 3 = D** e **Endereço 4 = A**.

Os *frames* IEEE 802.11b/g numa rede que use as duas versões simultaneamente têm um preâmbulo e informação de controlo comuns, codificados sempre com o débito de 1 Mbps, que permite sincronizar o emissor e o receptor, e tem informação que permite interpretar o resto do *frame* (o qual é codificado com o débito próprio do emissor). Este preâmbulo tem a dimensão fixa de 192 bits e a duração fixa de  $192 \mu s$ , independentemente do débito do canal. No funcionamento em modo infraestrutura, o AP assegura a adaptação do débito do emissor ao usado pelo receptor. Na versão IEEE 802.11g e nas versões mais recentes da norma (IEEE 802.11n e ac), este preâmbulo é substituído por versões mais curtas quando não se pretende garantir a interoperabilidade com as versões anteriores.

Tabela 14.3: Valores de alguns parâmetros das normas IEEE 802.11b/g com garantia de coexistência entre estações a usarem diferentes versões do protocolo MAC.

Parâmetro	Valor / Descrição
Preâmbulo + CTL	192 $\mu$ segundos, 192 bits transmitidos a 1 Mbps
<i>Slot time</i>	20 $\mu$ segundos
SIFS	10 $\mu$ segundos
DIFS	50 $\mu$ segundos ( <i>SIFS</i> + $2 \times$ <i>Slot time</i> )
EIFS	364 $\mu$ segundos
<i>CW</i> <sub>min</sub>	15 <i>Slot times</i> (minimum contention window size)
<i>CW</i> <sub>max</sub>	1024 <i>Slot times</i> (maximum contention window size)
Cabeçalho MAC	28 bytes
ACK	14 bytes
RTS	20 bytes
CTS	14 bytes

A Tabela 14.3 apresenta alguns valores característicos das normas IEEE 802.11b/g com garantia de interoperabilidade. Recorde-se que o *backoff period* é calculado como sendo *Random(CW)* onde CW representa a *Congestion Window*. Quando se pretende a coexistência no mesmo canal das diferentes versões, uma parte dos valores da tabela acima é fixo e independente do débito em que o emissor está a funcionar. O preâmbulo do *frame*, acrescentado do DIFS e de um *backoff period* de por exemplo 8 *slots*, totaliza  $192 + 50 + 8 \times 20 \approx 400\mu s$ . Se o emissor estivesse a funcionar a 54 Mbps, 1000 bytes seriam transmitidos em  $8000/54.10^6 \approx 150 \mu s$  o que dá a dimensão do *overhead* potencial (e.g., superior a 200%) introduzido pelo protocolo MAC sobretudo se o terminal fosse obrigado a emitir um só *frame* de cada vez. Claro que se o emissor emitir com o débito de 1 Mbps, os 1000 bytes são transmitidos em 8 ms, e nesse caso o *overhead* não ultrapassa 6%.

Para compensar o *overhead* significativo a débitos mais elevados, é possível ao emissor transmitir vários *frames* seguidos desde que tenha obtido o acesso ao meio

de comunicação. Esta opção é designada por TXOP (*Transmit Opportunity*) e foi introduzida na extensão 802.11e aos MAC 802.11b/g. A mesma suporta igualmente um mecanismo de ACK conjunto dos *frames*. O valor mínimo de CW também pode ser 7 ou 3 para os *frames* de mais alta prioridade e o mecanismo TXOP é definido de tal forma que o emissor pode ocupar o canal durante 1,5 ou 3 milissegundos seguidos.

Para aumentar o rendimento da rede, sempre que possível, é preferível fixar o funcionamento exclusivo na norma 802.11g, a de maior desempenho. Nesse caso serão usados cabeçalhos mais curtos, que só são legíveis pelas estações a usarem o débito de 54 Mbps. O mesmo se aplica com as versões mais actuais, nomeadamente as 802.11.n e 802.11.ac.

As normas IEEE 802.11 também suportam a emissão de *frames* em *broadcast*, i.e., dirigidos a todos os terminais. Estes *frames* são caracterizados por serem dirigidos ao endereço especial FF:FF:FF:FF:FF:FF e por não existir ACK de resposta, nem ser possível usar o mecanismo RTS/CTS. Por outro lado, em modo compatível com várias versões, são todos emitidos com o débito de 1 Mbps para que todas as estações os possam receber.

### Evolução da norma IEEE 802.11

Em 2009 foi introduzida uma nova evolução da norma IEEE 802.11, a versão IEEE 802.11n, também conhecida por MIMO Wi-Fi (*Multiple Input Multiple Output Wi-Fi*), ver a Secção 14.4. Uma das grandes alterações introduzidas consistiu na melhoria do nível físico que passou a poder utilizar vários feixes de ondas em simultâneo, codificações mais sofisticadas, maiores gamas de frequências por canal, assim como várias antenas em conjunto, o que desceu significativamente a taxa de erros e aumentou o débito. Com uma só antena, este sistema pode transmitir com o débito de 70 Mbps. No entanto, usando 4 antenas em cada terminal, canais ocupando uma maior gama de frequências, e se não houver competidores, é possível chegar a 600 Mbps. Este desempenho é, na prática, impossível num ambiente urbano engarrafado por diferentes redes Wi-Fi.

O simples aumento do débito não é suficiente para aumentar significativamente o débito útil real do canal pois o *overhead* do protocolo MAC pode ser significativo como vimos acima. Por esta razão, a norma introduz igualmente mecanismos TXOP, de *frame aggregation*, que permitem enviar vários *frames* de uma só vez, assim como um mecanismo de *block acknowledgement* que permite transmitir de uma só vez ACKs referentes a diferentes *frames*. A dimensão do *frame* máximo aumentou para 8000 bytes, transmitidos em vários fragmentos, e podem ser transmitidos até 64 Kbytes de uma só vez. No entanto, aproveitar todas estas vantagens no nível transporte é um desafio pois a interface tem de ter toda a informação disponível para a transmitir de uma só vez.

A coexistência com terminais a funcionarem nas versões anteriores (e.g., b, g) é possível, mas coloca problemas de desempenho. Por essa razão, o débito máximo teórico só é utilizável num modo em que só existam terminais 802.11/n, a utilizarem um cabeçalho específico deste modo, e a competirem pelo meio em exclusividade.

Em 2014 foi introduzida a versão IEEE 802.11ac, que utiliza a banda entre os 5 e os 6 GHz, melhores técnicas de codificação, mais antenas e mais feixes de ondas. Em condições teóricas óptimas esta versão pode atingir o débito de transmissão de 1,4 Gbps. Para os próximos anos estão previstas novas versões da norma com a forma geral IEEE 802.11a x. Estas utilizarão outras bandas de frequências IMS e terão âmbitos, técnicas de codificação e débitos diferentes.

Os canais sem fios são muito atractivos por serem cónmodos de instalar e suportarem directamente sistemas móveis. Por estas razões tem sido realizado um grande investimento em canais sem fios baseados em difusão, *half-duplex*, de grande débito, fáceis de instalar e pouco dispendiosos. Os canais da família de **normas IEEE 802.11**, também conhecidos por **canais Wi-Fi**, satisfazem estes requisitos, utilizam **bandas de frequências livres (IMS)** e, devido à baixa potência de emissão, têm o âmbito de algumas dezenas de metros em meio urbano.

Devido às características do sinal e do meio de propagação que utilizam, estes canais utilizam um **protocolo MAC do tipo CSMA/CA, ACKs dos frames, retransmissão** em caso de falha para compensar os erros, e mecanismos complementares de **alocação e reserva do canal (e.g., NAV e RTS/CTS)**.

Tem havido um grande progresso no débito nominal permitido pelas interfaces IEEE 802.11. No entanto, mesmo depois da introdução de inúmeros melhoramentos e optimizações no protocolo MAC, este continua a **apresentar um overhead significativo**. Adicionalmente, dado que existe uma grande **competição em meio urbano pela utilização das frequências disponíveis**, o débito real extremo a extremo é frequentemente uma fração relativamente pequena do débito nominal teórico.

Uma faceta relevante destes canais é o facto de utilizarem endereços MAC compatíveis com as normas IEEE 802, e ser relativamente fácil integrar vários canais sem fios IEEE 802.11 numa rede a funcionar com *switches* Ethernet como veremos no próximo capítulo. Este facto tem contribuído de forma significativa para o seu sucesso.

## 14.4 Resumo e referências

### Resumo

Os canais baseados em difusão funcionam necessariamente no modo *half-duplex* pois se várias interfaces emitem simultaneamente, sobre um mesmo meio de comunicação partilhado, dá-se uma **colisão** dos sinais que provoca a recepção de *frames* com erros e que por isso ficam inutilizados e têm de ser postos de lado, *i.e.*, ignorados. Para resolver este problema é necessário um mecanismo de **controlo de acesso ao meio**, que é abreviado pela sigla em língua inglesa **MAC – Medium Access Control**.

Existem muitas tecnologias de canais *half-duplex* baseados em difusão que utilizam vários tipos de meios de comunicação como a atmosfera ou meios guiados (cabos de cobre ou de fibra óptica). Nos canais sem fios, ao contrário dos canais baseados em meios guiados, a probabilidade de erros é muito elevada, e pode não ser possível um emissor detectar se o canal está ou não ocupado ou se houve ou não colisão.

Estes protocolos são vários e utilizam diferentes tipos de mecanismos, nomeadamente: **frames de ACK** quando a probabilidade de erro e de colisões é mais elevada, compassos de espera aleatórios (**Random Backoff Periods**) em caso de deteção de colisões ou antes de transmitirem um *frame* pela primeira vez, sondagem do meio de comunicação para verificar se este está livre ou ocupado (**CS - Carrier Sense**), e mecanismos de deteção de colisões (**CD - Collision Detection**). Uma associação específica de diferentes mecanismos é usada em função do meio de comunicação e de outras características necessárias em diferentes contextos. Os protocolos MAC foram normalizados pela IEEE, na família de protocolos com o prefixo **IEEE 802**.

As normas IEEE 802.3 destinam-se a canais guiados em que é possível detectar as colisões, a fiabilidade da transmissão é elevada quando não existem colisões, e é possível reservar o meio através de um pequeno *frame* desde que a distância máxima entre interfaces seja relativamente pequena. Este conjunto de razões permitem a utilização de um protocolo MAC do tipo CSMA/CD sem utilização de ACKs.

Uma outra faceta interessante da implementação na Ethernet do protocolo MAC CSMA/CD consistiu na adopção do método ***binary exponential backoff*** que adapta a duração dos compassos de espera aleatórios em função da maior ou menor probabilidade de colisão.

Modernamente, esta família de normas abrange um conjunto de tecnologias que comportam funcionamento em *half-* e *full-duplex*. Neste último caso, não existe necessidade de utilizar um protocolo MAC, nem limitações de distância, porque o canal funciona em modo ponto-a-ponto entre apenas duas interfaces.

Os canais sem fios são muito atractivos por serem cónmodos de instalar e suportarem directamente sistemas móveis. Por estas razões tem sido realizado um grande investimento em canais sem fios baseados em difusão, *half-duplex*, de grande débito, fáceis de instalar e pouco dispendiosos. Os canais da família de **normas IEEE 802.11**, também conhecidos por canais Wi-Fi, satisfazem estes requisitos, utilizam **bandas de frequências livres (IMS)** e, apesar da baixa potência de emissão, têm o âmbito de algumas dezenas de metros em meio urbano.

Devido às características do sinal e do meio de propagação que utilizam, estes canais utilizam um **protocolo MAC do tipo CSMA/CA, ACKs dos frames, retransmissão** em caso de falha para compensar os erros, e mecanismos complementares de **reserva do canal (e.g., NAV e RTS/CTS)**.

Tem havido um grande progresso no débito nominal permitido pelas interfaces IEEE 802.11. No entanto, mesmo depois da introdução de inúmeros melhoramentos e optimizações no protocolo MAC, este continua a apresentar um *overhead* significativo. Adicionalmente, dado que existe uma grande competição em meio urbano pela utilização das frequências disponíveis, o débito real extremo a extremo é frequentemente uma fração relativamente pequena do débito nominal teórico.

Uma faceta relevante destes canais é o facto de utilizarem endereços MAC compatíveis com os das normas IEEE 802.3, e ser relativamente fácil integrar vários canais sem fios IEEE 802.11 numa rede a funcionar com *switches* Ethernet como veremos no próximo capítulo. Este facto tem contribuído de forma significativa para o seu sucesso.

Alguns dos termos introduzidos no capítulo são a seguir passados em revista. Entre parêntesis figura a tradução mais comum em língua inglesa quando isso se justifica.

### Principais conceitos

**Acesso Múltiplo (MA - Multiple Access)** Canal baseado em difusão com meio de comunicação de acesso múltiplo.

**Colisão (collision)** Num canal baseado em difusão se dois ou mais emissores emitem sinal simultaneamente, os respectivos sinais entram em colisão e não é possível a um receptor destrinçar e obter os *frames* emitidos pelos emissores.

**Controlo de acesso ao meio (MAC – Medium Access Control)** Protocolo que permite coordenar os diferentes emissores ligados ao mesmo canal de forma a evitar colisões.

**Frames de ACK** *Frames* enviadas pelo receptor ao emissor num canal baseado em difusão para permitir correção de erros por retransmissão ao nível do canal. Estes *frames* permitem implementar um protocolo *stop & wait* ao nível canal.

**Compassos de espera aleatórios (Random Backoff Periods)** Períodos de espera aleatórios usados pelos diferentes emissores antes de emitirem, para tentarem decidir aleatoriamente qual deles tem acesso ao meio de comunicação em primeiro lugar.

**Janela de contenção** (*CW - congestion window*) O compasso de espera de cada emissor é dado por:  $\text{Random}(0..CW)$ .

**Janel de contenção crescente exponencial binária** (*binary exponential backoff*) Janelas de contenção exponencialmente crescentes, sendo multiplicadas por 2 na sequência de tentativas anteriores de resolução das colisões falhadas.

**Detecção da portadora** (CS - *Carrier Sense*) Mecanismo que permite detectar no meio de comunicação o sinal de outro emissor.

**Detecção da colisão** (CD - *Collision Detection*) Mecanismo que permite ao emissor detectar que a sua emissão colidiu com a de outro emissor.

**Evitar a colisão** (CA - *Collision Avoidance*) Mecanismo baseado em compassos de espera aleatórios, antes da primeira tentativa de emissão, para diminuir a probabilidade de que dois ou mais emissores entrem em colisão.

**CSMA/CD** Protocolo MAC que se baseia na detecção da portadora e de colisões.

**CSMA/CA** Protocolo MAC que se baseia na detecção da portadora e de tentar evitar colisões pois as estações uma vez iniciada a emissão não conseguem detectar colisões.

**Bandas de frequências livres (IMS)** (*Industrial, Medical, Scientific bands*) Gamas de frequências públicas não sujeitas a licenciamento prévio. Por essa razão podem ser utilizadas simultaneamente por diferentes canais, os quais podem entrar em competição pelo acesso ao meio de comunicação.

## Referências

A apresentação dos protocolos MAC nesta capítulo não inclui nenhuma tentativa de caracterizar de forma analítica o desempenho dos mesmos. Uma versão clássica deste tratamento, baseada num modelo simplificado do comportamento dos emissores e do canal, é apresentada no Capítulo 1 de [Marsic, 2013]. Esses tratamentos permitem comparar diferentes MACs e ordená-los por eficiência (menor *overhead*). No entanto, na prática, revelam-se demasiado optimistas pois o comportamento real do meio de comunicação e dos terminais são inadequadamente modelizados.

A maioria dos livros sobre redes de computadores contemplam um capítulo sobre o nível físico, onde os canais multi-ponto baseados em difusão, algumas vezes apresentados com o sub-título Redes Locais (LAN - *Local Area Networks*), são tratados dando um especial realce à problemática do controlo do acesso ao meio de comunicação. Por exemplo, [Tanenbaum and Wetherall, 2011] no Capítulo 4, [Kurose and Ross, 2013] nos Capítulos 5 e 6, [Peterson and Davies, 2012] no Capítulo 2, *etc.*

O artigo [Metcalfe and Boggs, 1976] apresenta a rede Ethernet. As normas IEEE 802.3 e IEEE 802.11 estão disponíveis no *site* da IEEE. No entanto, existem excelentes livros exclusivamente dedicados ao tema da Ethernet, *e.g.*, [Spurgeon, 2000; Held, 2003], assim como ao tema das redes sem fios em geral, incluindo as normas IEEE 802.11 em particular, como por exemplo [Stallings, 2011]. O livro [Beard and Stallings, 2015] cobre as versões mais recentes dessas normas.

Para o leitor interessado numa primeira abordagem das técnicas de transmissão MIMO (*Multiple Input Multiple Output*), o artigo [Halperin et al., 2010] é um bom ponto de partida.

## 14.5 Questões para revisão e estudo

1. Verdade ou mentira?

- (a) Num canal ponto-a-ponto 802.3 e *full-duplex*, se o emissor quiser emitir um *frame* pode sempre fazê-lo.

- (b) Num canal partilhado 802.3 baseado em difusão e *half-duplex*, se o emissor quiser emitir um *frame* pode sempre fazê-lo.
  - (c) Num canal 802.11, a funcionar em modo distribuído, as colisões só podem ocorrer durante um período de tempo limitado designado *collision window*.
  - (d) Num canal partilhado 802.3, baseado em difusão e *half-duplex*, o emissor sabe que as colisões só podem ocorrer durante um período de tempo limitado dependente do tempo de propagação de extremo a extremo. Por essa razão, a interface começa geralmente por medir o RTT do canal.
  - (e) Num canal 802.11, a funcionar em modo distribuído, um emissor de um *frame* tem a certeza que o mesmo foi bem recebido sempre que recebe um ACK do receptor.
  - (f) Num canal partilhado 802.3 baseado em difusão e *half-duplex*, o emissor de um *frame*, na ausência de colisão, tem sempre a certeza de que o receptor o recebeu bem.
2. Quais dos seguintes protocolos MAC usam *timeouts* e retransmissão de *frames*: ALOHA, Slotted ALOHA, CSMA, CSMA/CA 802.11 e CSMA/CD 802.3 em modo *half-duplex*?
3. Qual seria a repercussão sobre o protocolo TCP se o MAC 802.11 deixar de fazer qualquer retransmissão de *frames*?
4. Verdade ou mentira?
- (a) Os endereços MAC são hierárquicos e reflectem informação sobre a topologia da rede, i.e., informação sobre a região da rede em que a interface com o endereço se situa.
  - (b) Os endereços MAC são hierárquicos e reflectem o fabricante da interface.
  - (c) Os endereços MAC estão registados no DNS para que seja possível aos clientes encontrarem os endereços dos servidores.
  - (d) À saída da fábrica os endereços MAC são únicos no mundo inteiro.
  - (e) O cabeçalho dos *frames* nos canais multi-ponto não necessitam de conter endereços destino ou origem dos *frames*.
  - (f) Num canal IEEE 802.11 os *frames* só necessitam de dois endereços: destino e origem.
  - (g) Num canal IEEE 802.3 os *frames* só necessitam de dois endereços: destino e origem.
5. Um protocolo MAC com mecanismos aleatórios realiza a multiplexagem do canal entre  $n$  interfaces distintas. Como se pode caracterizar essa multiplexagem: TDM (*Time Division Multiplexing*), FDM (*Frequency Division Multiplexing*) ou multiplexagem estatística?
6. Um protocolo MAC do tipo CSMA/CA usado num canal sem fios apresenta um *overhead* superior ao de um do tipo CSMA/CD usado num canal com fios, ambos com o mesmo débito de transmissão. Quais dos seguintes factores contribuem mais significativamente para essa diferença?
- (a) O mecanismo de CS (*Carrier Sense*).
  - (b) Os mecanismos de reserva do canal sem fios (NAV, RTS, CTS).
  - (c) Os endereços nos *frames*.
  - (d) O protocolo de ACK dos *frames*.
  - (e) O algoritmo *binary exponential backoff*.

- (f) O tempo de transmissão dos *frames*.  
(g) O mecanismo de arbitragem do acesso inicial ao meio de comunicação.
7. Um conjunto de computadores com interfaces IEEE 802.3 estão interligados através de um equipamento central numa configuração em estrela. Algumas das interfaces têm o débito de 10 Mbps e outras de 100 Mbps, mas todas comunicam perfeitamente. O equipamento central é um *hub* ou um *switch* Ethernet? Justifique.
  8. Duas estações *a* e *b* estão ligadas via um AP numa rede IEEE 802.11 a funcionarem ambas com o débito de 11 Mbps e numa situação em que tem de ser assegurada a compatibilidade com estações a funcionarem com débitos mais baixos (*i.e.*, 802.11b/g em modo compatível). A estação *a* envia um *frame* com 11.000 bits à estação *b* usando o mecanismo RTS/CTS. Qual o menor tempo necessário para que o *frame* chegue ao destino admitindo que em todas as comunicações realizadas não houve contenção no acesso ao meio de comunicação (colisões / repetições). Calcule igualmente o débito extremo a extremo conseguido nessa situação optimista. Para efeitos dos cálculos utilize os dados fornecidos pela Tabela 14.3.
  9. Repita o problema anterior mas numa situação em que as duas estações estão ambas a funcionarem em modo exclusivo 802.11g. Neste caso todo o cabeçalho a restante parte do *frame* são transmitidos a 54 Mbps e, por hipótese, os tempos SIFS e DIFS tomam os valores 9 e 28  $\mu$  segundos.
  10. Duas estações *a* e *b* estão ligadas via um AP numa rede IEEE 802.11 a funcionarem ambas com o débito de 54 Mbps. Não existem nenhuma outras estações ligadas à mesma rede, nem outras redes na proximidade. Através do programa ping fizeram-se medidas do tempo de trânsito entre *a* e *b*, tendo-se verificado existirem constantemente variações do RTT. Liste todos os factores podem justificar essas variações sabendo que só *a* e *b* estão a usar este canal?
  11. Porque é que o mecanismo RTS/CTS do protocolo MAC IEEE 802.11 é de utilização opcional?
  12. Um conjunto de computadores com interfaces de rede IEEE 802.3 estão interligados através de um equipamento central numa configuração em estrela e o tempo de propagação é desprezável. O tempo de transmissão de um *frame* é  $T_t$ . Quanto tempo leva o *frame* a ser recebido pela interface de destino quando o equipamento central é um *hub* Ethernet e quando é um *switch* Ethernet? Justifique.
  13. O protocolo MAC do tipo CSMA/CA da norma IEEE 802.3 não tem a noção de diferentes prioridades dos *frames*. Indique uma forma de introduzir um esquema de prioridades no mesmo. Indique as vantagens e desvantagens da sua proposta.
  14. Porque razão os protocolos MAC do tipo CSMA/CD para canais com fios (*e.g.*, 802.3) não compensam os erros do canal usando ACKs e retransmissões, enquanto que os protocolos MAC do tipo CSMA/CA para canais sem fios (*e.g.*, 802.11) o fazem?
  15. Pretende-se utilizar o protocolo MAC CSMA/CD para controlar o acesso a um canal *half-duplex* baseado em difusão do tipo IEEE 802.3 com o débito de 10 Mbps e um tempo máximo de propagação de extremo a extremo de 50  $\mu$  segundos. Indique qual a dimensão em bits do menor *frame* Ethernet que tem de ser emitido nesse canal
  16. Pretende-se utilizar um protocolo MAC do tipo CSMA/CD num canal *half-duplex* do tipo Ethernet com o débito de 10 Mbps e um tempo de propagação de extremo a extremo de 75 micro segundos. Indique quantos bits uma estação

emissora tem de transmitir no mínimo para que o MAC CSMA/CD possa ser usado adequadamente. Comece por calcular a dimensão do *collision slot* para chegar à sua resposta.

17. É possível usar um protocolo MAC do tipo CSMA/CD num canal satélite sem fios com o débito de 1 Mbps e o tempo de propagação extremo a extremo de 250 milissegundos? Justifique.
18. Em geral o nível rede transmite por um canal um pacote de cada vez, fazendo corresponder cada pacote a um *frame* distinto. Existem situações em que seria preferível colocar mais do que um pacote no mesmo *frame*? No entanto, tendo em consideração o nível transporte, em que casos seria isso realista? Justifique as suas respostas.
19. Segundo a norma IEEE 802.3, num canal multi-ponto partilhado *half-duplex* com o débito de 10 Mbps, e com o comprimento máximo de 2500 metros, o *collision slot* tem a duração de  $51,2 \mu$  segundos, equivalente à transmissão de 512 bits. Pretende-se introduzir uma variante da norma para canais a funcionarem a 20 Mbps mas de tal forma que o *frame* mínimo continue a ter 512 bits. Com quais das formas a seguir indicadas é isso possível? Repare que pode haver mais do que uma ou várias equivalentes.
  - (a) Aumenta-se o comprimento máximo para 5.000 metros.
  - (b) Diminui-se o comprimento máximo para 100 metros.
  - (c) Diminui-se o comprimento máximo para 1.250 metros
  - (d) Muda-se o *collision slot* para ter a duração de  $25,6 \mu$  segundos
  - (e) Muda-se o *collision slot* para ter a duração de  $5,2 \mu$  segundos
20. Para interligar 12 computadores um repetidor (*hub*) foi substituído por um *switch* Ethernet com 12 portas com o mesmo débito das portas do *hub*. Internamente o *switch* Ethernet só consegue transferir um *frame* de cada vez entre cada duas portas. Associadas a cada uma das suas portas existem duas filas de espera, uma para os *frames* recebidos e outra para os *frames* a enviar. Indique se há ou não, potencialmente, alguma melhoria do débito extremo a extremo entre os computadores com esta substituição. Justifique a sua resposta.
21. Verdade ou mentira?
  - (a) O protocolo MAC IEEE 802.11 suporta o envio de *frames* em *broadcast*. O emissor recebe tantos ACKs quantos os receptores.
  - (b) Quando um emissor IEEE 802.11b/g (modo misto) envia um *frames* em *broadcast*, usa sempre na emissão o débito máximo da sua interface.
  - (c) Quando um emissor IEEE 802.11 envia *frames* de grande dimensão em *broadcast*, usa sempre o mecanismo RTS/CTS.
  - (d) A introdução de um mecanismo de NACK (*Negative ACK*) (e.g., 802.3) num protocolo MAC do tipo CSMA/CA permitiria ganhos de eficiência.

## ***Capítulo 15***

---

### ***Encaminhamento com base em inundação***

---

*Increasingly, people seem to misinterpret complexity as sophistication, which is baffling – the incomprehensible should cause suspicion rather than admiration.*

– Autor: *Niklaus Wirth*

No capítulo anterior vimos como fazer uma rede com um único canal multi-ponto baseado em difusão. Naturalmente, este tipo de rede tem necessariamente um âmbito limitado. No caso de redes Wi-Fi, este restringe-se a algumas dezenas de metros “sob telha” (*indoor*), um pouco mais ao ar livre dado existirem menos obstáculos à propagação. Neste capítulo continuaremos a ver soluções tão simples quanto possível para o encaminhamento de pacotes em redes de computadores, mas de forma a alargarmos o seu âmbito.

A solução terá de ser viável para uma rede arbitrária que interliga um conjunto de computadores através de canais e comutadores de pacotes. Alguns dos canais são ponto-a-ponto, enquanto que outros são multi-ponto. Essa rede deverá envolver caminhos alternativos e ter a configuração de uma malha arbitrária. Estes caminhos alternativos permitem que a rede continue a fornecer serviço mesmo quando alguns dos caminhos se tornam momentaneamente indisponíveis devido a avarias.

Os computadores comunicam através da troca de pacotes. Os pacotes têm um cabeçalho com pelo menos os endereços destino e origem. Os endereços dos computadores são todos diferentes uns dos outros e cada comutador tem ele próprio um endereço. Cada comutador que recebe um pacote consulta o endereço de destino, e caso este não o indique a ele próprio, decide o que fazer ao pacote e encaminha-o para o destino, geralmente via outro comutador.

Neste capítulo vamos explorar o mais simples dos métodos de encaminhamento de pacotes, que é conhecido pela designação de encaminhamento por inundação: quando um comutador recebe um pacote, se este não lhe é destinado, envia uma cópia do mesmo por todos os canais que conhece, excepto por aquele pelo qual o recebeu. Como é evidente, este método assegura que uma cópia do pacote chegue ao destino, mas eventualmente mais do que uma. Infelizmente também sobrecarrega a rede com um conjunto de cópias inúteis do mesmo pacote. Parece não ser uma muito boa ideia. No entanto, como veremos a seguir, o método de base pode ser complementado com um conjunto de mecanismos que o tornam mais realista e até de utilização comum.

A secção que se segue apresenta o método de encaminhamento com base em inundação e estuda os mecanismos necessários para tornar este método de encaminhamento realista e adequado em certos contextos e para algumas aplicações específicas. A seguir veremos como este método de encaminhamento pode ser usado pelos *switches* Ethernet para implementar uma rede simples de interfaces IEEE 802 (.3, .11, etc.) a comunicarem directamente. Finalmente, analisaremos como estas redes são utilizadas na prática hoje em dia, e como podem mascarar as falhas que ocorreram e adaptarem-se aos computadores que se ligam e desligam dinamicamente da rede.

### 15.1 Encaminhamento com base em inundação

O algoritmo é, como já se disse, muito simples: quando um nó da rede (o termo genérico para um computador ou um comutador) recebe um pacote, analisa o seu endereço de destino, e se este for diferente do seu, envia uma cópia do mesmo por cada uma das suas interfaces, excepto por aquela pela qual o pacote foi recebido. Se o pacote é dirigido ao nó, este processa-o sem continuar a encaminhá-lo (excepto se o endereço de destino for o endereço de *broadcasting*, caso em que a inundação deve prosseguir). Se um nó só tem um canal, e recebe um pacote que não lhe é dirigido, o algoritmo determina que o nó destrua esse pacote pois não pode voltar a enviá-lo pelo canal pelo qual o recebeu.

Esta forma de encaminhamento chama-se **encaminhamento por inundação** (*flooding*) pela analogia sugerida: a rede é inundada pelo pacote.

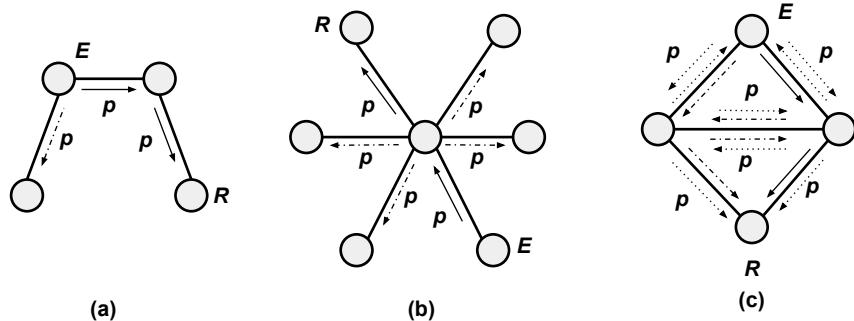


Figura 15.1: Encaminhamento por inundação do pacote  $p$  do emissor  $E$  para o receptor  $R$ . As setas correspondem a cópias do pacote: as a negro são as necessárias; as a ponto e traço são inúteis; e as a ponteado são algumas das cópias duplicadas.

A Figura 15.1 mostra a utilização do encaminhamento por inundação de um pacote ( $p$ ) emitido pelo nó origem ou emissor ( $E$ ), para o nó destino ou receptor ( $R$ ), em três redes diferentes. Nos primeiros dois casos ((a) e (b)), a inundação encaminha o pacote  $p$  de  $E$  para  $R$  correctamente, mas introduz desperdício pois há cópias inúteis do pacote que são encaminhadas para outros nós que acabam por as descartar por não terem mais alternativas de emissão. No caso (c), não só são introduzidas cópias inúteis, mas também pacotes duplicados (que os nós não identificam como tal) pelo que continuam a encaminhá-los sem fim, conduzindo a mais cópias inúteis e a outros duplicados, e o processo continua eternamente. Com efeito, o pacote  $p$  acaba por dar a volta em ciclo e retorna ao emissor inicial  $E$ , este não o reconhecendo como sendo duplicado do anterior, volta a enviá-lo como da primeira vez, e todo o processo recomeça sem

nunca acabar. Desde que a rede tenha ciclos, *i.e.*, caminhos que começam e acabam no mesmo nó, o processo não tem fim e não só o algoritmo não termina, como introduz uma **tempestade de difusão** (*broadcast storm*) que satura e inutiliza a rede.

A diferença entre os dois primeiros casos e o terceiro reside no facto de que nos primeiros as redes não têm ciclos, pelo que a inundação introduz cópias inúteis, mas não duplicados. No terceiro caso a rede tem ciclos e a inundação introduz cópias inúteis, duplicados, e uma inundação sem fim que provoca o colapso da rede.

O algoritmo de **encaminhamento por inundação** é muito simples: quando um nó recebe um pacote, analisa o seu endereço de destino, e se este for diferente do seu, envia uma cópia do mesmo por cada uma das suas interfaces, excepto por aquela pela qual o pacote foi recebido.

Este algoritmo consegue encaminhar um pacote até ao seu destino mas infelizmente, conduz ao envio de pacotes inúteis nas redes estruturadas em árvore, e a um fenómeno de colapso da rede designado por **tempestade de difusão** (*broadcast storm*) em redes com ciclos.

Veremos a seguir que existem métodos para obviar os problemas da inundação. Comecemos pelo mais grave, a tempestade de difusão no caso em que a rede tem ciclos.

### Deteção e supressão de pacotes duplicados

Uma primeira forma de acabar com os pacotes duplicados, ou de pelo menos minorar a sua presença, consiste em introduzir um TTL (*Time To Live*) como o dos pacotes IP. Este campo do cabeçalho do pacote contém um contador decrescente dos nós já atravessados pelo mesmo.

Quando o nó inicial emite pela primeira vez  $p$ , inicializa o seu **ttl** com um valor positivo, por exemplo 10. Sempre que um pacote chega a um nó, este decrementa o seu **ttl** e, se este valer 0, descarta o pacote e não o processa. Senão, processa o pacote com o novo valor do **ttl**. Se houver necessidade de enviar mais do que uma cópia do pacote, o valor do **ttl** de cada uma dessas cópias é sempre o mesmo. Ou seja, o **ttl** de  $p$  só pode decrescer. Naturalmente, o método baseado num TTL permite limitar a tempestade, mas não a estanca à nascença e, adicionalmente, o emissor não conhecendo a configuração da rede, usa um valor de TTL suficiente para não impedir nenhum pacote de chegar ao destino. Esse valor tem de ser necessariamente muito alto para a maioria dos casos pois o emissor não consegue saber qual seria o valor adequado. Dada uma rede, o valor por omissão do maior TTL necessário é o tamanho do maior caminho na mesma, também chamado o diâmetro da rede. A utilização de um TTL é uma segurança contra algo que funcione mal e acaba por descartar, mais ou tarde ou mais cedo, pacotes em ciclo pois esses pacotes constituem um perigo fatal para a rede se não forem bloqueados.

Existe, no entanto, um método mais complexo, mas também mais eficaz, de estancar os duplicados logo à nascença. Trata-se de um método que permite aos nós detectarem que um pacote é um duplicado.

Esse método consiste em colocar uma marca em cada um dos pacotes emitidos. Sempre que um nó recebe um pacote, começa por verificar se já viu um pacote com uma marca igual. Se sim, o pacote é um duplicado, senão, regista a sua marca e processa-o. Mas como arranjar uma marca distinta para cada pacote? Um método relativamente simples consiste em usar como marca dos pacotes o par (endereço do emissor original do pacote, número de sequência). O número de sequência é colocado no cabeçalho pelo emissor original sempre que emite um novo pacote. Nenhum outro nó altera esse

número de sequência. Desta forma, à entrada da rede, cada pacote recebe uma marca distinta de todos os outros, uma espécie de “número fiscal do pacote”.

Se o número de sequência usado em diferentes pacotes por cada emissor for sempre crescente, cada nó só tem que memorizar, para cada nó da rede, o último número de sequência recebido num pacote emitido originalmente por esse nó. Se os números de sequência não forem sempre crescentes, cada nó tem de memorizar tantos números distintos quantos os pacotes. Ou seja, com números de sequência crescentes, a tabela de números de pacotes já recebidos é  $O(\text{numero de nós da rede})$ ,<sup>1</sup> no caso contrário é  $O(\text{número de pacotes recebidos})$ .

No parágrafo que se segue vamos discutir uma variante do algoritmo de deteção de duplicados quando o objectivo é receber as mensagens por uma ordem crescente das marcas. Dado que esse algoritmo é utilizado a diferentes níveis da arquitectura das redes, utilizaremos o termo mensagem e não pacote para designar as unidades de informação trocadas pelos nós da rede.

### Difusão fiável de mensagens

O algoritmo de inundação implementa directamente a comunicação de um para muitos ou difusão (*broadcasting*), *i.e.*, a possibilidade de uma mensagem ser enviada a todos os nós da rede. Basta que nesse caso o endereço de destino seja especial, *i.e.*, o endereço de *broadcasting*. É claro que os problemas dos duplicados e dos pacotes inúteis se continuam a colocar e é necessário usar um mecanismo que os permita resolver. Pelo menos o primeiro para evitar uma tempestade de difusão. A utilização de números de sequência sempre crescentes para suprimir duplicados pode também ser usada para implementar uma versão do protocolo de difusão designada **algoritmo de difusão fiável**.

Este algoritmo permite que as mensagens enviadas em difusão por cada um dos nós sejam recebidas por todos os outros pela ordem com que cada uma delas foi emitida. No entanto, dependente da versão do algoritmo, existem várias variantes de relação de ordem. Na versão aqui apresentada, essa relação de ordem só garante a ordenação das mensagens emitidas pelo mesmo nó, não a das mensagens emitidas por diferentes nós. Do ponto de vista das garantias de recepção de todas as mensagens, existem também várias versões deste algoritmo, cada uma com diferentes garantias. Aquela que vamos apresentar é das mais simples e implementa apenas algumas das garantias possíveis, nomeadamente que um nó só considera as mensagens mais recentes, pela ordem de emissão, mas pode saltar por cima de alguma mensagem ainda não chegada se no entretanto recebeu outra mais recente.

O algoritmo é um algoritmo de inundação em que cada nó emite uma sequência de mensagens numeradas com números de sequência sempre crescentes e, desde que a rede seja conexa (*i.e.*, existe pelo menos um caminho entre quaisquer dois nós), as mensagens emitidas por um nó são recebidas pelos diferentes nós por uma ordem compatível com a ordem com que foram emitidas.

Para implementar esta versão da difusão fiável, a troca de mensagens entre nós é feita usando uma variante do protocolo *stop & wait*. A diferença relativamente a este é que quando o nó receptor recebe uma mensagem que já possui, porque o emissor está atrasado nas mensagens que já recebeu, ao invés de lhe enviar um ACK, envia-lhe uma cópia da mensagem mais recente que possui. O algoritmo garante que os diferentes nós recebem as mensagens por uma ordem compatível com a ordem com que foram emitidas por cada emissor, mas não garante que o nó tenha acesso à sequência integral das mensagens enviadas. O algoritmo é apresentado na Listagem 15.1 para uma rede só com canais ponto-a-ponto. Repare-se que o nó não analisa o endereço de destino pois pressupõe-se que este é o endereço de *broadcasting*.

---

<sup>1</sup> No caso de a rede poder trocar a ordem de entrega dos pacotes, ou perder alguns, é preferível memorizar intervalos de números de sequência já recebidos de cada nó da rede.

Listing 15.1: Pseudo-código do algoritmo de difusão flável - tratamento local da mensagem  $m$

```

1 // Message database with the most recent message from each participant
2 // node in the network, that has been received by this node. Usage:
3 //
4 //     msgDatabase.put(message)
5 //
6 //     message = msgDatabase.getMessage(node)
7 //
8 // returns the most recent message from node or null if none
9 // has been received from that node
10
11 processMessage ( message msg, interface in ) {
12     seq = msg.getSequence()
13     node = msg.getNode()
14     lastSeen = msgDatabase.getMessage(node)
15     if ( lastSeen != null ) lastSeq = lastSeen.getSequence()
16     if ( lastSeq == null ) { // first message from this node
17         msgDatabase.put ( msg )
18         send ( in, ACK )
19         flood ( in, msg )
20     }
21     else if ( lastSeq < seq ) { // more recent message
22         msgDatabase.put ( msg )
23         send ( in, ACK )
24         flood ( in, msg )
25     }
26     else if ( lastSeq == seq ) { // duplicate
27         send ( in, ACK )
28     }
29     else if ( lastSeq > seq ) { // late message
30         send ( in, lastSeen )
31     }
32 }
```

O algoritmo apresentado é um dos mais simples de difusão fiável. Este complica-se mais se se pretender que cada nó receba exactamente todas as mensagens emitidas por cada outro nó. No caso de se pretender também uma ordem total de todas as mensagens (comum a todos os nós), o algoritmo necessário é também mais complicado.

Como veremos no Capítulo 16 este algoritmo é usado para replicar a configuração de uma rede de forma fiável entre um conjunto de comutadores de pacotes. Para esse efeito, o algoritmo utiliza um conjunto de medidas complementares, nomeadamente as seguintes. A base de dados de mensagens tem um TTL associado a cada nó e se este não origina mensagens durante um período alargado (*e.g.*, algumas dezenas de minutos), o mesmo é “esquecido”. Desta forma, é mais fácil de garantir que os identificadores dos nós são sempre necessariamente diferentes uns dos outros. Também, quando um nó teve uma avaria e desaparece da rede mas volta a integrá-la antes de ser esquecido, assim que se manifestar, os seus vizinhos têm oportunidade de o actualizar sobre o último número de sequência que utilizou, sob pena de as suas mensagens, por terem um número de sequência muito atrasado, serem ignoradas até voltarem a ter um número de sequência maior que o último que utilizou antes de ter “morrido e ressuscitado”.

O algoritmo de encaminhamento por inundaçāo pode ser complementado com um mecanismo baseado em números de sequência que permite detectar as mensagens duplicadas e parar o processo de inundaçāo. Apesar de este mecanismo não evitar o envio de pacotes inúteis, permite implementar **algoritmos de difusão fiável baseados em inundaçāo**.

### **Supressão de pacotes inúteis - aprendizagem pelo caminho inverso**

Os métodos acima apresentados permitem minorar ou acabar de todo com os pacotes e mensagens duplicadas, mas não permitem minorar ou acabar com os pacotes inúteis que a inundaçāo introduz, que são bem evidentes nos casos (a) e (b) da Figura 15.1.

Nos contextos em que é possível acabar com os duplicados de forma segura, existe um algoritmo que permite evitar a introdução de pacotes inúteis. Esse algoritmo é designado por **aprendizagem pelo caminho inverso (*backward learning*)**, ou filtragem pelo caminho inverso. Quando a rede não tem ciclos é mais fácil perceber o algoritmo e é também mais fácil ficar convicto da sua eficácia. Uma rede sem ciclos é designada em teoria de grafos uma árvore. Numa tal rede, entre quaisquer dois nós *E* e *R* só existe um caminho, e por isso a inundaçāo não introduz duplicados, apenas pacotes inúteis.

O algoritmo de aprendizagem pelo caminho inverso baseia-se na existência em cada nó de uma tabela de nós conhecidos. Nessa tabela, associados a cada nó conhecido, existem dois valores: o número da interface que permite chegar a esse nó e um TTL.

Sempre que um nó recebe um pacote *p*, analisa nessa tabela a entrada correspondente ao endereço origem desse pacote: *p.origem()*. Se essa entrada estiver vazia, quer dizer que o nó nunca recebeu um pacote que tenha sido originalmente emitido por *p.origem()* e não o conhece. Por isso cria uma nova entrada na tabela e coloca na mesma o número da interface pela qual *p* foi recebido. Adicionalmente o TTL associado a cada entrada é inicializado com um valor normalizado, por exemplo 120 segundos. Caso o nó já seja conhecido, a entrada na tabela é refrescada com o número de interface e o TTL é de novo inicializado.

Mas para que serve a tabela? Quando um nó recebe um pacote *p* dirigido ao nó *p.destino()*, o receptor vai ver se já conhece esse nó consultando a sua tabela. Se sim, então já sabe porque interface o pode alcançar, senão só lhe resta fazer inundaçāo. Com efeito, se todos os canais forem bidireccionais, numa rede em que só existe um caminho entre cada dois nós, o caminho inverso permite chegar ao nó origem.

O processo é ilustrado na Figura 15.2. Inicialmente o nó de comutação de pacotes não conhece nada sobre os computadores que lhe estão ligados. Mais tarde, o com-

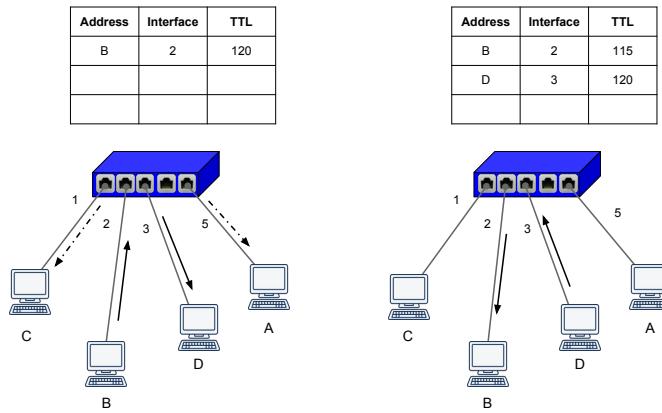


Figura 15.2: Encaminhamento por inundação e aprendizagem pelo caminho inverso: B envia um pacote a D, depois D responde a B. As setas a ponto e traço representam mensagens inúteis.

Computador com o endereço B envia um pacote para o computador com o endereço D. O nó de comutação, não conhece D pelo que só lhe resta fazer a inundação, mas antes ficou a saber que B está ligado à sua interface 2 e regista esse facto na tabela com o TTL 120. Passados 5 segundos, D responde a B, mas desta vez o nó de comutação já conhece a localização de B pelo que não necessita de fazer inundação. Aproveita no entanto para registar a interface que dá acesso a D.

Este algoritmo de inundação e aprendizagem pelo caminho inverso é usado pelos nós de comutação (*switches*) Ethernet para encaminharem *frames* entre os computadores que lhe estão ligados. O algoritmo que executam figura na Listagem 15.2 que apresenta o seu pseudo código. A tabela dos *switches* toma então o nome de *MAC Address Table* ou simplesmente `macTable` na listagem.

Listing 15.2: Pseudo-código do algoritmo de tratamento por um nó de um pacote *p* recebido pela interface *in* através do algoritmo de inundação com aprendizagem pelo caminho inverso.

```

1  TTL = 120
2  processPacket ( packet p, interface incoming ) {
3      macTable.put(p.getOrigin(), incoming, TTL)
4      // is p locally addressed ?
5      if ( p.getDestination == self.getID() ) {
6          // packet p got to its destination
7          locally process packet p
8          return // done
9      }
10     interface outgoing = macTable.get(p.getDestination)
11     if ( outgoing == null ) flood(p) // not in macTable
12     else if ( outgoing != incoming ) outgoing.send(p)
13     // else ignore p
14 }
```

Repare-se que o endereço de *broadcasting* nunca é endereço origem pelo que nunca estará presente na tabela, logo todos os pacotes dirigidos a esse endereço são necessariamente *flooded*, o que implementa naturalmente a difusão.

## 15.2 Comutação Ethernet (Ethernet *switching*)

Quando os *switches*<sup>2</sup> Ethernet<sup>3</sup> foram introduzidos, foi necessário decidir como seria possível introduzir o conceito de comutação de pacotes para encaminhamento de *frames* Ethernet entre vários canais de difusão. Por diversas razões, provavelmente ligadas a não se pretender alterar o cabeçalho dos *frames* Ethernet, nem introduzir complexidade suplementar nos comutadores, optou-se por não usar nenhum algoritmo de deteção de duplicados (com números de sequência) ou mesmo de simples limitação do seu tempo de vida (com TTLs) e por essa razão optou-se por só admitir a introdução destes equipamentos em redes com a configuração de uma árvore.

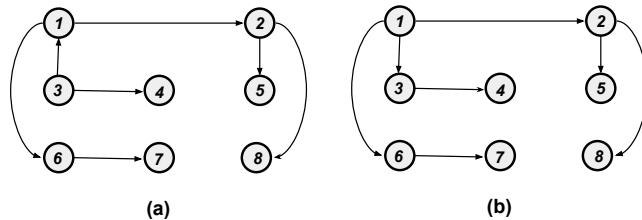


Figura 15.3: A difusão não introduz duplicados numa rede estruturada em árvore. No caso (a) o emissor é o nó 3 e no caso (b) é o nó 1.

Numa rede estruturada em árvore, o algoritmo de inundaçāo permite o encaminhamento de um nō para outro nō, assim como a difusão, sem necessidade de recorrer à deteção de duplicados porque a inundaçāo não os introduz numa rede sem ciclos.

Por outro lado, o mecanismo de **aprendizagem ou filtragem pelo caminho inverso (backward learning)** com base nos endereços MAC das interfaces Ethernet dos computadores, permite restringir significativamente o envio de *frames* inúteis.

Como os *frames* IEEE 802.11 podem ser facilmente transformados em *frames* IEEE 802.3 pelos APs, a rede pode conter comutadores de pacotes (*switches* Ethernet e APs), canais ponto-a-ponto, canais multi-ponto (IEEE 802.3 ou 802.11) e interfaces *full* e *half-duplex*.

A Figura 15.4 mostra um exemplo de uma rede Ethernet, organizada em árvore, composta por vários tipos de canais, *switches*, um AP e um *hub*, que formam uma rede em que todos os computadores podem endereçar sem problemas *frames* uns aos outros, incluindo de um para todos. Uma rede deste tipo costuma chamar-se uma **rede Ethernet comutada** ou **switched Ethernet network** na terminologia em língua inglesa.

O algoritmo de aprendizagem pelo caminho inverso tem por objectivo implementar uma optimização da inundaçāo, transformando-a num encaminhamento por um

<sup>2</sup> Como já foi várias vezes referido, um *switch* Ethernet pode ser designado na língua portuguesa por comutador Ethernet ou nō de comutação de *frames* Ethernet. No entanto, por esses termos não serem utilizados pelos profissionais de redes, continuaremos a utilizar termos em língua inglesa.

<sup>3</sup> Na verdade os *switches* Ethernet foram antecedidos por outros equipamentos chamados *Bridges* que tinham grosso modo as mesmas funcionalidades que os *switches* mas que usavam outro tipo de canais de interligação ponto-a-ponto.

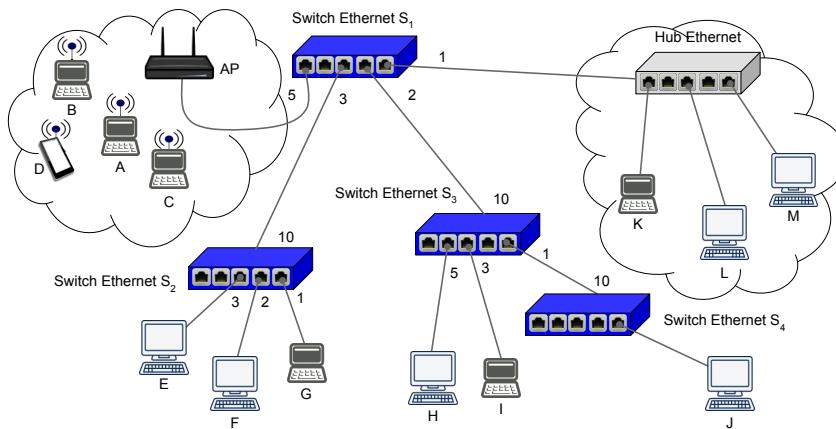


Figura 15.4: Rede com diferentes tipos de comutadores e de canais com encaminhamento por inundação e aprendizagem pelo caminho inverso.

caminho directo. Com efeito, na sua ausência os *frames* chegariam ao destino mas a rede conteria muito tráfego inútil. Com este algoritmo, é feita uma primeira inundação quando um endereço MAC de destino é desconhecido mas, a partir daí, deixam de ser enviados *frames* inúteis para a origem desse pacote.

O teste na linha 12 da Listagem 15.2 assegura que se um *frame* chegou ao *switch* pela interface por detrás da qual já se sabe estar o seu destino, então este já o deve ter recebido e é inútil continuar a encaminhá-lo. O próprio conceito de inundação determina que um pacote nunca deve ser enviado pelo canal pelo qual chegou ao nó de comutação. Por outro lado, se a essa interface estivesse ligado um canal multi-ponto, sem esse teste o algoritmo provocaria a recepção de duplicados pelo destino.

Desde que a rede seja organizada em árvore, o algoritmo de aprendizagem funciona perfeitamente. Por exemplo, se o computador E enviar um *frame* ao computador M, como o destino não é conhecido, o *frame* inunda a rede e todos os *switches* e o AP (o AP também executa o algoritmo), passarão a saber porque porta poderão aceder a E. Por exemplo, o *switch* S<sub>4</sub> passará a saber que E está por detrás da sua porta 10. Após todos os computadores enviarem *frames* uns aos outros, todos serão conhecidos por todos os nós de comutação. Por exemplo, o *switch* S<sub>1</sub> saberá que os computadores H, I e J estarão por detrás da sua porta 2. A maneira mais simples de um computador se dar a conhecer a todos os equipamentos de comutação (*switching*) é enviar um *frame* dirigido ao endereço de broadcast.

Um aspecto cuja discussão é interessante tem a ver com o dinamismo dos computadores e a gestão dos TTLs. Para que servem os TTLs das entradas das tabelas usadas pelo algoritmo de aprendizagem pelo caminho inverso? Existem várias razões para a sua presença. A mais óbvia tem a ver com o facto de que os computadores se podem desligar e, nesse caso, é inútil ter entradas nas tabelas ocupadas com os seus endereços (apesar de isso poder poupar algumas inundações). Assim, se o computador deixar de emitir pacotes, as entradas que lhe dizem respeito acabarão por expirar e deixarão de ocupar espaço.

A outra razão está ligada com o facto de que pode haver mais computadores do que entradas disponíveis nas tabelas. Quando houver falta de espaço para introduzir mais um endereço MAC na tabela, é necessário retirar uma das entradas para criar espaço. Uma solução possível consiste em retirar da tabela a entrada há mais tempo sem ser usada, i.e., aquela cujo TTL é o mais baixo. Esta razão tem hoje em dia

bastante menos razão de ser em redes pequenas pois a maioria dos *switches* Ethernet têm tabelas com pelo menos centenas de entradas.

No entanto, é preciso ter em atenção que o processamento de um *frame* deve ser tão rápido quanto possível quando comparado com o tempo da sua transmissão. Ora um *frame* com 1.000 bits é transmitido em  $1\ \mu$  segundo a 1 Gbps ( $1000/10^9 = 10^6$ ). Por esta razão, os *switches* usam circuitos integrados especiais<sup>4</sup> com memórias muito rápidas para implementar as tabelas de endereços MAC e os algoritmos que as manipulam. Essas memórias rápidas são caras e consomem muita energia. Os *switches* de gama mais alta têm tabelas destas com 64 K entradas o que aumenta bastante o seu custo.

Finalmente, o TTL está também ligado ao facto de que os computadores hoje em dia são na sua maioria portáteis e portanto podem mudar de AP ou de porta a que estão ligados. Quando isso acontece, a tabela fica desactualizada e poderia deixar de ser possível enviar *frames* ao computador que migrou de localização na rede. Até o TTL expirar, os *frames* dirigidos a esse endereço MAC seriam dirigidos para o local errado.

Na verdade, isto só seria verdade se esse computador só recebesse tráfego e não enviasse nenhum. Na prática, o computador que mudou de localização acabará por enviar *frames*, e assim que o fizer, começa a actualizar as entradas que lhe dizem respeito nas diferentes tabelas. Tudo depende do destino dos *frames* que enviar. A maneira mais eficaz de assegurar a actualização total e instantânea consiste em o computador que migrou enviar um *frame* em difusão assim que se voltar a ligar à rede. Desta forma todas as entradas são imediatamente actualizadas. Actualmente, a maioria dos sistemas de operação dos computadores e os APs, sempre que uma interface muda de localização, fazem automaticamente esse *broadcast*.

As razões que acabámos de apresentar mostram que continua a ser importante usar um TTL associado a cada entrada das tabelas de endereços, mas deixou de ser muito importante discutir qual o valor com que este deve ser inicializado. Basta assegurar que este não induz inundações inúteis com demasiada frequência.

### 15.3 Árvores de cobertura e STP

Na prática, quando uma rede tem um pequeno âmbito e está sobredimensionada para o tráfego que a utiliza, é relativamente fácil organizá-la como uma árvore. Nessa rede o tempo de propagação de extremo a extremo é desprezável, a probabilidade de avarias é baixa, e o tráfego não seguir pelo caminho mais directo não é grave. A Figura 15.5 representa uma rede em que dois canais multi-ponto, a distâncias geográficas de vários quilómetros, estão interligados por dois canais ponto-a-ponto. O objectivo é garantir que em caso de avaria de um desses canais é possível continuar a comunicar usando o outro, pois a reparação de canais de longa distância pode levar bastante tempo. Infelizmente a configuração tem um ciclo e essa rede de *switches* Ethernet entraria em colapso com uma tempestade de difusão (*broadcast storm*).

Para resolver este problema foi introduzido um protocolo que a todo o momento calcula de forma distribuída uma árvore de cobertura da rede e desliga os canais que não fazem parte dessa árvore. No entanto, se um dos canais que faz parte da árvore se avariar, é recalculada uma nova árvore e a rede é reconfigurada automaticamente.

Dado um grafo conexo, *i.e.*, em que existe pelo menos um caminho entre quaisquer dois nós, uma **árvore de cobertura** do mesmo é um seu sub-grafo contendo todos os nós e que é uma árvore, *i.e.*, no qual só existe um caminho entre quaisquer dois nós.

---

<sup>4</sup> Estes circuitos designam-se por CAMs - *Content Addressable Memories*.

O protocolo que permite a um conjunto de *switches* Ethernet calcular uma árvore de cobertura e apenas operar de acordo com a mesma, chama-se **SPT - Spanning Tree Protocol** e foi normalizado pela norma IEEE 802.1D.

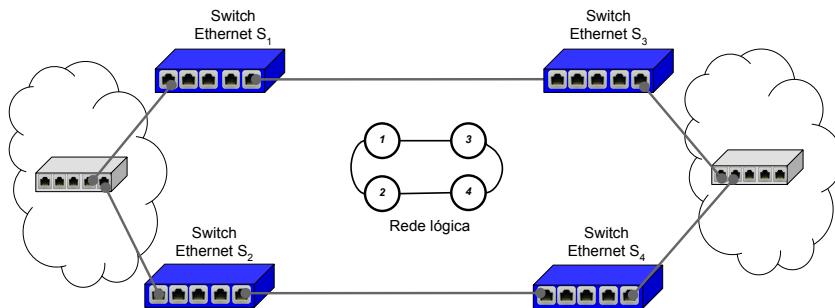


Figura 15.5: Rede com uma configuração necessária para efeitos de tolerância a falhas mas que introduz ciclos.

### O protocolo STP

O protocolo seleciona na rede os canais que fazem parte da árvore e coloca em inactividade os que não fazem, ver a Figura 15.6. Nela, a parte (a) mostra a rede completa e a (b) uma árvore de cobertura calculada pelo STP. Após a avaria do canal que liga o nó 1 ao 2, o STP reconfigura a rede e adopta a árvore de cobertura (c), desactivando uns canais e activando outros, de forma a manter de novo uma configuração em árvore que cobre todos os nós. Nesta secção usaremos com frequência os termos originais em língua inglesa introduzidos com a descrição do protocolo.

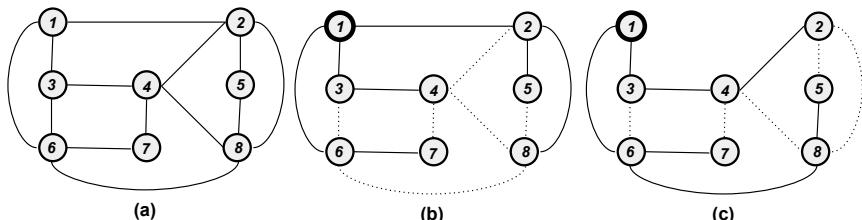


Figura 15.6: Papel do protocolo STP: (a) rede completa, (b) uma sub-rede calculada pelo STP, (c) outra sub-rede calculada pelo STP após a avaria do canal que liga os nós 1 e 2.

O STP calcula uma árvore começando por eleger um *switch* com o papel de raiz da árvore. No caso das redes das figuras 15.6 e 15.7 a raiz é o nó 1. Todos os outros *switches* se ligam a essa árvore. A inundação vai depois funcionar enviando *frames* na direção da raiz, no sentido ascendente, e *frames* no sentido das folhas, no sentido descendente, ver a Figura 15.7. Uma forma de visualizar este tipo de encaminhamento é imaginar que o *switch* no qual cada *frame* emitido tem origem, funciona como raiz

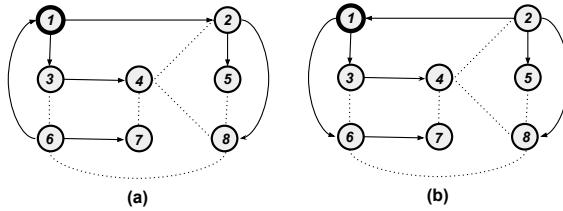


Figura 15.7: Uso da árvore STP para inundaçāo. A raiz da árvore STP é o *switch* 1. Encaminhamento de um *frame* emitido pelo *switch* 6 (a) e pelo *switch* 2 (b) usando a árvore calculada pelo STP.

de uma árvore para efeitos da inundaçāo. Repare-se, no entanto, que estas árvores, pelo menos nos exemplos apresentados, são bastante desequilibradas.

O protocolo funciona com base nos conceitos a seguir introduzidos.

**Switch ID** é o identificador de cada *switch*. É composto pela concatenação de 12 bits de prioridade com os 48 bits do endereço MAC do *switch*.

**Root switch** ou *switch* raiz é o *switch* cujo identificador é o mais baixo em toda a rede. O administrador da rede pode forçar um certo *switch* a ser a raiz atribuindo-lhe uma prioridade mais baixa que a dos outros, senāo a raiz é simplesmente escolhida com base nos endereços MAC, de forma mais ou menos aleatória. O *root switch* será a raiz da árvore calculada pelo STP.

**Port ID** é o identificador de cada porta de um *switch*. Todas as portas de um *switch* têm identificadores distintos.

**Link cost** é o custo de cada canal. O custo varia de 1 a 200.000.000. O custo dos diferentes canais é definido pela norma e é inversamente proporcional ao seu débito (quanto menor o débito, maior o custo). Por exemplo, o custo de um canal a 1 Gbps é 4, a 100 Mbps é 19, a 10 Mbps é 100, etc.

**Root path cost** é o custo do caminho (o somatório dos custos dos canais que este atravessa) que liga um dado *switch* ao *root switch*. Em cada *switch*, a porta que o liga ao canal onde começa o caminho para a raiz é designada a **Root port**. Só pode haver uma destas portas por *switch*.

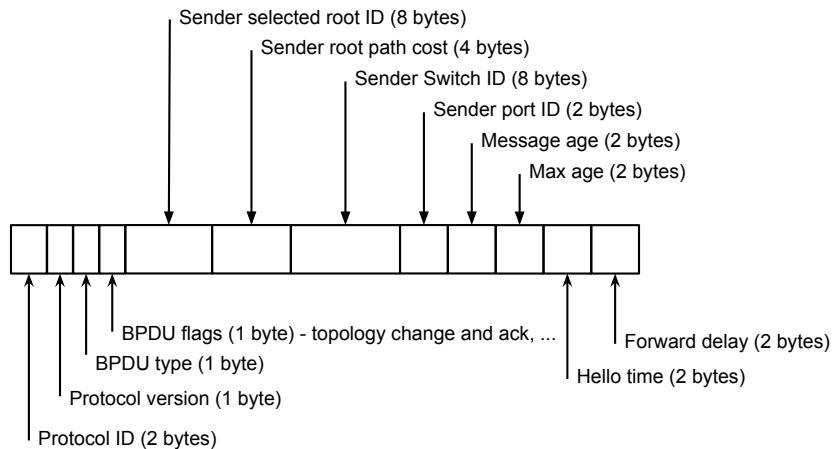
**Designated port** é a porta de um canal responsável por encaminhar *frames* até à raiz. Este conceito é especialmente importante em portas ligadas a canais multi-ponto ao qual estão ligados diversos *switches*. Este aspecto será retomado a seguir.

A árvore seleccionada pelo STP é caracterizada por ser uma árvore de caminhos mais curtos (*i.e.*, mais “baratos”) desde a raiz até cada uma das folhas. Nos casos em que existirem diversas alternativas de caminhos mais curtos, o STP dispõe de regras para optar de forma unívoca por um deles.

O protocolo desenvolve-se em várias fases, nomeadamente: 1. eleição da raiz; 2. calcular os caminhos mais curtos da raiz para cada *switch*; 3. bloquear os canais que não fazem parte desses caminhos; e 4. activar posteriormente os restantes canais.

O STP baseia-se num único tipo de mensagens, designadas *frames* BPDU<sup>5</sup> e descritas na Figura 15.8. Através de um BPDU o emissor comunica aos seus vizinhos

<sup>5</sup> Os primeiros equipamentos que usaram este protocolo eram designados por pontes (*bridges*). Esta designação tinha por origem a ideia de que o seu principal papel era a interligação

Figura 15.8: STP BPDUs (*Bridge Protocol Data Unit*)

qual é, na sua opinião, a raiz, a que distância (custo) esta está, qual o seu ID e a porta porque enviou este BPDU, assim como diversos parâmetros de funcionamento do protocolo. Alguns dos campos já foram referidos ou serão referidos a seguir mas de qualquer forma seguem-se algumas breves observações sobre alguns deles. Os campos ID da raiz segundo o emissor do BPDU e ID do emissor são expressos 8 bytes cada: 2 bytes com a prioridade (0x8000 por omissão mas este valor pode ser alterado pelo gestor da rede) mais 6 bytes com o MAC Address. O campo com o ID da porta do emissor é expresso em 1 byte para a prioridade (pode ser alterada pelo gestor) mais 1 byte para o número de porta. O campo *maximum age* é o valor do temporizador que dá o alarme caso não sejam recebidos BPDUs. O campo *Hello time* é o tempo que separa a emissão de cada BPDU pela raiz (geralmente 2 segundos) e o campo *forward delay* é o tempo que leva uma porta a transitar para o estado *forwarding*. Todos os tempos são codificados em múltiplos de 8 ms (1/256 segundos).

#### Eleição da raiz e selecção do caminho mais curto para a mesma

Numa primeira fase, após a inicialização de um *switch*, ou na sequência da deteção de um evento que obriga a uma reconfiguração, a rede elege o *root switch*. Para esse efeito, todos os *switches* se candidatam a ser raiz e começam a difundir periodicamente (*e.g.*, a cada 2 segundos), por inundação, BPDUs em que se consideram raiz da rede. Estes BPDUs iniciais contêm como raiz o ID do emissor, visto que este se considera a si próprio raiz, e por isso também anuncia que o seu custo até à raiz é 0.

Quando um *switch* recebe BPDUs de outros, compara o identificador da raiz recebida com o identificador do *switch* que considera raiz (ou seja ele próprio no início). Se o ID da raiz recebido for maior, ignora a mensagem recebida, se for menor, passa a considerar o ID anunciado como sendo o ID da nova raiz, e o canal pelo qual recebeu este BPDU, o canal da *root port*, *i.e.*, o canal do caminho mais curto para a raiz. Se for igual, mas recebeu este BPDU de uma porta diferente da que considera como *root port*, muda de *root port* caso o custo para chegar à raiz seja menor pela nova porta, ou sendo igual, se o ID do vizinho, ou o da porta pelo qual este lhe enviou o BPDU, são inferiores.

---

de vários canais multi-ponto entre si como ilustrado pela Figura 15.5. As pontes foram completamente substituídas pelos *switches* Ethernet e o termo caiu em desuso, no entanto, o termo BPDU (*Bridge Protocol Data Unit*) está na norma STP e mantém-se em uso.

Quando um *switch* recebe um BPDU enviado pelo *switch* que elegeu (ou acaba de eleger) como raiz e que lhe chegou pela *root port*, actualiza o custo até à raiz e a idade do BPDU, e difunde aos vizinhos por inundação um novo BPDU com a raiz e o seu custo para lá chegar. Senão, o BPDU recebido é ignorado pois não acrescentaria nenhuma informação relevante aos outros *switches*. Só o *switch* raiz é que continua a enviar periodicamente BPDUs, os restantes apenas difundem por inundação os BPDUs recebidos do *switch* que consideram raiz.

Este procedimento elege uma raiz única e permite igualmente a cada *switch* seleccionar o caminho mais curto para a mesma. Nos casos em que há diversas alternativas de caminho mais curto, as regras sobre os identificadores dos *switches* e das portas resolvem a ambiguidade.

A Figura 15.9 ilustra algumas dessas situações. No caso (a) o *switch* 1 é eleito para raiz (menor identificador) e o *switch* 4 escolhe o 2 para chegar à raiz visto que o 3, que oferece um caminho de igual custo, mas tem um identificador maior que o 2. No caso (b) existem dois canais entre os *switches* 1 e 3, mas aquele que do lado do *switch* 1 está ligado ao número de porta mais baixo é seleccionado visto que os dois canais têm o mesmo custo. No caso (c) entre os *switches* 1 e 3 é escolhido o canal de mais baixo custo visto que o outro tem um custo superior.

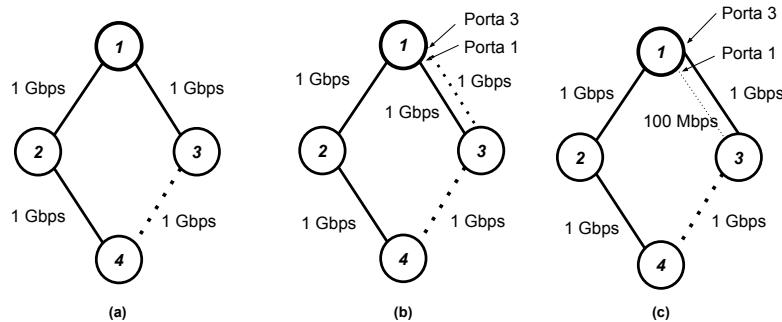


Figura 15.9: Escolha do caminho mais curto em casos aparentemente ambíguos. Cada *switch* é representado por um círculo com o seu ID no centro.

O protocolo STP baseia o seu funcionamento na inundação periódica da rede com *frames* de controlo chamados BPDUs, emitidos inicialmente por todos os *switches*, mas mais tarde apenas pelo *switch* eleito como raiz.

Cada participante na inundação, altera os BPDUs de forma que torna possível realizar a eleição da raiz e determinar o caminho mais curto até à mesma. Este caminho também é o mais curto desde a raiz até cada *switch* se todos os canais tiverem o mesmo débito nos dois sentidos, o que é o caso mais comum.

### Escolha das *designated ports*

Em todos os canais que fazem parte de caminhos mais curtos para a raiz (os únicos canais que ficarão activos) é necessário escolher uma *designated port*. Caso o canal seja ponto-a-ponto e *full-duplex* só existem duas portas ligadas ao mesmo e a *designated port* é a que está no extremo oposto à da *root port*.

No entanto, no caso de canais multi-ponto, o problema é mais delicado pois ao mesmo canal podem estar ligados mais do que um *switch* e eventualmente cada um terá caminhos mais curtos com o mesmo custo para a raiz.

Nesse caso é necessário eleger apenas uma *designated port* entre os diferentes *switches* e portas ligados ao mesmo canal. Esta porta será a porta do *switch* com a menor distância até à raiz, ou com o menor ID se existirem vários à mesma distância da raiz, ou com o menor ID de porta se existirem várias do mesmo *switch* ligadas a esse canal, e que nenhuma é *root port* desse *switch* pois uma porta não pode ter os dois papéis. A eleição segue o mesmo processo que o usado na eleição da raiz. Cada *switch* considera cada porta candidata como *designated port* mas à medida que vai recebendo por ela BPDUs dos outros, ou se não receber nenhum, pode tomar uma decisão de acordo com as regras indicadas acima.

Uma *designated port* assume a responsabilidade de enviar para o canal os *frames* recebidos da raiz e de enviar para a raiz os *frames* recebidos do canal. As outras portas não eleitas devem ficar bloqueadas se forem de *switches* que executam o protocolo STP. De facto, qualquer nó ligado à mesma rede (computador, *switch* sem suporte de STP, *hub*, etc.) apenas passa os BPDUs sem os interpretar e não gera BPDUs. Como os BPDUs são sempre dirigidos ao endereço de *broadcast* nunca são interceptados por *switches* sem suporte de STP.

A Figura 15.10 ilustra o processo e põe em evidência que sem esta precaução seria introduzida uma tempestade de difusão. Sem a noção de *designated port* um *frame* emitido por um dos computadores chegaria em duplicado aos diferentes *switches* pois o canal multi-ponto, necessário para chegar aos computadores, introduziria igualmente um canal entre os *switches* 2 e 3.

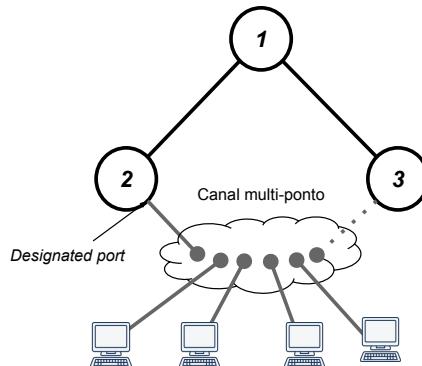


Figura 15.10: Eleição da *designated port* num canal multi-ponto

#### Estabilização do protocolo e início do encaminhamento

Enquanto o STP não estabiliza as suas decisões, não podem ser encaminhados *frames* de dados pois pode existir o risco de os encaminhar através de ciclos. Assim, só depois de o STP estabilizar é que os *switches* passam a encaminhar os *frames* que recebem via as portas eleitas como *root ports* ou *designated ports*. Até lá as portas estão num estado que se chama *blocked* (bloqueado) em que só encaminham BPDUs. Quando estão activas, passam ao estado *forwarding* (a encaminhar *frames* de dados).

Mas como saber se o STP já estabilizou? Para mais esta decisão tem de ser tomada por cada *switch* de forma independente. Na versão IEEE 802.1D a decisão é tomada tendo em consideração a “ausência de alterações nos últimos tempos”. Durante a execução do STP as portas estão todas no estado bloqueado, i.e., sem receberem ou

enviarem *frames* diferentes dos BPDUs. Após passar um período de tempo desde que teve lugar a última modificação do estado da raiz ou de uma porta, o *switch* considera que a situação está estável e passa o estado das portas eleitas como *root* ou *designated* de *blocked* para *forwarding*. Esse período de tempo tem normalmente a duração de várias dezenas de segundos pelo que em caso de necessidade de reconfiguração, a rede fica inoperacional durante muito tempo, o que é mau.

A implementação consiste simplesmente em inicializar todas as portas no estado bloqueado, e passar todas as portas ao estado bloqueado quando se dá uma reconfiguração, uma alteração da raiz ou do caminho para a mesma, e só passar uma porta ao estado *forwarding* após esta ter estado nesse estado sem alterações durante um período de tempo significativo. Actualmente um valor típico para este compasso de espera é de 15 segundos.

Existem diversas situações que desencadeiam a execução de uma reconfiguração da rede, a qual passa por todos os *switches* se considerarem de novo raiz e começarem uma nova eleição da raiz e uma nova escolha dos caminhos mais curtos para chegar à mesma.

Se a raiz corrente tiver uma avaria ou se for desligada da rede, os outros *switches* deixam de receber os seus anúncios periódicos. Como conhecem o limite de tempo sem receberam BPDUs periódicos (através do parâmetro **Maximum Age** dos BPDUs da raiz), mudam de estado e uma nova eleição é desencadeada. A mesma começa com o primeiro *switch* que detectou a situação a enviar um BPDU pelo qual se considera raiz e com a flag **Topology Change** posicionada.

A reconfiguração da rede também pode ser desencadeada por um *switch* que detecta uma avaria do canal ligado à sua *root port*. Eventualmente, a partir de informação memorizada sobre outros anúncios recebidos, ou quando os receber, o *switch* pode mudar para outro caminho para a raiz e escolher uma nova *root port*. Se isso não for possível, o *switch* considera-se raiz e inicia o processo de eleição. Se a avaria tiver lugar numa *designated port* serão os outros *switches* que desencadearão o processo.

Um *switch* STP também só activa uma sua porta ligada a um computador, ou a toda uma zona sem STP, depois do compasso de espera que permite passar uma *designated port* ao estado de *forwarding*. Nesse caso, a ligação de um computador a uma dessas portas implica esperar vários segundos até a porta estar activa. Isto tem de ser assim pois não sabe se por detrás dessa porta está ou não outro equipamento com outras ligações que poderiam introduzir ciclos. Muitos *switches* admitem a parametrização de portas reservadas a computadores como estando fora do processo STP e portanto não sujeitas a este tipo de compassos de espera aborrecidos.

**Chama-se convergência de um protocolo de encaminhamento** ao tempo necessário para, na sequência de uma qualquer alteração de estado dos nós ou canais, todos os nós de comutação tomarem em consideração essas alterações, reconfigurarem a rede, e voltarem a colocá-la em funcionamento de acordo com a nova configuração.

O protocolo STP converge muito lentamente pois faz depender a convergência de, na sequência da deteção de uma alteração, passar um tempo de espera longo sem se produzir nenhuma nova alteração, nem as inundações periódicas, que se processam sempre ao mesmo ritmo, implicarem novas alterações.

Durante a convergência, o encaminhamento tem de ser bloqueado, porque não existem nenhum outros mecanismos (*e.g.*, TTLs ou deteção de duplicados) que estancassem uma eventual tempestade de difusão (*broadcast storm*).

### Observações finais sobre o STP

A capacidade de uma rede Ethernet gerida por STP é em geral subaproveitada pois muitos canais disponíveis têm de ficar bloqueados para que a rede tenha a configuração de uma árvore. Não é possível aproveitá-los para fornecerem um melhor caminho entre a origem e o destino, nem fazer distribuição de carga.

De facto os *frames* só seguem por “caminhos bons” enquanto esses caminhos coincidirem com os caminhos mais curtos entre a origem e o destino. Isso só se verifica em caminhos que “subam” ou “desçam” a árvore, mas não nos que incluam os dois tipos de trajectos. Por exemplo, se a maioria do tráfego for gerado de e para um servidor, idealmente este deve estar ligado à raiz da rede. O parâmetro prioridade dos *switches* permite ao gestor da rede forçar que um dado *switch* seja a raiz da rede independentemente do seu endereço MAC.

Por outro lado, se a rede tiver uma grande quantidade de computadores, serão realizadas inundações com bastante frequência, o que também não é um bom aproveitamento da sua capacidade.

No entanto, estas redes têm muitas vantagens pois são simples e geridas de forma automática, *i.e.*, com pouca ou nenhuma intervenção de um gestor. Por outro lado, quando utilizadas em áreas confinadas (dentro de um edifício ou de um grupo de edifícios) e com capacidade abundante para as necessidades, os inconvenientes assinalados têm um menor impacto.

O grande inconveniente nessa situação são os compassos de espera em caso de necessidade de reconfiguração, que levam a que a rede esteja inoperacional durante muitos segundos. Vários serviços críticos (voz, vídeo, *etc.*) não suportam essa espera. Por esta razão foi introduzida uma versão de STP, chamada Rapid STP - RSTP (norma IEEE 802.1w de 2001) que consegue reconfigurar a rede em menos de 1 segundo quando a avaria é de um canal. A norma RSTP foi incorporada na norma IEEE 802.1D na versão de 2004 que continua a ser compatível com a versão do STP descrita nesta secção.

Todos os algoritmos e protocolos que apresentámos para suporte do funcionamento de uma rede Ethernet comutada (*switched*), *i.e.*, baseada em inundações com aprendizagem pelo caminho inverso e no protocolo STP, pressupõem que todos os equipamentos funcionam de acordo com o esperado pelos protocolos e não violam ou modificam os seus endereços MAC.

Caso contrário é possível bloquear a rede enviando falsos BPDUs, ou até desviar o tráfego, por exemplo enviando periodicamente *frames* em *broadcasting* com endereços MAC origem falsos de forma a atrair para si o tráfego que lhes era dirigido. Deixada entregue a si própria, sem outros mecanismos de segurança, uma tal rede torna-se muito frágil.

## 15.4 Virtual Local Area Networks (VLANs)

O êxito das redes Ethernet comutadas permitiu montar redes destas com dezenas e dezenas de nós de comutação, abrangendo vários edifícios e servindo para a interligação de milhares de computadores. No entanto, esta abrangência levanta pelo menos dois problemas.

O primeiro tem a ver com segurança pois todos esses computadores podem endereçar-se mutuamente e, como foi referido acima, é possível, por exemplo, atacar a rede enviando *frames* em difusão com endereços MAC origem falsos para desviar o tráfego. Por outro lado, sempre que há uma inundação, as interfaces recebem *frames* que lhes

não são destinados e *frames* inúteis invadem toda a rede o que prejudica o seu funcionamento. Por isso é também possível provocar um ataque de negação de serviço enviando muitos *frames* em difusão.

É portanto mais prudente seccionar a rede em vários segmentos distintos e protegidos uns dos outros, organizando a rede em sub-redes disjuntas e independentes, uma para cada fim específico. Por exemplo, numa universidade poder-se-iam criar redes distintas para os estudantes usarem livremente, outras para os laboratórios de aulas, outras para os docentes, outras para a administração, *etc.*

Isso pode ser feito usando cabos UTP e equipamentos distintos, mas só funcionaria bem se houvesse a possibilidade de facilmente ligar computadores às diferentes redes nos mesmos locais, o que implicaria a multiplicação de cabos e equipamentos pelas diferentes áreas da universidade. O mesmo aconteceria numa empresa de grande dimensão: qualquer reorganização da distribuição dos funcionários pelos diferentes gabinetes, implicaria mexidas nos cabos e nos equipamentos da rede.

O segundo problema tem a ver com o facto de estas redes se basearem no algoritmo de inundação. Quanto maior a rede maior o número e âmbito das inundações e maior o desperdício da maioria delas. Idealmente é também desejável confinar as inundações a diferentes sub-redes.

Os *switches* Ethernet mais sofisticados incluem mecanismos de seccionamento da rede em áreas disjuntas, cada uma das quais se chama uma **rede lógica ou virtual ou VLAN (Virtual Local Area Network)**<sup>6</sup>. As VLANs são independentes entre si e não podem comunicar usando os *switches*. No entanto, para montar várias VLANs não é necessário dispor de equipamentos afectados a cada uma delas. Tudo é feito por software e pode ser reconfigurado dinamicamente em função das necessidades.

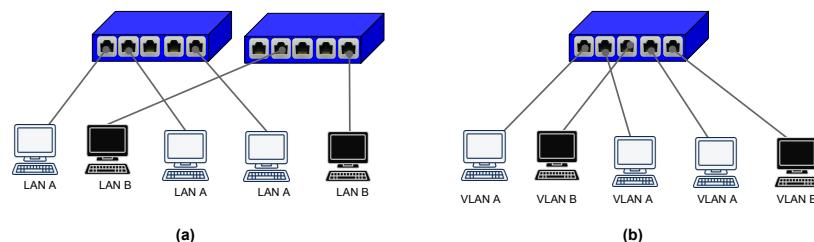


Figura 15.11: Duas redes distintas fisicamente (a) e uma implementação equivalente através de um único *switch* e duas VLANs distintas (b).

A Figura 15.11 ilustra a solução. À esquerda, parte (a), são implementadas duas redes distintas usando dois *switches*. À direita, parte (b), são implementadas duas VLANs usando um *switch* com suporte de VLANs. Cada uma das portas desse *switch* pertence a uma VLAN distinta. Isso quer dizer que qualquer *frame* recebido numa dessas portas só pode ser encaminhado (por inundação ou usando as tabelas de comutação com endereços MAC) para portas pertencentes à mesma VLAN. O mesmo se aplica a *frames* enviados para o endereço de broadcast pois o âmbito de uma inundação está sempre restrinida às portas que pertencem à VLAN do emissor.

É fácil de imaginar como implementar VLANs com um só *switch* pois este conhece de que porta um *frame* foi originado, qual a VLAN a que esta pertence, e pode restringir as portas em que é realizada a inundação às portas associadas à mesma VLAN. Mas como generalizá-lo ao conjunto da rede de tal forma que a mesma VLAN possa abranger computadores ligados a diferentes *switches*? A Figura 15.12 mostra uma

<sup>6</sup> O termo VLAN tem por origem o facto de que o nome tradicional das redes baseadas em canais de difusão se designarem por LANs - Local Area Networks.

rede física em que há computadores pertencentes à mesma VLAN ligados a diferentes *switches*. Como pode um *switch* que recebe um *frame* de outro *switch* saber a que VLAN o mesmo se destina?

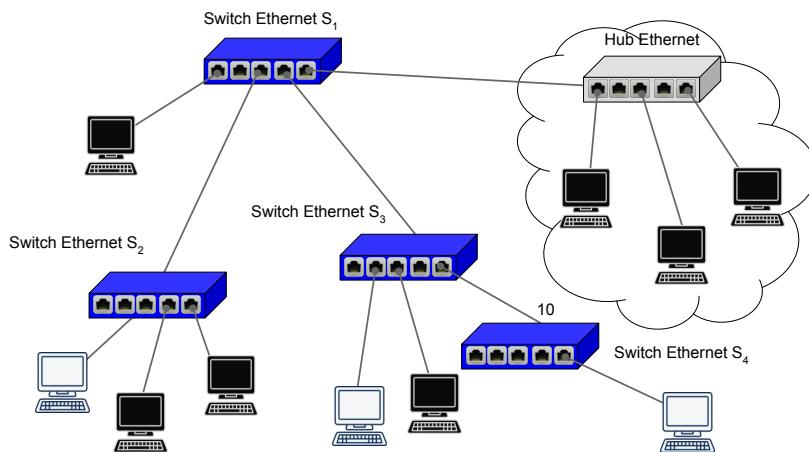


Figura 15.12: Rede com vários *switches*. Cada um deles liga computadores de duas VLANs distintas. Computadores a claro pertencem a uma VLAN, e os outros pertencem a outra VLAN. Os computadores não necessitam de “saber” o que é uma VLAN e os computadores ligados ao mesmo canal multi-ponto pertencem necessariamente à mesma VLAN.

A maneira mais simples de resolver esse problema consiste em expandir o cabeçalho IEEE 802.3 com um campo que indica a que VLAN o *frame* pertence. Esse novo cabeçalho, foi definido na norma IEEE 802.1Q de 1998 e está representado na Figura 15.13.

Esse novo campo, também designado por **marca de VLAN ou VLAN tag**, é colocado no cabeçalho de um *frame* pelo *switch* com suporte de VLANs que o recebeu pela primeira vez, e é suprimido pelos *switches* que o enviam por uma porta ligada a equipamentos que não suportam VLANs. Desta forma não é necessário inutilizar as interfaces e os equipamentos que não suportam nem conhecem VLANs. O mesmo problema já surgiu na interligação de APs IEEE 802.11 com redes IEEE 802.3. Os equipamentos de interligação têm de modificar o cabeçalho para o adaptar ao domínio para que os *frames* transitam.

Assim, as portas dos *switches* passam a ser de dois tipos, dependentes do canal que está ligado às mesmas. Se a porta apenas pode receber *frames* sem VLAN tag, então a porta está geralmente associada ela própria a uma VLAN e o *switch* coloca a tag respectiva em cada *frame* que recebe pela mesma. Por outro lado, só *frames* dirigidos a essa VLAN podem ser enviados por essa porta depois de a tag ter sido retirada. As outras portas, tipicamente as que interligam os *switches*, só podem ser atravessadas por *frames* com VLAN tags para que o destino saiba como os processar. Esses canais e portas são geralmente designados por canais de tipo *trunk* ou de interligação.

O mecanismo de aprendizagem pelo caminho inverso também pode ser expandido e, quando um *switch* recebe um *frame*, para além de colocar na tabela de endereços MAC a porta de acesso ao equipamento com esse endereço, também pode anotar a

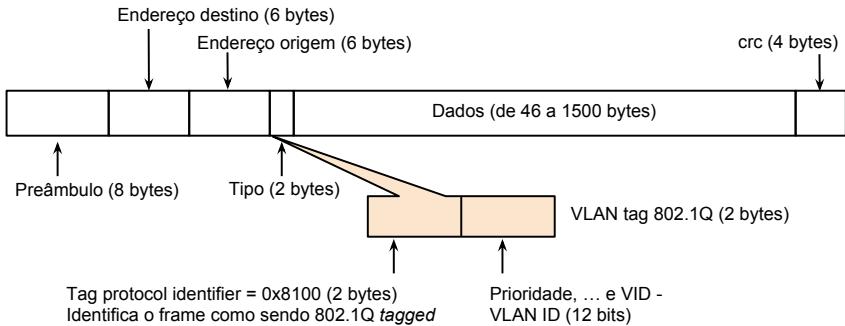


Figura 15.13: Formato dos *frames* Ethernet que incluem o cabeçalho IEEE 802.1Q. O valor que indica a presença deste foi escolhido de tal forma que permite distinguir um *frame* 802.1Q de um convencional pois é diferente de todos os valores legais do campo *Tipo*.

que VLAN o mesmo pertence. Assim, um *frame* enviado a partir de uma VLAN a que o destino não pertence, pode imediatamente ser suprimido.

As VLANs são um poderoso instrumento para o administrador de uma rede a seccionar em sub-redes distintas, isoladas por motivações de segurança, mas também para reduzir o âmbito das inundações. Adicionalmente, os *switches* mais sofisticados, para além de disporem de mecanismos de gestão de VLANs por software, também permitem parametrizar o STP para construir, ou não, árvores de cobertura distintas por VLAN. É também possível “podar” as árvores de cobertura das diferentes VLANs de forma a evitar que *switches* sem ligações a uma VLAN, recebam BPDUs referentes a essa instância de STP, ou inundações com *frames* destinados a essa VLAN.

As redes Ethernet equipadas com *switches* com suporte de VLANs ficam dotadas de uma mecanismo muito flexível que permite seccionar a rede em sub-redes distintas e isoladas, cada uma das quais associada a uma VLAN distinta.

Este mecanismo é fundamental para organizar uma rede complexa em sub-redes distintas por razões de segurança, isolamento e desempenho. Os APs com suporte de 802.11 também podem ter suporte de VLANs e separarem os utilizadores e os SSIDs por VLANs distintas.

Um domínio com VLANs inclui sempre uma VLAN especial, a VLAN 1, à qual pertencem todos os *switches* para efeitos de gestão dos mesmos. Por omissão, sem indicações contrárias, todas as portas e todos os equipamentos com suporte de VLANs pertencem à VLAN 1. Ou seja, mesmo que os *switches* suportem STP e VLANs, o administrador que não necessite das VLANs pode continuar a montar e gerir uma rede completamente auto configurável visto que o STP para a VLAN 1 está sempre activo por omissão.

## 15.5 Resumo e referências

### Resumo

O algoritmo de **encaminhamento por inundação** é muito simples: quando um nó recebe um pacote, analisa o seu endereço de destino, e se este for diferente do seu, envia uma cópia do mesmo por cada uma das suas interfaces, excepto por aquela pela qual o pacote foi recebido. Este algoritmo consegue encaminhar um pacote até ao seu destino mas, infelizmente, conduz ao envio de pacotes inúteis nas redes estruturadas em árvore, e a um fenómeno de colapso da rede designado **tempestade de difusão** (*broadcast storm*) em redes com ciclos.

O algoritmo pode ser complementado com um mecanismo baseado em números de sequência que permite detectar os pacotes duplicados e parar o processo de inundação. Apesar de este mecanismo não evitar o envio de pacotes inúteis, permite implementar **algoritmos de difusão fiável baseados em inundação**.

Numa rede estruturada em árvore, o algoritmo da inundação permite o encaminhamento de um nó para outro, assim como a sua difusão, sem necessidade de recorrer à deteção de duplicados porque estes não aparecem neste tipo de redes. Apenas serão enviados pacotes inúteis. No entanto, o mecanismo usado nas redes Ethernet comutadas conhecido como **aprendizagem ou filtragem pelo caminho inverso** (*backward learning*) com base nos endereços MAC das interfaces Ethernet dos emissores, permite restringir significativamente o envio de *frames* inúteis.

Dado um grafo arbitrário, uma árvore de cobertura do mesmo é um seu sub-grafo contendo todos os nós e que é uma árvore, *i.e.*, no qual só existe em caminho entre quaisquer dois nós. O protocolo que permite a um conjunto de *switches* Ethernet calcular uma árvore de cobertura e apenas operar de acordo com a mesma, chama-se **SPT - Spanning Tree Protocol**, norma IEEE 802.1D.

O protocolo STP baseia o seu funcionamento na inundação periódica da rede com *frames* de controlo chamados BPDUs, emitidos inicialmente por todos os *switches*, mas mais tarde apenas pelo *switch* eleito como raiz. Cada participante na inundação altera os BPDUs de forma que torna possível realizar a eleição da raiz e determinar o caminho mais curto até à mesma.

Chama-se **convergência de um protocolo de encaminhamento** ao tempo necessário para, na sequência de uma qualquer alteração de estado dos nós e canais, todos os nós de comutação tomarem em consideração essas alterações, reconfigurarem a rede, e voltarem a colocá-la em funcionamento de acordo com a nova configuração.

O protocolo STP converge muito lentamente pois faz depender a convergência de, na sequência da deteção de uma alteração, passar um tempo de espera longo sem se produzir nenhuma nova alteração, nem as inundações periódicas implicarem novas alterações. Durante a convergência o encaminhamento tem de ser bloqueado porque não existem nenhum outros mecanismos (*e.g.*, TTLs ou deteção de duplicados) que estancassem uma eventual tempestade de difusão (*broadcast storm*).

Todos os algoritmos e protocolos apresentados para suporte do funcionamento de uma rede Ethernet comutada (*switched*), *i.e.*, baseada em inundação com aprendizagem pelo caminho inverso e no protocolo STP, pressupõem que todos os equipamentos funcionam de acordo com o esperado pelos protocolos e não violam ou modificam os seus endereços MAC.

Caso contrário é possível bloquear a rede enviando falsos BPDUs, ou até desviar o tráfego, por exemplo enviando periodicamente *frames* em *broadcasting* com endereços MAC origem falsos de forma a atrair o tráfego que lhes era dirigido. Deixada entregue a si própria, sem outros mecanismos de segurança, uma tal rede torna-se muito frágil.

As redes Ethernet equipadas com *switches* com suporte de VLANs ficam dotadas de uma mecanismo muito flexível que permite seccionar a rede em sub-redes distintas e isoladas, cada uma das quais associada a uma VLAN distinta.

Este mecanismo é fundamental para organizar uma rede complexa em sub-redes distintas por razões de segurança, isolamento e desempenho. Os APs com suporte de 802.11 também podem ter suporte de VLANs e separarem os utilizadores e os SSIDs por VLANs distintas.

Alguns dos termos introduzidos no capítulo são a seguir passados em revista. Entre parêntesis figura a tradução mais comum em língua inglesa.

### Principais conceitos

**Inundação** (*flooding*) Algoritmo muito simples de encaminhamento numa rede: ao receber um pacote, um nó analisa o seu endereço de destino, e se este for diferente do seu, envia uma cópia do mesmo por cada uma das suas interfaces, excepto por aquela pela qual o pacote foi recebido. Numa rede normal este algoritmo introduz duplicados e pacote inúteis. Numa rede estruturada em árvore, este algoritmo apenas introduz pacotes inúteis que podem ser confinados usando aprendizagem pelo caminho inverso.

**Tempestade de difusão** (*broadcast storm*) Fenómeno que tem lugar numa rede a funcionar com encaminhamento por inundação caso não exista um mecanismo de detecção de duplicados, ou a rede não esteja estruturada como uma árvore. Uma tal tempestade satura e inutiliza a rede.

**Aprendizagem pelo caminho inverso** (*backward learning*) Técnica que permite minorar o tráfego inútil numa rede estruturada em árvore: se foi recebido um pacote por uma interface, com um dado endereço origem, então o nó com esse endereço está necessariamente por detrás dessa interface.

**Protocolo STP** (*Spanning Tree Protocol*) Protocolo (normalizado inicialmente pela norma IEEE 802.1D) que permite a um conjunto de *switches* Ethernet calcular uma árvore de cobertura e apenas encaminhar *frames* de acordo com a mesma. Caso haja uma avaria na rede, o STP reconfigura a árvore de cobertura.

**BPDU** (*Bridge Protocol Data Unit*) *Frames* de controlo usados pelo STP para calcular uma árvore de cobertura.

**Convergência** de um protocolo de encaminhamento é o tempo necessário para, na sequência de uma qualquer alteração de estado dos nós ou canais, todos os nós de comutação tomarem em consideração essas alterações, reconfigurarem a rede, e voltarem a colocá-la em funcionamento de acordo com a nova configuração. O STP convencional converge muito lentamente, *i.e.*, em dezenas de segundos.

**VLAN** (*Virtual Local Area Network*) Mecanismo muito flexível que permite seccionar uma rede Ethernet comutada em várias sub-redes distintas e isoladas, cada uma das quais associada a uma VLAN distinta. Os *frames* Ethernet que atravessam domínios com noção de VLAN passam a incluir um campo especial no cabeçalho (normalizado inicialmente pela norma IEEE 802.1Q) e designado por VLAN tag.

### Referências e notas históricas

Os algoritmos e tecnologias descritos neste capítulo foram introduzidos fundamentalmente pelos fabricantes de equipamentos informáticos nas então designadas redes locais (LAN - *Local Area Networks*), *i.e.*, redes confinadas a edifícios ou grupos de edifícios. O termo era usado por oposição a WAN (WAN - *Wide Area Network*), uma área tradicionalmente dominada ao nível do fornecimento de canais pelos operadores de telecomunicações. Tratou-se, no essencial, de um processo lento de resposta às necessidades das instituições que ia sendo tornado possível pela própria evolução da possibilidade de fabricar equipamentos cada vez mais sofisticados, mas de custo acessível. A procura da simplicidade foi uma constante.

O artigo original que introduziu o algoritmo mais tarde adoptado pelo protocolo STP foi publicado em 1985 por Radia Perlman [Perlman, 1985], na altura a trabalhar

num grande fabricante de computadores, a DEC - Digital Equipment Corporation. Todas as tecnologias descritas evoluíram nos laboratórios dos fabricantes e foram depois normalizadas no seio da IEEE na série de normas com o prefixo IEEE 802.

A quantidade de normas desenvolvidas para as redes Ethernet comutadas é inúmera e continua a crescer. Essa evolução inclui igualmente algumas revisões e fusions de normas. Por exemplo, na descrição da norma IEEE 802.1Q-2014 no site [www.ieee802.org](http://www.ieee802.org) pode ler-se: “This standard incorporates IEEE Std 802.1Q<sup>TM</sup>-2011, IEEE Std 802.1Qbe<sup>TM</sup>-2011, IEEE Std 802.1Qbc<sup>TM</sup>-2011, IEEE Std 802.1Qbb<sup>TM</sup>-2011, IEEE Std 802.1Qaz<sup>TM</sup>-2011, IEEE Std 802.1Qbf<sup>TM</sup>-2011, IEEE Std 802.1Qbg<sup>TM</sup>-2012, IEEE Std 802.1aq<sup>TM</sup>-2012, IEEE Std 802.1Q<sup>TM</sup>-2011/Cor 2-2012, and IEEE Std 802.1Qbp<sup>TM</sup>-2014. The standard includes much functionality previously specified in 802.1D.” Ou seja, a versão de 2014 da norma IEEE 802.1Q incorpora todas as funcionalidades do STP, Rapid STP, Multiple Spanning Trees, VLANs, Per VLAN Spanning Trees, etc.

Com o aparecimento de canais Ethernet *full-duplex* de longa distância baseados em fibras ópticas, e *switches* de alta capacidade, estas redes deixaram de se confinhar ao espaço das LANs (termo que começou por isso a cair em desuso), e passaram a ser usadas também pelos operadores de longa distância, por exemplo para fornecerem serviços de interconexão de redes Ethernet com base em tecnologias Ethernet. Quando estas encaminham *frames* Ethernet de clientes que já usam VLANs, necessitam de preservar as VLANs dos clientes. Isso levou à introdução de um campo de VLAN *tagging* que inclui duas VLANs, a do cliente e a do operador. Por outro lado, o campo tradicional com o número de VLAN apenas permite cerca de 4000 VLANs distintas, um número insuficiente para um operador. Por isso este campo também teve necessidade de crescer.

As VLANs também são usadas nas redes de centros de dados para isolar os diferentes clientes que utilizam o centro de dados. O número de diferentes VLANs disponíveis também se tornou demasiado curto.

A entrada da Ethernet na longa distância e nas redes de operadores, assim como nos centros de dados de grande dimensão, tornou impossível continuar a adoptar tecnologias de encaminhamento sem preocupações de optimização da rede e dos caminhos seguidos pelos pacotes. Nesta sequência, começam a ser adoptadas para as redes Ethernet os mesmos algoritmos e técnicas de encaminhamento desenvolvidos para as redes gerais, que começaremos a estudar no próximo capítulo.

Esta linha de evolução está a materializar-se em torno de duas propostas, uma designada TRILL (Transparent Interconnection of Lots of Links) que foi desenvolvida no seio do IETF, ver [Perlman and Eastlake, 2011] e o RFC 6362, e a outra designada Shortest Path Bridging [Luo and Suh, 2011], ou norma IEEE 802.1aq. Ambas visam substituir as árvores de cobertura por alternativas mais sofisticadas. Estas propostas estão igualmente a ser desafiadas pelo aparecimento das Software Defined Networks, que são uma forma alternativa de melhor gerir redes de comutadores de pacotes e que será também apresentada nos capítulos seguintes.

Como diria um conhecido autor (A. S. Tanenbaum): “a beleza das normas é que há sempre muitas por onde escolher”.

### Apontadores para informação na Web

- <http://www.ieee802.org> – É o site oficial das normas IEEE 802.
- <http://ietf.org/rfc.html> – É o repositório oficial dos RFCs, nomeadamente dos citados neste capítulo.

### 15.6 Questões para revisão e estudo

1. Uma rede Ethernet comutada (*switched*) organizada em árvore usa o algoritmo de encaminhamento por inundação com filtragem baseada em aprendizagem pelo caminho inverso.
  - (a) Nesta rede é necessário existir um mecanismo de detecção de duplicados?
  - (b) Admitindo que o endereço MAC de destino *Addr* é conhecido de todos os *switches*, o encaminhamento de um *frame* dirigido a *Addr* é realizado usando cópias inúteis?
2. Considere o algoritmo de encaminhamento de pacotes por inundação (*flooding*). Qual ou quais das seguintes afirmações são verdadeiras no contexto desse algoritmo de encaminhamento?
  - (a) O algoritmo de inundação implementa naturalmente a difusão de mensagens para todos os computadores ligados à rede.
  - (b) O algoritmo de encaminhamento por inundação é muito ineficiente e por isso nunca é usado na prática
  - (c) Uma maneira de suprimir a introdução de todos os pacotes duplicados através de inundação numa rede estruturada em malha (com ciclos) é usar um TTL adequado.
  - (d) Numa rede estruturada em árvore, a inundação encaminha sempre pelo melhor caminho a primeira cópia a chegar ao destino.
  - (e) Numa rede totalmente conexa (cada nó tem pelo menos um caminho para qualquer um dos outros) o algoritmo de inundação assegura que todas as mensagens chegam a todos os nós desde que não existam erros de transmissão.
  - (f) Caso a topologia da rede não tenha ciclos, a inundação é um processo razoavelmente eficiente de implementar o encaminhamento porque nunca enviará mensagens em duplicado, nem mensagens inúteis.
3. Indique uma ou mais razões para associar um TTL a cada endereço MAC registado na tabelas de endereços dos *switches* que realizam aprendizagem pelo caminho inverso.
4. Num dado *switch* constatou-se que a porta pela qual este recebeu um *frame* com endereço origem A foi a porta P2, mas 2 minutos atrás tinha sido a porta P1. Que poderá explicar esta situação?
5. Num dado *switch* constatou-se que a porta pela qual este recebeu um *frame* com endereço origem A foi a porta P2, mas 100  $\mu$  segundos atrás tinha sido a porta P1. Que poderá explicar esta situação?
6. Numa rede Ethernet comutada (*switched*) não podem existir ciclos pois a presença dos mesmos introduziria uma situação designada por tempestade de difusões (*broadcast storm*). Explique em que consiste esta situação e dê uma sugestão de como a mesma poderia ser detectada pelos *switches*.
7. Numa rede Ethernet comutada os *switches* dispõem de tabelas de endereços MAC.
  - (a) Para programar essas tabelas qual a estrutura de dados que escolheria?
  - (b) Porque razão, ou razões, é possível a dimensão das tabelas ser inferior ao número de diferentes endereços vistos pelo *switch*?

- (c) Indique um mecanismo particularmente eficaz na ajuda da convergência das tabelas dos *switches* que um computador pode usar sempre que muda de porta, i.e., sempre que a sua interface é ligada e depois se volta a ligar.
- (d) Indique uma forma de um computador conduzir um ataque de negação de serviço a uma rede deste tipo (mais eficaz no caso em que há mais interfaces do que entradas nas tabelas de comutação). O ataque tem por consequência que o tráfego dos computadores normais passa a ser encaminhado por inundação pois o mecanismo de aprendizagem deixa de ser eficaz.
- (e) Idealize uma ou mais formas de combater este tipo de ataques.
8. 1000 computadores estão ligados a uma rede comutada (*switched*) Ethernet com 3 comutadores (*switches*) Ethernet. Estes equipamentos são idênticos a menos do seguinte aspecto: os comutadores A e B têm uma tabela de endereços MAC com 32 entradas e o comutador C tem uma tabela de endereços MAC com 1024 entradas. Após fazer medidas, você verificou que os comutadores A e B realizavam mais inundações que o comutador C. Que justifica a diferença observada?
9. Considere a rede Ethernet comutada (*switched*) da Figura 15.14. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. O computador com o endereço  $MAC_1$  enviou um *frame* em difusão, seguido de um *frame* dirigido ao computador com o endereço  $MAC_6$  e este enviou um *frame* em difusão seguido de um *frame* dirigido ao computador com o endereço  $MAC_1$ .

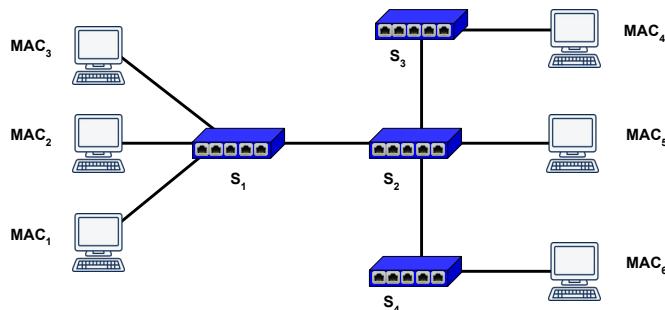


Figura 15.14: Rede de comutadores Ethernet do exercício 9.

- (a) No final desta troca de *frames* quais os endereços MAC que o comutador  $S_2$  passou a conhecer?
- (b) No final desta troca de *frames* quais os endereços MAC que o comutador  $S_3$  passou a conhecer?
10. Considere a rede Ethernet comutada (*switched*) da Figura 15.15. Cada computador é designado pelo seu endereço MAC. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. O seguinte conjunto de comunicações teve lugar: primeiro o computador  $MAC_1$  enviou o *frame*  $f_1$  ao computador  $MAC_8$ ; depois o  $MAC_9$  enviou o *frame*  $f_9$  ao

computador  $MAC_3$ ; depois o computador  $MAC_8$  enviou o *frame*  $f_8$  ao computador  $MAC_1$ ; e finalmente o computador  $MAC_3$  enviou o *frame*  $f_3$  ao computador  $MAC_9$ . Quais dos diferentes *frames* enviados chegaram à interface do computador  $MAC_4$ ?

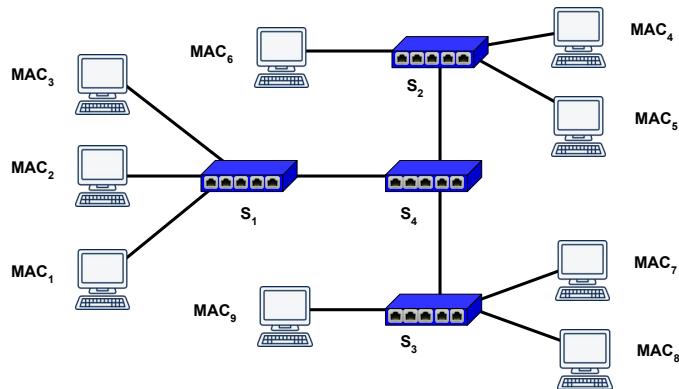


Figura 15.15: Rede de comutadores Ethernet do exercício 10.

11. Considere uma rede Ethernet comutada (*switched*) com a configuração da Figura 15.16, em que os canais têm o débito indicado em cada um. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. Comece por indicar os custos de cada canal, desenhe a árvore calculada pelo protocolo STP e indique o custo com que cada *switch* vê a raiz escolhida.

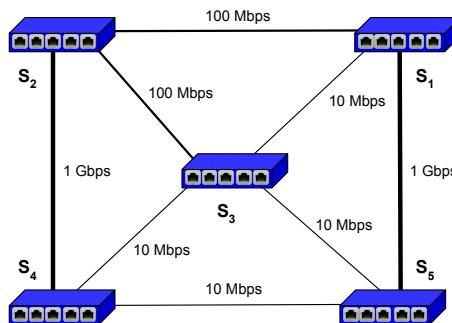


Figura 15.16: Rede de comutadores Ethernet do exercício 11.

12. Considere a rede Ethernet comutada (*switched*) da Figura 15.17. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*.
- Comece por indicar o custo de cada canal, desenhe a árvore calculada pelo protocolo STP e indique o custo com que cada *switch* vê a raiz escolhida.

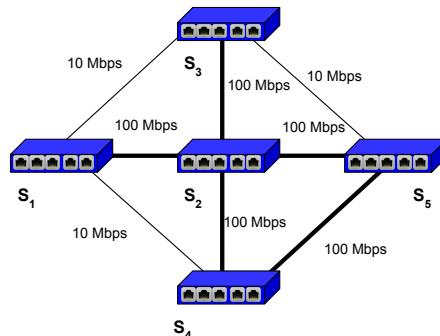


Figura 15.17: Rede de comutadores Ethernet do exercício 12.

- (b) Dada a árvore calculada pelo protocolo STP dê o exemplo de 2 caminhos ótimos (do ponto de vista do custo extremo a extremo) e o exemplo de um caminho não ótimo seguido pelos *frames* na rede através da árvore calculada. Indique cada caminho como uma sequência de identificadores de *switches*.
- (c) Por hipótese, o *switch*  $S_1$  passou a não suportar o protocolo STP e ignora os seus *frames*, tratando-os como todos os outros, isto é, propagando-os normalmente por todas as interfaces visto que o seu endereço de destino é o endereço de *broadcast*. Calcule a nova árvore calculada pelo STP nesse caso e indique o custo com que cada *switch* vê a nova raiz escolhida.
- (d) Qual é a *Designated Port* do canal multi-ponto que passou a existir na rede?
13. Considere uma rede Ethernet comutada (*switched*) com a configuração da Figura 15.18, em que os canais têm todos o débito de 100 Mbps e os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. As portas dos *switches* estão numeradas de tal forma que as portas dos canais verticais têm sempre um número inferior ao número das portas a que estão ligados os canais horizontais. Nessa rede, existe um único servidor que está ligado ao *switch*  $S_5$ . Associado a cada um dos outros *switches* existe um cliente.
- Comece por indicar o custo de cada canal, desenhe a árvore calculada pelo protocolo STP e indique o custo com que cada *switch* vê a raiz escolhida.
  - Admita que a rede é reconfigurada e o *switch*  $S_5$  e o *switch*  $S_1$  trocam de posição, mas mantendo-se o servidor no centro (agora ligado ao *switch*  $S_1$ ). Calcule a nova árvore.
  - O servidor tem de transferir um ficheiro de 1 Gbyte para cada um dos clientes por TCP, abrindo uma conexão para cada um deles. Em qual das configurações a transferência acabaria mais depressa? (isto é, todos os clientes recebiam a sua cópia). Justifique a sua resposta apresentando argumentos rigorosos e baseados em factos, ou seja, mostrando que num caso, algumas conexões TCP terminariam necessariamente mais tarde a transferência do que no outro.
14. Considere uma rede com a configuração da Figura 15.19, em que os canais têm todos o mesmo débito. Os endereços MAC dos *switches* estão ordenados de

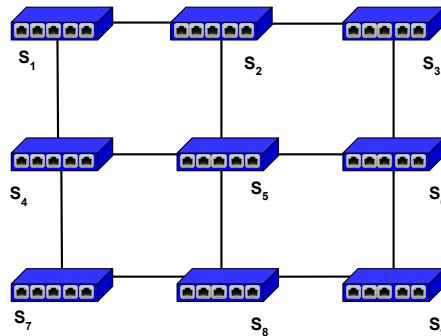


Figura 15.18: Rede de comutadores Ethernet do exercício 13.

forma compatível com a ordem dos números de cada *switch*. Por baixo de cada *switch* estão indicados os números (IDs) das portas. Diga qual dos *switches* era seleccionado pelo STP como raiz e quais os canais activos e quais os bloqueados.

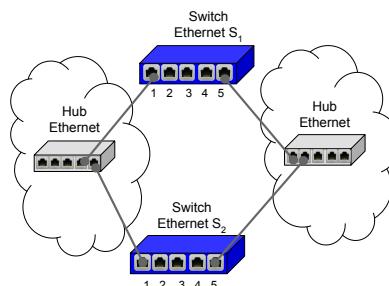


Figura 15.19: Rede de comutadores Ethernet do exercício 14.

15. Considere uma rede Ethernet comutada (*switched*) com a configuração da Figura 15.20, em que os canais têm o mesmo débito. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. Qual a árvore de cobertura escolhida pelo protocolo STP?
16. Considere uma rede Ethernet comutada (*switched*) com a configuração da Figura 15.21 em que os canais têm os débitos indicados. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*.
  - (a) Comece por indicar o custo de cada canal, desenhe a árvore calculada pelo protocolo STP e indique o custo com que cada *switch* vê a raiz escolhida.
  - (b) Na mesma rede são usadas várias VLANs distintas, cada uma das quais espalhada pela maioria dos *switches*. Existe alguma desvantagem em não usar uma única STP comum a todas as VLANs?
  - (c) Na mesma rede existe uma VLAN confinada aos *switches*  $S_1$  e  $S_3$ . Existe alguma desvantagem em usar uma única STP que cubra apenas esses dois *switches*?

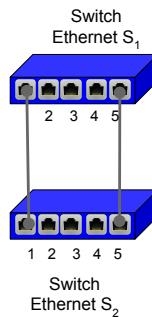


Figura 15.20: Rede de comutadores Ethernet do exercício 15.

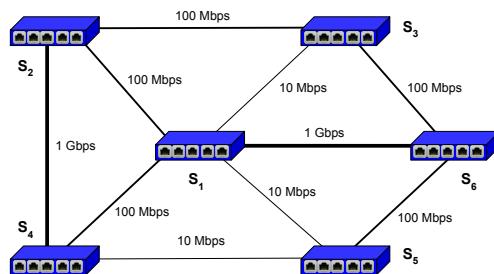


Figura 15.21: Rede de comutadores Ethernet do exercício 16.

17. Considere uma rede Ethernet comutada (*switched*) com a configuração da Figura 15.22, em que os canais a cheio têm o débito de 100 Mbps ou os a tracejado de 10 Mbps. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. Os *switches* estão parametrizados para admitirem várias VLANs e com uma versão do protocolo STP que suporta árvores distintas por cada VLAN. Nas portas de acesso dos *switches* para ligação de computadores estão presentes as seguintes VLANs:  $S_1$  a  $S_3$  — VLANs 1 e 100;  $S_4$  a  $S_6$  — VLANs 1 e 200 e  $S_7$  a  $S_9$  — VLANs 1, 100 e 200.
- Os *switches* aplicam filtros às portas de *trunking* de forma a que os *frames* (incluindo os do protocolo STP) permitidos em entrada ou saída pela porta só podem pertencer às VLANs a que os *switches* dão acesso e os *switches* desconhecem completamente as VLANs para os quais não têm portas.
- Comece por indicar o custo de cada canal, desenhe a árvore calculada pelo protocolo STP e indique o custo com que cada *switch* vê a raiz escolhida.
18. Considere uma rede Ethernet comutada (*switched*) com a configuração da Figura 15.23, em que os canais mais grossos têm o débito de 100 Mbps e os mais finos 10 Mbps. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. O tráfego principal na rede é o que circula entre os *switches* 7, 8 e 9, sendo o tráfego entre os restantes *switches* muito inferior. A rede tem um erro na sua concepção. Proponha uma solução para o mesmo tendo em consideração que só pode alterar a parametrização do STP.
19. Considere a rede da Figura 15.24 baseada num conjunto de troços Ethernet *full-*

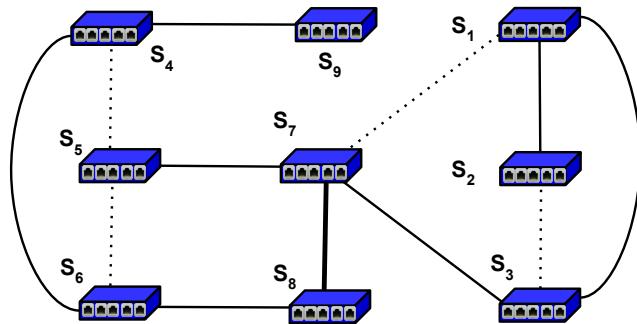


Figura 15.22: Rede de comutadores Ethernet do exercício 17.

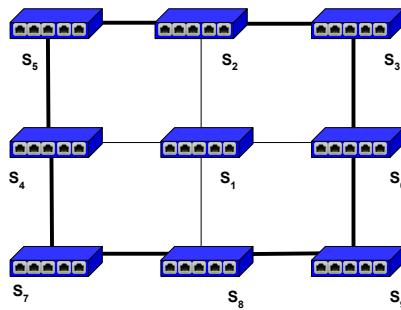


Figura 15.23: Rede de comutadores Ethernet do exercício 18.

*duplex* (a 100 Mbps os a traço forte e a 10 Mbps os a traço fino) interligando um conjunto de *switches* que executam o protocolo STP. Os endereços MAC dos *switches* estão ordenados de forma compatível com a ordem dos números de cada *switch*. A rede é complementada por um *hub* cujas portas interligam os *switches* indicados através das ligações ponteadas com o débito de 10 Mbps. Desenhe a árvore de cobertura determinada pelo protocolo STP.

20. As mensagens do protocolo STP têm os campos indicados na Figura 15.8.
- Indique quais desses campos são importantes para um *switch* calcular qual a sua *Root Port*.
  - O protocolo STP usa um temporizador chamado *Forward Delay*. Diga para que serve e qual a influência do seu valor sobre a qualidade do protocolo quando um canal ou um *switch* têm uma avaria.
  - Descreva como poderia um gestor da rede tentar que a convergência do STP fosse melhorada em caso de avaria de um canal, baixando as temporizações dos diferentes alarmes usados pelo protocolo, cujos valores por omissão são elevados. Por exemplo: *Forward Delay* = 15 segundos, *Hello Time* = 2 segundos e *Max age* = 6 segundos. Suponha que o maior caminho na rede tem  $K \leq 20$  hops, que cada canal tem um tempo de propagação inferior a 1 ms e uma capacidade de pelo menos 10 Mbps. Repare que mesmo

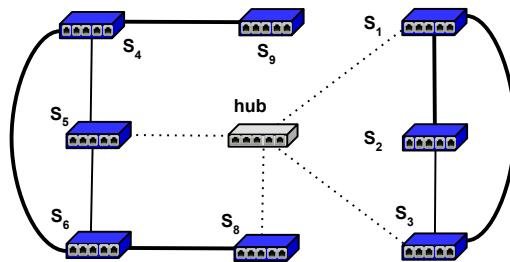


Figura 15.24: Rede de comutadores Ethernet do exercício 19.

havendo a possibilidade de se perderem BPDUs por erros nos canais, a probabilidade de tal suceder é muito baixa.

21. Quando um *switch* Ethernet recebe um *frame* instancia-o num objecto da classe *EtherFrame*, a qual tem os seguintes métodos:

```
void send(Port port)    // envia o frame pela porta port
void flood(Port port)   // inunda todas as portas com o frame excepto port
MACAddress getOrigin()  // devolve o endereço origem
MACAddress getDestination() // devolve o endereço destino
```

O *switch* tem também uma tabela (*macTable*) de endereços MAC:

```
Map< MACaddress, Port > macTable =
    new HashMap < MACaddress, Port > ();
```

com os métodos convencionais *put* e *get*.

O *switch* acabou de receber um *frame* *f* pela porta *p*. Escreva o pseudo-código necessário para o processar:

```
processFrame ( EtherFrame f, Port p ) { }
```

22. Numa rede Ethernet comutada (*switched*) especial são usados *frames* em cujo cabeçalho figuram também um TTL e um número de sequência colocado pelo emissor. A ideia é desta forma detectar duplicados e poder usar o algoritmo de inundação numa rede arbitrária, com eventuais ciclos. Quando um *switch* Ethernet recebe um *frame* instancia um objecto da classe *SpecialEtherFrame*, a qual tem os seguintes métodos:

```
void send(Port port)    // envia o frame pela porta port
void flood(Port port)   // inunda todas as portas com o frame excepto port
MACAddress getOrigin()  // devolve o endereço origem
MACAddress getDestination() // devolve o endereço destino
int getSequence() // devolve o número de sequência
void decreaseTTL() // decrementa o TTL
boolean expiredTTL() // true se TTL <= 0
```

```
int getTTL() // devolve o TTL do frame
void setTTL(int t) // fixa o TTL do frame
```

O *switch* tem uma tabela de endereços MAC, a tabela `macTable`:

```
Map< MACaddress, Port > macTable =
    new HashMap < MACaddress, Port > ();
```

com os métodos `put` e `get` como é convencional. O *switch* tem também uma tabela (`frameTable`) de endereços MAC origem e números de sequência com os métodos:

```
void register( SpecialEtherFrame f);
// regista o frame f na tabela
boolean isDuplicate( SpecialEtherFrame f);
// verifica se f já foi registado
```

O *switch* acabou de receber um *frame* *f* pela porta *p*. Escreva o pseudo-código necessário para o processar:

```
processFrame ( SpecialEtherFrame f, Port p ) { }
```

23. Numa hipotética rede Ethernet comutada (*switched*), organizada em malha, *i.e.*, com vários ciclos e caminhos redundantes, utiliza-se um protocolo inspirado do STP para calcular tantas árvores de cobertura quantos os *switches* existentes. Cada uma das árvores tem raiz num *switch* diferente.
  - (a) Sugira as modificações a introduzir no protocolo STP para que isso seja possível.
  - (b) Para encaminhar os *frames* é também usada uma variante do algoritmo normal. Essa variante consiste em um frame *frame f* ser sempre encaminhado pela árvore com raiz no *switch* em que *f* entrou na rede. Para que isto seja possível, teriam de ser realizadas modificações no cabeçalho dos *frames* Ethernet? Se sim sugira uma hipótese.
  - (c) Qual o custo do encaminhamento de um *frame* Ethernet nesta variante? Compare-o sucintamente com os custos de encaminhamento numa Ethernet comutada com uma única árvore de cobertura.

## ***Capítulo 16***

---

### ***Encaminhamento pelo caminho mais curto***

---

*Make everything as simple as possible, but not simpler.*

– Autor: *Albert Einstein (1879-1955)*

O encaminhamento com base em inundação é uma aplicação directa do princípio KISS (*Keep It Simple, Stupid*), i.e., trata-se de uma solução tão simples quanto possível. Essa solução é adequada quando os requisitos de uma solução de encaminhamento não são demasiado exigentes, nomeadamente: a rede tem uma elevada capacidade face às necessidades, o custo dos canais é suficientemente baixo para se poderem ter canais de reserva sem serem utilizados, o âmbito geográfico da rede não é elevado pelo que o impacto do tempo de trânsito de extremo a extremo também pode ser ignorado, e é aceitável realizar a inundação de toda a rede com pacotes desde que o rácio entre inundações e encaminhamento ponto-a-ponto seja baixo.

No entanto, caso a rede tenha um elevado âmbito e o número de computadores ligados à mesma seja muito elevado, esta solução deixa de ser realista pois os canais de longa distância são dispendiosos e deixa de ser viável sobre-dimensioná-los. Adicionalmente, caso a rede tenha de cobrir muitas cidades, deixa de ser aceitável realizar frequentemente inundações e também seria um desperdício inaceitável não usar os melhores caminhos para atingir o destino. Cobrir um grande número de cidades com uma rede estruturada em árvore só é realista com distâncias geográficas curtas. Finalmente, com várias dezenas ou centenas de milhar de interfaces distintas, também seria catastrófico para o desempenho provocar uma inundação cada vez que cada uma fosse activada ou endereçada pela primeira vez. Em resumo, a inundação, apesar de simples, só é uma solução viável em casos particulares.

No caso mais geral, é necessário usar soluções mais sofisticadas que permitam responder a vários constrangimentos e que tornem possível satisfazer o maior número possível de clientes, com o menor custo possível. Ou seja, no caso geral, o encaminhamento envolve facetas de correção (os pacotes têm de chegar ao destino) mas também de optimização: quer no que diz respeito a cada fluxo de pacotes em particular, quer no que diz respeito à operação global da rede, minimizando os seus custos, ao mesmo tempo que se maximiza a qualidade e quantidade do serviço prestado. Por outro lado, como a rede é formada por muitos nós e canais, em caso de avaria de alguns é necessário mantê-la operacional, reconfigurando-a rápida e dinamicamente se necessário for.

O problema do encaminhamento de pacotes é um dos problemas centrais das redes de computadores e também um dos mais complexos, pois envolve várias facetas e requisitos:

**teóricas** para modelização e optimização da rede;

**algorítmicas** para concretizar essas soluções;

**de engenharia** para implementar um sistema complexo que responda aos requisitos e minimizar o seu custo;

**de monitorização** para obter dados sobre o seu funcionamento real;

**operacionais e de planeamento** para responder às necessidades do dia a dia e adaptar a rede à continua evolução dos requisitos que lhe são colocados.

Assim, existem inúmeras soluções de gestão do encaminhamento que são função de requisitos específicos e das condições concretas da rede. Neste capítulo vamos estudar uma solução relativamente geral e bem conhecida que é designada por encaminhamento pelo “caminho mais curto” (*shortest path routing*). Neste quadro, usa-se o conceito de distância como forma de medir a adequação de um caminho.

O capítulo começa por enquadrar o problema do encaminhamento e explicar por que razão encaminhar pelo caminho mais curto é uma solução adequada em muitos contextos. Em seguida apresenta um algoritmo para determinar o caminho mais curto entre quaisquer dois nós de uma rede. Finalmente, apresenta várias soluções de engenharia para concretização desta forma de encaminhamento e faz referência às suas concretizações em termos de protocolos de encaminhamento normalizados.

## 16.1 Encaminhamento: o problema e uma solução

À primeira vista, o problema do encaminhamento de pacotes é simples de enunciar: quando se pretende encaminhar um pacote de  $x$  para  $y$ , ou quando um nó de comutação  $x$  pretende enviar um pacote para o destino  $y$ , que caminho escolher? No entanto, esta simplicidade esconde alguma complexidade e questões suplementares que também são importantes.

Se o caminho variar de pacote para pacote, pode ser necessário manter a estabilidade do encaminhamento do conjunto dos pacotes relacionados com a mesma conexão, variando potencialmente o caminho apenas de conexão para conexão. Se na rede se pretende dar diferentes qualidades de serviço a diferentes aplicações ou a diferentes clientes, pode ser necessário tomar em consideração não só a origem e o destino, mas também quais são as aplicações e os clientes envolvidos. Dependendo de diversos factores, o melhor caminho pode ser o com o maior débito de extremo a extremo, o com o menor tempo de trânsito de extremo a extremo, o com o menor custo monetário, ou ainda o que for mais fiável (*i.e.*, que perder menos pacotes).

Se se pretender optimizar a utilização dos canais disponíveis para aumentar a quantidade de tráfego por unidade de tempo que a rede encaminha, os caminhos podem ter de ser adaptados em função da carga da rede para evitar a saturação de alguns canais, ao mesmo tempo que outros estão sub-aproveitados. Isto é, caso a rede disponha de caminhos alternativos, será também desejável poder explorá-los para realizar **distribuição de carga**.

Assim, para além dos critérios relacionados com a finalidade propriamente dita da rede, que impõem que o encaminhamento seja **correcto**, *i.e.*, que os pacotes chegam

ao seu destino, é também necessário considerar critérios de **qualidade do serviço** mas também de **optimização da rede**.

### Optimização

Com efeito, a solução deste problema vai condicionar a qualidade de serviço oferecida pela rede aos seus utilizadores. Por exemplo: como diminuir o tempo de trânsito? Como maximizar a utilização da rede de forma a servir o melhor que possível o maior número possível de utilizadores? Como manter a equidade entre os diferentes utilizadores? Como dar garantias de serviço aos utilizadores (em termos do tempo de trânsito, da taxa de perca de pacotes, do débito mínimo de extremo a extremo)?

### Problemas operacionais

Adicionalmente, é também necessário tomar em consideração requisitos de natureza operacional. Assim, é necessário encontrar soluções que permitam **responder automaticamente e dinamicamente a avarias**. Ou seja, caso alguns caminhos deixem de ser possíveis porque alguns nós ou canais avariaram repentinamente, deve ser possível escolher caminhos alternativos para manter o encaminhamento dos pacotes que atravessavam anteriormente as zonas com avarias.

É possível os arquitectos da rede fazerem uma análise detalhada da mesma, esco-lherem os diferentes caminhos e parametrizarem os nós de comutação de pacotes para realizarem essa comutação de forma **estática**, no entanto, se a rede for de dimensão apreciável, é preferível dispor de um sistema que automaticamente se adapte à evolução da mesma, quer de curto prazo, porque ocorreram avarias, quer de médio prazo porque a rede cresceu para servir mais nós que se juntaram à mesma. Assim, a solução para o encaminhamento deve envolver a possibilidade de se adoptarem soluções de **encaminhamento dinâmico**.

### Uma solução de compromisso: encaminhamento pelo caminho mais curto

Existe uma solução de compromisso cuja realização permite responder a um conjunto alargado dos requisitos anteriores. Essa solução consiste em encaminhar os pacotes pelo caminho mais curto entre a origem e o destino.

Com efeito, o **encaminhamento pelo caminho mais curto (*shortest path routing*)<sup>1</sup>** tenta optimizar o serviço prestado a cada pacote em particular, e daí a cada fluxo e a cada cliente. Enquanto a rede não tiver canais saturados, tudo indica que esta solução é aceitável.

No fundo, ela é equivalente à que é praticada numa rede viária fora das horas de ponta: o caminho escolhido pelos automobilistas é o caminho mais curto desde a origem até ao destino. Caso não existam engarrafamentos, não é necessário recorrer a caminhos alternativos mais longos, por os mais directos estarem engarrafados.

Se conseguirmos implementar esta solução de tal forma que os outros requisitos operacionais sejam satisfeitos, dispomos de uma solução razoável que apenas coloca de parte as questões de optimização global da rede e as de diferenciação da qualidade de serviço.

Mas em que consiste um caminho mais curto? Numa rede viária o caminho mais curto é o que encurta a distância geográfica. Numa rede de computadores, por omissão, o caminho mais curto é o que encurta o tempo de trânsito de extremo a extremo dos pacotes. Esse tempo é função do débito dos canais atravessados, do tempo de propagação dos mesmos e até da sua carga, visto que esta influencia a dimensão das filas de espera. Voltaremos a este ponto mais tarde, mas por agora podemos abstrair esta característica de um canal através de uma função, chamada custo, que devolve

---

<sup>1</sup> A noção de “mais curto” depende da forma como se calcula a distância, pelo que esta noção pode acomodar as várias variantes de melhor caminho acima enunciadas.

um valor estritamente maior que zero. O custo de um caminho será a soma do custo dos canais que este atravessa. Quanto “pior” o canal, maior deve ser o seu custo.

Por isso, rigorosamente, este encaminhamento deveria chamar-se encaminhamento pelo caminho de menor custo. No entanto, a tradição dita que se use a terminologia característica de se considerar o custo equivalente à distância pois foi dessa forma que o problema foi originalmente enunciado em teoria dos grafos.

O problema do encaminhamento numa rede de comutação de pacotes é um problema fundamental das redes de computadores que envolve facetas de modelização e optimização, algorítmicas, de engenharia para sua concretização, operacionais, e de planeamento e gestão.

Uma primeira solução de compromisso relativamente comum consiste em usar encaminhamento pelo caminho mais curto. **Esta solução é aceitável quando os caminhos mais curtos estão bem dimensionados para o tráfego existente.**

## 16.2 Determinação de caminhos mais curtos

Num grafo, um algoritmo de selecção de um caminho mais curto do nó  $a$  para o nó  $b$  seleciona um caminho cujo custo é mínimo entre todos os caminhos simples (sem ciclos) de  $a$  para  $b$ . Com efeito, pode existir mais do que um caminho mais curto. O algoritmo que vamos apresentar deve-se a E. W. Dijkstra. Dado um nó origem  $orig$ , este algoritmo calcula caminhos mais curtos para todos os outros nós do grafo. Esse conjunto de caminhos forma uma árvore de cobertura de caminhos mais curtos com raiz em  $orig$ .

Neste contexto, a rede é definida por um grafo  $G = (N, E)$ , onde  $N$  é o conjunto dos vértices ou nós e  $E$  é o conjunto dos arcos (*edges*). Seja  $n$  o número de nós em  $N$ , i.e.,  $n = \#N$ . O grafo  $G$  é *simples* (um grafo sem lacetes nem arcos paralelos), *orientado* (onde  $(c, d)$  e  $(d, c)$  denotam arcos distintos) e *pesado*, sendo o peso ou custo de cada arco estritamente positivo, i.e.,  $\forall e \in E, cost(e) > 0$ . Todos os caminhos em que estamos interessados são *simples*, i.e., todos os nós de um caminho são diferentes, o que é equivalente a o caminho não ter ciclos.

Neste modelo, os nós do grafo modelizam os computadores e os nós de comutação de pacotes. Os arcos modelizam os canais. O custo de um arco representa o custo do canal. Para não perder generalidade, considera-se que cada computador e cada nó de comutação podem ser origem ou destino de pacotes.

Seja  $orig \in N$  o nó origem, a raiz de uma árvore de cobertura de caminhos mais curtos que se pretende calcular. No contexto do algoritmo, o custo de um nó  $x$  para outro nó  $y$  é  $cost(x, y)$  se existe o arco  $(x, y) \in E$  e  $\infty$  no caso contrário. Dado um nó  $x$ , a distância de  $orig$  a  $x$  ( $distance(x)$ ) é definida como sendo a soma dos custos dos arcos de um caminho mais curto entre  $orig$  e  $x$ . Esta distância corresponde ao custo de encaminhar tráfego de  $orig$  até  $x$  por esse caminho. Por definição, a distância do nó  $orig$  a si próprio é nula, i.e.,  $distance(orig) = 0$ .

O algoritmo usa o conjunto de nós  $S$  ( $S$  de *Stable*) para os quais já se conhece um caminho mais curto desde  $orig$ . Inicialmente  $S = \{orig\}$ . Em cada iteração do algoritmo, um novo nó  $x$  é seleccionado, de tal forma que o caminho mais curto entre  $orig$  e  $x$  foi determinado.  $x$  é acrescentado a  $S$  e passará a ser a base para determinar o novo nó a incluir em  $S$  na próxima iteração. Por esta razão, chamaremos a esse nó o nó *pivot* em cada iteração. A chave do algoritmo consiste em seleccionar o *pivot* de tal forma que é garantido que não existe nenhum outro caminho mais curto para

Listing 16.1: Pseudo código simplificado do algoritmo de Dijkstra para cálculo de caminhos mais curtos

```

1 // Input:
2 //      N - set of network nodes, n = #N
3 //      E - set of network edges (links) with costs C
4 //      orig - origin node
5 // Local variables:
6 //      S - set of nodes for which a shortest path is already known
7 //      pivot - a pivotal node for which a shortest path is known
8 // Output:
9 //      distance int[1 .. n] - distance to orig
10 //      predecessor node[1 .. n] - predecessor in the
11 //          selected shortest path to orig
12
13 S = {orig}
14 for each (node x in N) {
15     // if (x,y) not in E, cost(x,y) = infinity
16     distance[x]=cost(orig,x)
17     if ( (orig,x) in E ) predecessor[x] = orig
18     else predecessor[x] = null
19 }
20 distance[orig] = 0
21 predecessor[orig] = orig
22 pivot = orig
23 while ( S != N ) {
24     find node x in N : x not in S and distance(x) is minimal
25     pivot = x
26     S += pivot // since there is no shorter path to pivot
27     // then update the distance for all nodes y not in S
28     // that are directly connected to pivot
29     for each (node y in N : y not in S and (pivot,y) in E) {
30         d = min ( distance(y), distance(pivot)+cost(pivot,y) )
31         if ( d < distance(y) ) {
32             // we found a better alternative
33             distance(y) = d
34             predecessor(y) = pivot
35         }
36     }
37 }

```

esse nó. O algoritmo termina quando  $S = N$ . O seu pseudo código é apresentado na Listagem 16.1. Ele usa o vector `distance[]` para representar o resultado do cálculo de  $distancia(i)$ , que vai actualizando conforme vai progredindo. Inicialmente, este vector é inicializado da seguinte forma:

$\forall x \in N \text{ distance}[x] = \text{cost}(\text{orig}, x)$

`e  $distance(orig)=0$ , ver as linhas 13 a 22. O vector predecessor[] é usado para representar o nó que é o predecessor de cada nó  $x$  no caminho mais curto que foi escolhido de  $orig$  até  $x$ . No fim, este vector permitirá ao nó  $orig$  saber que caminho mais curto tomar para chegar a  $x$ . Inicialmente,  $\forall x \in N$  predecessor[x]=null com excepção dos casos particulares em que existe um arco do nó para  $orig$  (predecessor provisório) e do caso predecessor[orig]=orig, ver também as linhas 13 a 22.`

Para clarificar a forma como o algoritmo opera e como os seus resultados podem ser usados para implementar um protocolo de encaminhamento, vamos aplicá-lo à rede apresentada na Figura 16.1 (a), que é modelizada pelo grafo apresentado na Figura 16.1 (b). Nas figuras onde se apresenta a configuração de uma rede, os símbolos genéricos que são mais frequentemente usados para representar comutadores de pacotes são os que figuram na Figura 16.1 (a), *i.e.*, um cilindro com um conjunto de setas que simbolizam uma encruzilhada à qual chegam e da qual partem pacotes de dados. Os custos dos canais usados no modelo são inversamente proporcionais ao respectivo

débito. Para simplificar consideram-se todos os canais bidireccionais com custos iguais e equivalentes a dois arcos, um em cada sentido. Como os custos são iguais, apenas são indicados uma vez em cada arco.

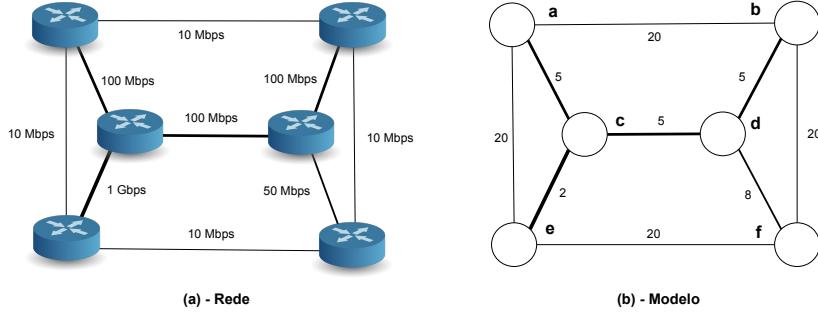


Figura 16.1: Rede (a) e sua modelização via um grafo (b)

A Figura 16.2 (a) mostra o estado do algoritmo ao grafo após a sua inicialização. O nó origem é o nó **a** que é o único elemento do conjunto  $S$ . Nessa figura e seguintes, os membros do conjunto  $S$  estão assinalados através de um círculo forte e o seu interior é sombreado. O nó pivot inicial é o próprio nó **a**. Em cada iteração o nó pivot está assinalado por uma seta nas figuras.

Adicionalmente, durante a inicialização, as entradas correspondentes aos nós **b**, **c**, **e** nos vectores **distance** e **predecessor** recebem o custo do arco que os liga directamente ao nó **a** e este torna-se o nó predecessor destes nós. Esta informação está apresentada no interior dos círculos correspondentes a esses nós. Em cada iteração, a distância e o nó predecessor são actualizados nos nós directamente ligados ao nó pivot (nos restantes não se podem alterar), e esse facto é assinalado sombreando o interior dos nós não pertencentes a  $S$  em que isso tem lugar. Por analogia, os nós directamente ligados a **a** estão sombreados no fim da inicialização.

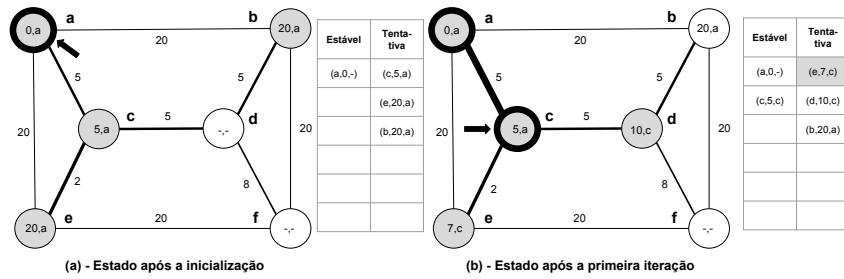


Figura 16.2: Estado do grafo após a inicialização (a) e o mesmo estado após a primeira iteração (b). À direita de cada grafo figura a visão do encaminhamento pelo nó **a**.

Para se ver como o algoritmo pode ser usado por um nó de comutação de pacotes, é necessário ter presente que estes necessitam de uma tabela que lhes permita, dado o destino de um pacote, saber por que interface transmitir o mesmo. Neste exemplo, ilustra-se essa tabela no nó **a**, pois estamos a usar o algoritmo para calcular os caminhos

mais curtos de  $a$  para os outros nós, ou seja, para todos os destinos possíveis. Essas tabelas chamam-se tabelas de comutação ou de encaminhamento e serão discutidas na secção seguinte.

Ao lado direito de cada grafo, é apresentada uma tabela que é uma primeira aproximação da tabela de encaminhamento do nó  $a$ , com a indicação da interface que o nó  $a$  tem de usar para atingir outro. Esta informação está codificada num triplo ordenado assim constituído (**destino**, **custo para lá chegar**, **vizinho para que enviar o pacote**). Na coluna *Estável* consideram-se apenas caminhos mais curtos. Na coluna *Tentativa* consideram-se formas conhecidas de chegar a outros nós, não necessariamente por um caminho mais curto.

Na Figura 16.2 (b) é apresentado o estado após a primeira iteração. Primeiro é seleccionado o novo **pivot** que será o nó não pertencente a  $S$  com a distância mínima. O nó seleccionado é o nó  $c$ . Esse nó é inserido em  $S$  e passou a ser conhecido um caminho mais curto para o mesmo, passando a figurar o mesmo na coluna *Estável*<sup>2</sup>.

Em seguida, é actualizada a distância a todos os nós directamente ligados ao novo **pivot** e que não pertencem a  $S$ . Os nós em questão são os nós  $\{e, d\}$ . Para o nó  $d$  passou a ser conhecido um caminho e por isso este entra na coluna *Tentativa*. No caso do nó  $e$  descobriu-se um caminho mais curto, via o nó  $c$ , e por isso a sua entrada na coluna *Tentativa* é alterada e está marcada a sombreado.

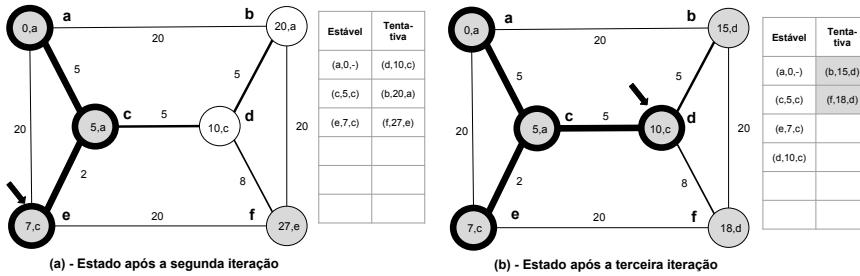


Figura 16.3: Estado do grafo após a segunda iteração (a) e após a terceira iteração (b). À direita de cada grafo figura a visão do encaminhamento pelo nó  $a$ .

Na Figura 16.3 (a) é apresentado o desenrolar da segunda iteração. Primeiro é seleccionado o novo **pivot**: o nó  $e$  passa a pertencer a  $S$  e entra na coluna *Estável* pois é o nó com menor distância. Depois é actualizada a distância e o nó sucessor dos nós que não pertencem a  $S$  mas que estão directamente ligados a  $e$ . O único nessas condições é o nó  $f$  que entra na coluna *Tentativa*.

A terceira iteração consta da Figura 16.3 (b). O nó  $d$  é o novo **pivot**, é inserido em  $S$  e na coluna *Estável*. Depois a distância e predecessor aos nós  $\{b, f\}$  são actualizadas e como em ambos os casos se descobrem novos caminhos mais curtos, as informações que lhes estão associadas na coluna *Tentativa* são alteradas em ambos os casos.

Na Figura 16.4 (a) é apresentado o estado no fim da quarta iteração. O nó  $b$  é o novo **pivot**, é inserido em  $S$  e na coluna *Estável*. Em seguida tenta-se actualizar a informação sobre os nós não pertencentes a  $S$  directamente ligados a  $b$ . Trata-se apenas do nó  $f$ , mas como não existe via  $b$  um melhor caminho para  $f$  tudo continua

<sup>2</sup> Visto que se chega a cada nó em  $S$  por um caminho mais curto, o caminho conhecido para se chegar ao nó não pertencente a  $S$  com função distância mínima, i.e., o novo **pivot**, é necessariamente um caminho mais curto porque qualquer outro implicaria que os caminhos conhecidos para os membros de  $S$  não seriam mais curtos. A demonstração realiza-se por absurdo.

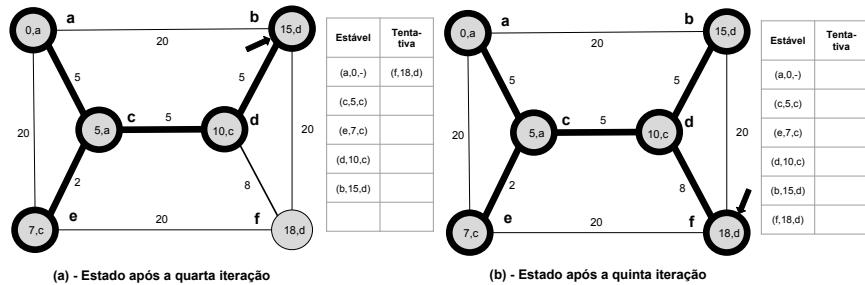


Figura 16.4: Estado do grafo após a quarta iteração (a) e após a quinta e última iteração (b). À direita de cada grafo figura a visão do encaminhamento pelo nó  $a$ .

na mesma. Finalmente, a Figura 16.4 (b) apresenta o estado final, após a quinta iteração. O novo nó **pivot** é inserido em  $S$  e na coluna *Estável* e mais nada se altera.

No fim, a coluna *Tentativa* está vazia, pois conhecem-se caminhos mais curtos para todos os nós, e a coluna *Estável* contém uma primeira versão da tabela de encaminhamento do nó  $a$ , em que figura, para cada destino existente na rede, a indicação do custo para lá chegar através de um caminho mais curto e o vizinho a usar para esse efeito.

Ao longo da evolução do algoritmos foi sendo igualmente marcada a árvore de cobertura de caminhos mais curtos que o algoritmo determinou. É interessante assinalar neste momento que este algoritmo acaba por calcular uma árvore de cobertura de caminhos mais curtos de forma centralizada, enquanto que o STP, se elegesse como raiz o nó  $a$ , calcularia de forma distribuída a mesma árvore (ou uma equivalente devido a poder existir mais do que um caminho mais curto).

Convém, no entanto, não perder de vista que o algoritmo distribuído usado pelo STP apenas calcula uma árvore de cobertura única para a rede inteira, que serve para encaminhar por inundação, e que com aprendizagem pelo caminho inverso, esse encaminhamento é óptimo apenas no caso em que o emissor é a raiz da árvore<sup>3</sup>. Enquanto que o algoritmo de Dijkstra pode ser aplicado a cada um dos nós e determinará nesse caso tantas árvores de encaminhamento óptimas quanto o número de nós existentes na rede, uma para cada origem.

Uma outra observação relevante tem a ver com o facto de que o STP usa um mecanismo para seleccionar, entre as diferentes opções de caminhos mais curtos, um caminho específico. Para a sua seleção usa os identificadores dos *switches* e das interfaces. O algoritmo de Dijkstra tem o mesmo problema quando o número de potenciais *pivots* é maior que um, ou quando ao actualizar a distância e o predecessor de um nó, se constata que existem diversas alternativas. Nesses casos o algoritmo tem também de tomar uma decisão arbitrária. As implementações reais podem optar por registar as diversas alternativas de caminhos mais curtos na tabela *Estável*.

Para terminar, assinalamos que este algoritmo requer um visão global do grafo, incluindo os nós, arcos e seus custos, e que, tal como foi apresentado, a sua complexidade temporal é  $O(n \times n)$ . É no entanto possível implementar uma versão cuja complexidade temporal é  $O(n \log n)$ .

---

<sup>3</sup> Rigorosamente, neste caso o encaminhamento é óptimo apenas do ponto de vista em que só usa caminhos mais curtos.

O algoritmo de Dijkstra é um algoritmo centralizado que, dado um grafo com arcos pesados e um nó origem, calcula uma árvore de cobertura de caminhos mais curtos com raiz no nó origem.

Este algoritmo, aplicado repetidamente a cada um dos nós de comutação de pacotes de uma rede, permite determinar caminhos mais curtos de cada nó para todos os outros, ou seja permite determinar tantas árvores de cobertura quantos os nós do grafo. Seja qual for o nó que toma uma decisão de encaminhamento de um pacote, encaminhá-lo-á por um caminho mais curto.

### 16.3 Encaminhamento pelo estado dos canais

Nesta secção e na seguinte vamos ver duas concretizações possíveis do **encaminhamento pelo menor caminho**, muitas vezes também designado *shortest path routing*. Antes vamos ver alguns aspectos comuns às duas formas de encaminhamento.

#### Routers ou comutadores inteligentes de pacotes

Os comutadores de pacotes com inteligência para executarem um protocolo genérico de encaminhamento de pacotes são muitas vezes designados em língua inglesa por *routers*, pois conhecem a noção de rota ou caminho (*route* ou *path*). O termo *router* pode ser traduzido por roteador ou encaminhador. No entanto, no resto do capítulo, continuaremos a usar o termo comutador de pacotes com inteligência, ou simplesmente comutador de pacotes, por ser essa a tradição no mundo dos profissionais de redes. A outra alternativa consiste em usar a terminologia inglesa que é de facto a que é usada mais frequentemente. Voltaremos a discutir a problemática da terminologia a usar no capítulo seguinte pois o problema merece maior aprofundamento e não é simplesmente uma questão da língua utilizada, ver o Capítulo 17.

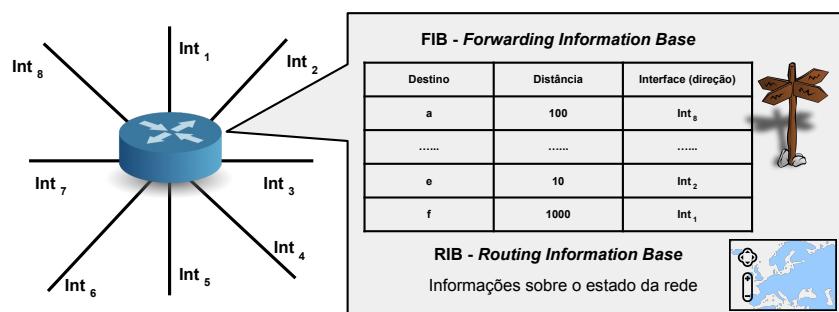


Figura 16.5: Comutador de pacotes e as duas bases de informação: FIB - *Forwarding Information Base* e RIB - *Routing Information Base*

Um comutador de pacotes contém em geral duas bases de informação, ver a Figura 16.5. A primeira chama-se **tabela de comutação ou de encaminhamento** e a terminologia em língua inglesa para a designar é **Forwarding Information Base (FIB)**, que será o termo que também usaremos. A FIB contém a informação que permite tomar rapidamente a decisão de comutação e é concretizada através de uma tabela de triplos (*destino*, *distância*, *interface*). Assim, quando recebe um novo

pacote, o comutador começa por extrair o endereço de destino do mesmo, depois consulta a FIB para saber qual a interface pela qual o pacote deve continuar a sua viagem até ao destino e, caso exista uma, o pacote é transmitido pela mesma. Senão, o pacote é ignorado pois o comutador não sabe que lhe fazer.

Caso um mesmo destino seja alcançável por várias interfaces, a que der acesso ao caminho mais curto será a escolhida. Naturalmente, se várias interfaces diferentes derem acesso a caminhos diferentes mas com o mesmo custo, é possível escolher uma delas de forma a tentar distribuir a carga pelos diferentes caminhos.

O comutador é, deste ponto de vista, semelhante a um cruzamento onde é necessário decidir qual a direção a seguir até ao destino. Para tal é usado o equivalente a um conjunto de “tabuletas a apontar para as diferentes direções”. No entanto, existe uma diferença, se um destino não está na tabela, o comutador não sabe como alcançá-lo e não consegue encaminhar o pacote. Por isso, é um cruzamento cheio de tabuletas, com milhares delas em redes de grande dimensão.

Mas como é que a FIB é preenchida? Uma das formas mais simples consiste em o administrador do comutador definir o conteúdo da FIB. Este tipo de encaminhamento chama-se encaminhamento estático e, como já foi referido, não é uma solução muito razoável, salvo em casos particulares. No caso geral, é o software que executa no comutador, que através de um protocolo de encaminhamento, preenche e actualiza a FIB de forma automática. Esse protocolo é executado em colaboração com outros comutadores, ou com servidores de controlo da rede. A informação que vai sendo adquirida pelo comutador através desses protocolos designa-se genericamente por ***Routing Information Base (RIB)***. A RIB permite ao comutador actualizar a FIB.

Por oposição à FIB, que pode ser assimilada a um “conjunto de tabuletas com distâncias e direções”, a RIB é como se fosse um “mapa das estradas” que permite conhecer os diferentes caminhos entre quaisquer duas cidades. Através de algoritmos é possível saber as rotas existentes, seleccionar a ou as melhores e preencher a FIB.

### Métricas de encaminhamento

Para comparar caminhos e canais para efeitos de encaminhamento é necessário conhecer o seu custo. O custo de um caminho é a soma dos custos dos canais que o formam. Cada protocolo de encaminhamento usa uma função, também chamada métrica do custo ou simplesmente métrica, que associa um custo a cada canal.

A métrica mais simples consiste em considerar que todos os canais têm o mesmo custo, por exemplo o custo 1. Nesse caso, o custo de um caminho é equivalente ao seu número de canais, o que se designa por comprimento do caminho. Este tipo de métrica é usada por alguns protocolos de encaminhamento mais simples, mas a mesma só é realista se todos os canais tiverem débitos, cargas e tempos de propagação semelhantes.

Em alternativa, poder-se-ia medir periodicamente o tempo necessário para atravessar um canal até ao próximo comutador. Para esse efeito poder-se-ia usar a mesma técnica que a empregada pelo programa ping, ver a Secção 3.3. No entanto, esta forma de determinar o custo teria alguns problemas.

Por um lado, a noção de custo está associada à direção do canal e, se as duas direções tiverem débitos diferentes, o custo associado a cada uma deve ser diferente. Mesmo que as duas direções de comunicação fossem equivalentes em débito e distância, o custo pode variar com a direção devido à intensidade do tráfego e devido ao estado das filas de espera. Ora o programa ping só mede o tempo de ida e volta e a sua divisão por dois pode não ser a mais adequada. É possível medir mais rigorosamente o tempo de trânsito em cada direção, mas isso exige relógios sincronizados nas duas extremidades, o que é uma exigência muito forte, apesar de mais realista hoje em dia usando GPS.

Por outro lado, a prática mostrou que o custo de um canal deve ser relativamente estável e não variar num curto espaço de tempo, como será discutido no final desta

secção. Por isso, no contexto do encaminhamento pelo estado dos canais, o custo deve ser apenas proporcional ao débito, combinado com outros factores como o tempo de propagação por exemplo. O custo dos canais pode ser fixado pelos administradores da rede, mas uma solução mais simples e mais comum é usar um custo fixado automaticamente e baseado apenas no débito. Por exemplo, usando a seguinte fórmula:

$$\text{Custo do canal} = \text{débito de referência}/\text{débito do canal}$$

O débito de referência é o débito correspondente ao custo 1. Se este débito for 1 Gbps ( $10^9$ ), então um canal com 100 Mbps terá o custo  $10 (10^9 \times 10^{-8})$  e um de 1 Mbps o custo 1000 ( $10^9 \times 10^{-6}$ ). Todos os comutadores de uma rede que usem uma métrica baseada em débito devem usar o mesmo débito de referência e este deve tomar o valor do débito dos canais de maior débito da rede.

Esta métrica é designada **métrica por omissão** nas implementações concretas.

Os *routers* ou comutadores de pacotes inteligentes dispõem de uma **tabela de comutação ou de encaminhamento (Forwarding Information Base (FIB))**, que lhes permitem comutar rapidamente os pacotes, e uma **Routing Information Base (RIB)**. A RIB é preenchida pelos protocolos de encaminhamento e permite ao comutador actualizar a FIB.

Para computar a FIB, os protocolos de encaminhamento necessitam de associar um custo ou métrica a cada canal da rede. Existem diversas métricas de encaminhamento. Uma das mais simples e comum, designa-se **métrica por omissão**, e consiste em calcular um custo inversamente proporcional ao débito do canal.

### Encaminhamento pelo estado dos canais

O encaminhamento conhecido como **encaminhamento pelo estado dos canais (link state routing)** baseia-se em encontrar uma forma de replicar em todos os comutadores um grafo com o estado global da rede. Como este estado global replicado é idêntico, cada comutador usa o algoritmo de Dijkstra para determinar uma árvore de cobertura de caminhos mais curtos para cada destino existente na rede. Depois, através destes caminhos, os quais constituem a totalidade das rotas baseadas em caminhos mais curtos existentes na rede, preenche a FIB com a informação que vai ser usada para encaminhar os pacotes.

Se todos os comutadores usarem da mesma descrição da rede, as diferentes FIBs serão compatíveis e encaminharão os pacotes pelas mesmas árvores de caminhos mais curtos desde a sua origem.

Caso o estado dos canais se altere, essa informação é de novo actualizada em todos os comutadores da rede e as FIBs são também de novo actualizadas em função do novo estado da rede.

Assim, cada comutador executa as seguintes operações:

1. Inicialmente, ou na sequência de uma alteração, cada comutador difunde aos outros o custo e o estado (*up / down*) dos canais que lhe estão ligados, assim como a identificação do nó na extremidade. Normalmente, estas informações chamam-se **Informações de Adjacência (Adjacency Announcements)**.
2. Depois de uma alteração local, ou remota mas recebida dos outros comutadores, cada comutador executa de novo o algoritmo de Dijkstra e determina quais são as entradas da sua FIB que se alteraram.

3. Por fim, o comutador repercute as alterações na sua FIB.

É por cada comutador adquirir por difusão o estado replicado da rede, que esta técnica de encaminhamento se pode também caracterizar por encaminhamento centralizado, pois baseia-se na execução de um algoritmo centralizado por cada comutador para preencher a FIB. A parte distribuída do protocolo consiste na difusão do estado entre comutadores.

Com este tipo de encaminhamento, a RIB dos comutadores designa-se ***Link State Database (LSDB)*** ou ***Adjacencies Database*** e contém o estado de todos os canais da rede, cada um dos quais caracterizado pelos identificadores (dos comutadores) de origem e destino, pelo seu custo e pelo seu estado (*up / down*). Na verdade, se se considerar que um canal no estado *down* (inativo) tem o custo infinito, então basta o custo para caracterizar o estado de cada canal. Repare-se que cada canal *full-duplex* é equiparado a dois canais *simplex* e corresponde a duas adjacências.

Como é que a LSDB é replicada por todos os comutadores? Cada comutador envia periodicamente (com um período de vários minutos), ou sempre que se produz uma alteração (incluindo na sua inicialização), **anúncios de estado dos canais (LSAs - Link State Announcements)** ou **anúncios de adjacência (adjacency announcements)** dirigidos a todos os outros comutadores. Cada comutador só origina a informação que lhe é local, *i.e.*, o conjunto das suas adjacências. A LSDB global é formada pela união de todas as informações locais.

Um LSA tem um identificador origem, *i.e.*, o identificador do comutador que o gera, um número de sequência e uma lista das suas adjacências. Os LSAs são depois propagados a toda a rede através do algoritmo de difusão fiável descrito na Secção 15.1. Recorde-se que esse algoritmo permite replicar em todos os membros da rede (os comutadores) mensagens (os LSAs) ordenadas por um número de sequência. Desta forma, sempre que se produz uma alteração, e periodicamente com um período alargado (de vários minutos), todos os comutadores enviam aos outros a informação necessária para estes preencherem as suas RIBs e, salvo durante a propagação das alterações de estado, todas as LSDBs dos diferentes comutadores terão os mesmos valores.

Associado a cada LSA existe também um parâmetro, designado “Idade” que denota o tempo desde que o LSA foi emitido pela última vez e é usado para evitar ter na LSDB informações que não são actualizadas há muito tempo. Assim, se um comutador for retirado da rede, ou se um canal for suprimido, ao fim de alguns minutos os seus LSAs deixarão de ser refreshados, a sua idade cresce até à idade máxima e será suprimido da LSDB.

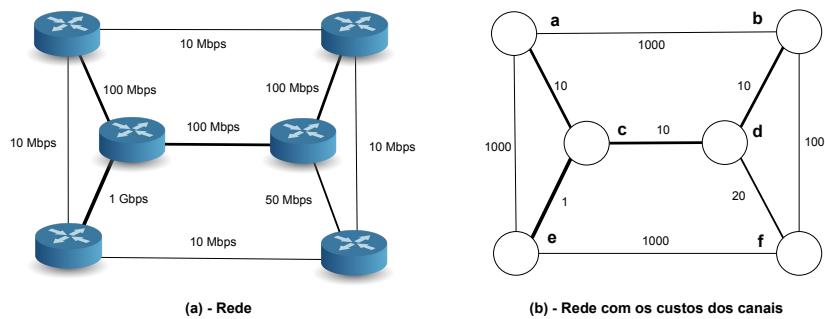


Figura 16.6: Rede e informações necessárias para preencher a LSDB da mesma

A Figura 16.6 mostra uma rede com as informações necessárias para preencher a LSDB. Esta base de dados está parcialmente ilustrada na Tabela 16.1

Tabela 16.1: Visão parcial da LSDB da rede da Figura 16.6. As entradas são a informação de adjacência e mostram o comutador que a originou, o número de sequência, a idade em minutos e a informação de adjacência.

Nó origem	# sequência	Idade	Adjacência
a	9	45	a - b, custo 1000
a	9	45	a - c, custo 10
a	9	45	a - e, custo 1000
b	12	16	b - a, custo 1000
b	12	16	b - d, custo 10
b	12	16	b - f, custo 1000
.....	.....	.....	.....
f	3	31	f - b, custo 1000
f	3	31	f - d, custo 20
f	3	31	f - e, custo 1000

Depois desta caracterização a alto nível deste tipo de protocolos de encaminhamento, vamos a seguir analisar mais em detalhe algumas das facetas mais importantes dos mesmos.

### Determinação do estado de um canal ou de uma adjacência

Para determinarem o estado dos seus canais, os comutadores enviam periodicamente para a outra extremidade de cada canal uma mensagem `hello`. O comutador da extremidade deve responder com outra mensagem `hello`. Desta forma os comutadores ficam a saber a identificação da outra extremidade e o estado de cada um dos seus canais. Se este for o único meio que permite ao comutador saber se o canal está operacional ou não, a velocidade com que uma avaria é detectada depende do período com que estas mensagens são enviadas. Uma deteção muito rápida requer uma frequência mais alta, por exemplo de 100 em 100 milissegundos, visto que só depois de várias tentativas falhadas o canal é dado como inoperacional.

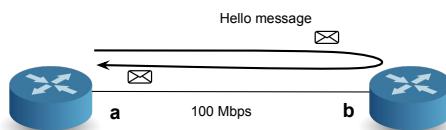


Figura 16.7: Execução do protocolo *Hello* entre dois comutadores interligados directamente por um canal ponto-a-ponto

Modernamente, a maioria dos canais ponto-a-ponto dispõem de mecanismos que permitem detectar a operacionalidade da outra extremidade através de sistemas de sincronização de baixo nível (detecção da onda portadora ou de sinais de sincronização). Nesse caso a interface lança alarmes rápidos caso o canal deixe de funcionar (*e.g.*, em cerca de 10 ms). De qualquer forma, mesmo que este tipo de mecanismos existam, por segurança continua a usar-se o protocolo `hello`, pois este também permite conhecer os identificadores de ambas as extremidades e reconhecer que, para além do canal estar

operacional, a outra extremidade também participa no protocolo. Para este efeito, basta enviar mensagens `hello` espaçadas de um ou dois minutos.

Finalmente, os comutadores conseguem saber através das suas interfaces qual o débito dos canais e assim calculam o custo dos mesmos.

Como o algoritmo de Dijkstra só “sabe” lidar com canais ponto-a-ponto, os canais multi-ponto de interligação de vários comutadores levantam um problema suplementar. Uma forma de resolver esse problema consistiria em transformar um canal multi-ponto de interligação de  $n$  comutadores em  $n \times (n - 1)/2$  canais ponto-a-ponto ou o dobro de adjacências. Para evitar esta explosão de adjacências, a solução consiste em um dos comutadores ser eleito, por um processo semelhante ao usado pelo protocolo STP, e tornar-se o *designated router* do canal. Este comutador introduz um espécie de comutador virtual suplementar na rede e associa um identificador virtual de comutador à sua interface para o canal. Assim, os  $n$  comutadores interligados passam a corresponder a  $2 \times (n - 1)$  adjacências,  $n - 1$  de interligação do *designated router* com os outros  $n - 1$  comutadores, e outros  $n - 1$  no sentido contrário. Todas as comunicações entre estes  $n - 1$  comutadores passam necessariamente pelo *designated router*, ver a Figura 16.8.

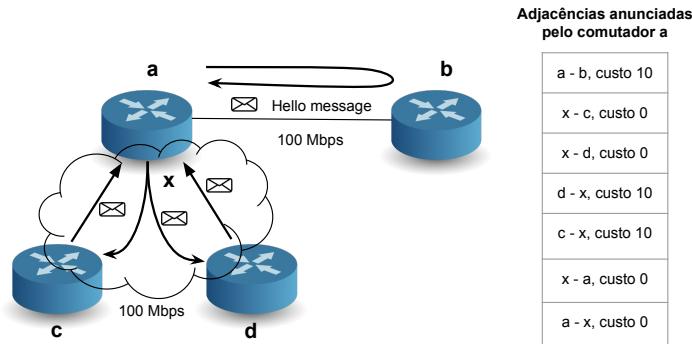


Figura 16.8: Entre os comutadores directamente ligados a um canal multi-ponto, um é eleito *designated router* e passa a ser responsável por um pseudo comutador, com o objectivo de evitar a explosão do número de canais ponto-a-ponto da rede.

#### Disseminação fiável do estado e partições da rede

Diz-se que se forma uma partição da rede quando a mesma fica dividida em duas ou mais partes incapazes de comunicarem entre si. Por exemplo, se na rede da Figura 16.9 (a) o canal entre os comutadores *c* e *d* avariar, o seu custo torna-se infinito e as duas partições resultantes, ver a Figura 16.9 (b), ficam incomunicáveis entre si.

Na sequência da avaria, o comutador *c* dissemina aos comutadores *a* e *e* que o canal passou a ter o custo infinito. A aplicação do algoritmo de Dijkstra nos outros comutadores conduz a uma LSDB e uma FIB nos comutadores *a*, *c* e *e* em que os destinos *b*, *d* e *f* estão à distância infinito, i.e., inacessíveis. Simetricamente, nas FIBs dos comutadores *b*, *d* e *f* os destinos *a*, *c* e *e* passaram a estar à distância infinito. Se a partição durar mais que o TTL máximo, de cada lado da rede, os LSAs emitidos pelos comutadores da outra partição acabarão por ser esquecidos por falta de refreshamento, visto que os LSAs de um lado não passam para o outro e vice-versa.

No entanto, que acontece se algum tempo depois da avaria esta recupera e a comunicação se estabelece de novo? Até que novos LSAs sejam emitidos por todos

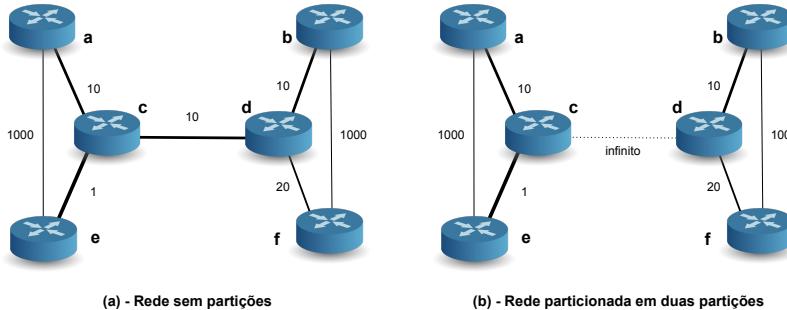


Figura 16.9: Rede completa com todos os canais operacionais (a). Na sequência da avaria do canal entre os comutadores c e d, a rede fica partitionada em duas incomunicáveis entre si (b).

os comutadores de cada lado, a outra parte continuaria com uma visão eventualmente desfasada do estado da rede. Para recuperar o mais rapidamente que possível de uma partição, quando um comutador se apercebe que um evento acabou de reparar uma partição com outro, ambos os comutadores cuja comunicação foi restabelecida trocam mutuamente todos os LSAs que conhecem. Ao executarem o protocolo de disseminação fiável da Listagem 15.1 para cada um dos LSAs, sincronizam e actualizam as respectivas adjacências.

Este processo designa-se nas implementações concretas *to bring up adjacencies* e é optimizado através da troca de mensagens designadas por *database descriptive packets*, trocadas entre os comutadores, e que contém os identificadores dos LSAs conhecidos e os respectivos números de sequência.

O mesmo mecanismo permite que um comutador que acabou de se ligar à rede passe a conhecer toda a LSDB da rede e possa computar uma FIB que lhe possibilitará o envio de pacotes para todos os destinos acessíveis na mesma.

Adicionalmente, o mecanismo que permite a um comutador, quando recebe um LSA atrasado, enviar ao outro a versão actual do mesmo LSA, permite que um comutador que se desligou da rede, e se liga mais tarde antes que o TTL não leve ao seu esquecimento pela rede, não seja ignorado por todos os outros comutadores. Com efeito, esse comutador inicializa o número de sequência dos seus LSAs a 1 mas os outros conhecem-no por números de sequência mais elevados. O mecanismo permite-lhe ficar a conhecer o último número de sequência que usou.

### Convergência

Quando um ou mais eventos que alteraram a configuração da rede têm lugar, geram-se LSAs que são propagados pelos diferentes comutadores. Quando os outros comutadores recebem esses LSAs, alteram as suas LSDBs e na sequência disso alteram as respectivas FIBs.

Chama-se **tempo de convergência** (*convergence time*) de um protocolo de encaminhamento ao tempo que passa desde que se produz uma alteração da rede (*e.g.*, um canal avariou-se), até ao momento em que todos os comutadores actualizaram as RIBs e FIBs e toda a rede voltou a estar num estado estável.

Este processo tem lugar de forma assíncrona nos diferentes comutadores e por isso todas as LSDBs e as FIBs dos comutadores não são alteradas instantaneamente e de forma sincronizada. Enquanto essas alterações não estabilizarem, os diferentes comutadores têm visões distintas da rede e podem encaminhar os pacotes em contradição

uns com os outros. Como resultado dessas incoerências, alguns pacotes podem entrar em ciclo, como é ilustrado na Figura 16.10.

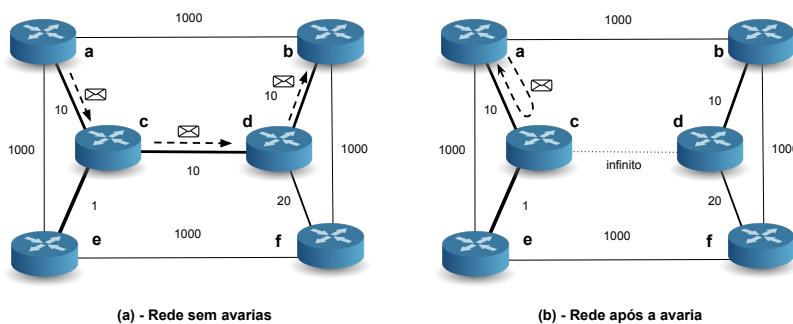


Figura 16.10: Antes da avaria do canal entre os comutadores c e d, parte (a), os pacotes que vão de a para b seguem via o caminho a, c, d, b por este ser o mais curto. Depois da avaria, parte (b), o comutador c acha que o melhor caminho é via a, b mas o comutador a ainda não actualizou a sua FIB e os pacotes entram em ciclo.

Para que estas incoerências momentâneas não desencadeiem uma catástrofe, todos os pacotes de dados assim encaminhados devem dispor de um mecanismo de segurança baseado num TTL. Se algum pacote entrar em ciclo durante a incoerência, acabará, mais tarde ou mais cedo, por ser suprimido.

Idealmente, o tempo de convergência deve ser o mais breve que possível, quer para evitar gastar recursos inutilmente durante a instabilidade, quer porque durante a mesma se perdem pacotes, o que pode afectar as aplicações, nomeadamente as que envolvem facetas de tempo real e que são não elásticas, como a generalidade das aplicações multimédia.

À primeira vista, a maneira mais simples de diminuir o tempo de convergência consiste em diminuir todos os tempos de reação aos diferentes eventos: deteção da alteração, início da difusão do LSA, recepção do LSA, computação do novo caminho mais curto e alteração da FIB. Infelizmente, a prática mostrou que isso pode conduzir à instabilidade da rede.

Com efeito, uma alteração, por pequena que seja, conduz ao envio de mais do que um LSA. Pelo menos dois, *i.e.*, um por cada um dos comutadores envolvidos se se tratar de um canal ponto-a-ponto. Se a avaria for de um comutador, serão enviados tantos LSAs quantos os canais ao qual ele estava ligado. Em certas situações isso pode envolver dezenas de LSAs.

Se a reação for muito rápida, isso pode conduzir a que cada comutador da rede desencadeie vários cálculos e alterações das FIB e RIB sucessivamente, cada uma das quais envolvendo cada um dos LSAs recebidos. Por outro lado, se a rede for complexa, o algoritmo de Dijkstra envolve muitos nós e canais e é computacionalmente pesado. Por isso, é preferível não o executar senão um certo tempo depois de o ter executado pela última vez, e também deixar passar algum tempo desde que chega um LSA antes de o desencadear, na esperança que cheguem todos os LSA relacionados com o mesmo evento.

Por outro lado, é frequente produzirem-se avarias intermitentes. Por exemplo, um canal cujo estado alterna entre estar operacional e deixar de o estar. Resultado, uma instabilidade de uma componente da rede pode desencadear uma instabilidade generalizada da rede. Uma maneira simples de realizar um ataque de negação de

serviço consiste em enviar periodicamente LSAs falsos que conduzam a reconfigurações constantes de todos os comutadores.

Na prática, os temporizadores associados ao desencadear da reação aos diferentes eventos têm de ser elevados. Quando os protocolos de encaminhamento com base no estado dos canais foram introduzidos, o valor por omissão dos alarmes usados levavam a que o tempo de convergência fosse da ordem de grandeza de 10 segundos. Isso era reforçado pelo facto de que os comutadores usados na época tinham pouco poder computacional e pouca memória, e considerava-se que era preferível uma convergência mais lenta, do que uma grande instabilidade em toda a rede induzida por constantes reconfigurações, ou comutadores com a CPU saturada e incapazes de responderem aos diferentes eventos que iam tendo lugar.

Na verdade, na gestão das redes, é necessário fugir “como o diabo da cruz” de toda a instabilidade, seja qual for a sua origem: *bugs* de software nos comutadores, comutadores com a CPU saturada e incapazes de reagirem atempadamente aos eventos, instabilidades no hardware, canais com problemas, erros dos gestores por precipitação em casos de emergência, *etc*. Actualmente fizeram-se bastantes progressos na engenharia dos comutadores e na implementação concreta destes protocolos e é possível obter tempos de convergência inferiores a 1 segundo em redes complexas [Francois et al., 2005].

### Métricas de custo

O problema da convergência permite igualmente ilustrar porque razão as métricas de custo usadas na prática não tomam em consideração alterações relacionadas com modificações da intensidade do tráfego e da dimensão das filas de espera. A Figura 16.11 mostra porquê. Na mesma, as etiquetas de cada canal mostram a intensidade do tráfego que o atravessa. Por hipótese, neste exemplo, o custo de um canal considera-se proporcional a essa intensidade de tráfego, dado que os débitos e tempos de trânsito dos canais são idênticos. Atendendo a que o custo é função da intensidade do tráfego, o custo varia com a direcção do tráfego e um canal tem custos distintos nas duas direções. Quando o custo não está assinalado na figura numa dada direcção, isso quer dizer que o mesmo custo é desprezável pois não há tráfego nessa direcção.

As setas junto a cada comutador mostram o tráfego que cada um deles injecta na rede. Por hipótese, os comutadores **a** e **c** injectam a mesma quantidade  $x$  de tráfego na rede, em ambos os casos dirigido ao comutador **b**. O comutador **d** injecta a quantidade  $\delta$  (muito pequena em face de  $x$ ) também dirigida ao comutador **b**.

A Figura 16.11 (a) mostra as decisões de encaminhamento tomadas pelos comutadores logo após o aparecimento do tráfego  $\delta$ . O comutador **d** toma a decisão de enviar o tráfego  $\delta$  pela direita, ora essa decisão faz com que o custo visto por **c** para enviar tráfego para **b** seja superior ao de enviá-lo pela outra alternativa, e a rede reconfigura-se para usar o encaminhamento ilustrado na Figura 16.11 (b). Só que agora o comutador **a** vê o custo com que envia tráfego para **b** aumentar e a rede altera-se para a configuração de encaminhamento da Figura 16.11 (c). O mesmo raciocínio leva a uma nova reconfiguração para a situação da Figura 16.11 (d), e assim sucessivamente.

Por conseguinte, fazer a intensidade de tráfego ser considerada de forma muito “apertada” na métrica do custo pode conduzir a situações de instabilidade suplementares.

Em conclusão, a técnica de encaminhamento designada como encaminhamento pelo estado dos canais requer cuidados na sua concretização, relacionados com a não atomicidade com que a alteração da LSDB nos diferentes comutadores tem lugar. Isso complica a implementação concreta desta forma de fazer o encaminhamento, nomeadamente no que diz respeito à coordenação da evolução do estado replicado nos diferentes comutadores. Estas dificuldades estão relacionadas com alguns dos problemas fundamentais dos sistemas distribuídos, nomeadamente a gestão de estado replicado e a coordenação das diferentes componentes do sistema distribuído. Estas dificuldades

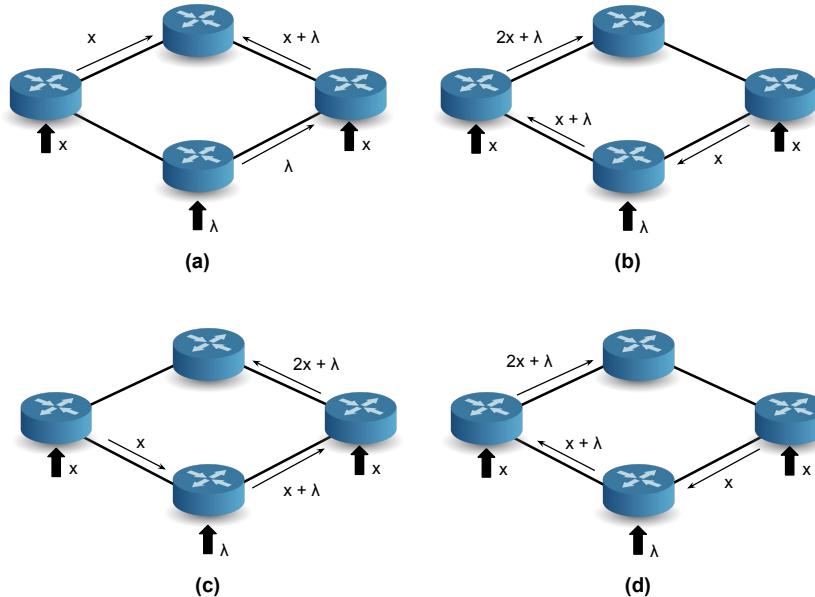


Figura 16.11: Considerar a intensidade do tráfego ou a dimensão das filas de espera de forma significativa na métrica do custo de um canal pode conduzir a instabilidade no encaminhamento.

estão relacionadas com a ausência de um relógio global que sincronize as diferentes componentes do sistema.

A técnica de encaminhamento pelo caminho mais curto com base no **estado dos canais** (*link-state routing*) baseia-se na replicação por todos os comutadores, por difusão fiável, de todas as adjacências existentes na rede. Desta forma, cada comutador adquire a visão do grafo que caracteriza a rede e, usando o algoritmo de Dijkstra, pode calcular os melhores caminhos para todos os destinos existentes e assim construir a sua FIB.

Sempre que têm lugar modificações do estado de qualquer componente da rede, os comutadores que as detectam difundem actualizações, também designadas SLAs (*Link State Announcements* ou *Adjacency Announcements*) e, logo que a difusão fiável se completar, as actualizações são repercutidas nas FIBs de todos os comutadores e a rede converge voltando a um estado estável.

Existem dois protocolos concretos baseados nesta filosofia de encaminhamento: o protocolo IS-IS (*Interior System to Interior System*), definido pela norma ISO/IEC (ISO/IEC 10589:2002, Second Edition), e o protocolo OSPF (*Open Shortest Path First*), ver os RFCs 2328 (IPv4) e 5340 (IPv6). O primeiro foi definido, inicialmente, como sendo mais genérico e mais flexível pois permitia a extensão do tipo de informações transportadas pelos LSAs, um mecanismo conhecido como *type-length-value* (TLV), que o tornou independente do protocolo de encaminhamento concreto que suportava pois é capaz de replicar diversos tipos de informações.

O segundo foi introduzido como uma especialização do primeiro para a comutação de pacotes IPv4. Esta especialização teve repercussões posteriores. Por exemplo, a utilização de IS-IS para IPv6 não requereu nenhuma nova versão do protocolo, enquanto que no caso do OSPF foi necessário introduzir uma nova versão. No entanto, a quantidade de opções do OSPF para a gestão concreta de uma rede IP era mais completa e a extensão do tipo da informação transportada pelos LSAs também foi posteriormente introduzida no OSPF.

A opção por um ou por outro protocolo depende muitas vezes da maturidade da implementação nos equipamentos e das opções de gestão dos mesmos. Geralmente, o IS-IS é mais usado pelos ISPs, enquanto que o OSPF é preferido nas redes empresariais. No entanto, esta regra não é obrigatória.

O protocolo IS-IS é também usado para replicar estado genérico entre um grupo de comutadores fora do mundo da comutação de pacotes IP. Por exemplo, o IS-IS é usado para replicar estado nas novas propostas de encaminhamento para *switches* Ethernet com base em endereços MAC, nomeadamente TRILL (Transparent Interconnection of Lots of Links) e Shortest Path Bridging, ambas referidas no final do capítulo anterior.

## 16.4 Encaminhamento pelo algoritmo Bellman-Ford

Antes da introdução do método de encaminhamento com base no estado dos canais foram usadas outras formas de encaminhamento. Nas redes mais antigas e simples, era comum usar exclusivamente encaminhamento estático, com todos os defeitos deste método já assinalados. Foram também usados métodos dinâmicos baseados num controlador central que alterava as FIBs dos comutadores se o estado da rede se alterasse. Em ambos os casos os comutadores não necessitavam de uma RIB, apenas dispunham de uma FIB que era remotamente alterada.

Mais tarde, surgiu a ideia de dar inteligência e autonomia aos comutadores de forma que estes se adaptassem automaticamente às alterações do estado da rede. No entanto, antes da introdução do método visto na secção anterior, foi usada uma outra forma de encaminhamento distribuído, autónomo e dinâmico, baseado no algoritmo de Bellman-Ford, do nome dos seus dois inventores. Estes introduziram o mesmo algoritmo de forma independente e quase simultânea, pelo que é tradicional dar a ambos o crédito pela sua invenção. No final da secção compararemos as duas abordagens: encaminhamento com base nos estados dos canais e encaminhamento com base no algoritmo de Bellman-Ford.

Esta última forma também costuma ser designada por encaminhamento com base em vectores de distâncias (a distância é, como vimos atrás, a nomenclatura convencional para o custo de um caminho), pois uma das formas mais populares da sua concretização passa por os comutadores trocarem entre si vectores contendo a lista dos destinos que cada um conhece e a que distâncias os consegue alcançar.

Esta abordagem parte de uma ideia muito simples: se um comutador  $x$  conhece um conjunto de destinos, pode comunicar esse facto aos seus vizinhos directos. Quando um deles recebe essa informação, dado que conhece a distância (custo) que o separa de  $x$ , fica a conhecer uma (nova) forma de alcançar esses destinos somando o custo de chegar ao vizinho  $x$ , que lhe fez o anúncio, às distâncias por este anunciadas. Se essa forma de alcançar um dado destino  $k$  for melhor que a que o comutador conhecia previamente, então pode passar a encaminhar os pacotes dirigidos a  $k$  via o vizinho  $x$ . Os anúncios enviados pelos comutadores aos seus vizinhos chamam-se **vectores de distâncias** (*distance vectors*) ou **anúncios de visibilidade** (*reachability announcements*), ver a Figura 16.12.

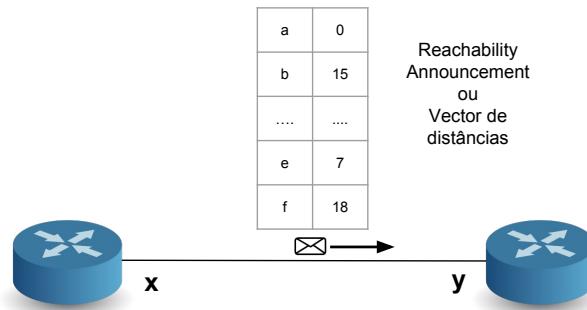


Figura 16.12: O comutador  $x$  passa a um seu vizinho  $y$  o conjunto de destinos que conhece e a que distância os consegue alcançar, através de um vector de distâncias ou *reachability announcement*.

Cada comutador guarda na sua RIB os anúncios de visibilidade que recebeu dos seus vizinhos directos. Através destes pode determinar qual o caminho mais curto para chegar a cada destino. Antes de prosseguirmos, convém ter presente que um comutador só passa anúncios aos seus vizinhos directos. Estes, processam os anúncios recebidos, preenchem a sua FIB e, periodicamente, ou sempre que tiver lugar uma alteração, constroem de novo um anúncio de visibilidade, e passam-no a todos os seus vizinhos directos. Não se trata portanto de um algoritmo de difusão fiável pois cada comutador envia aos seus vizinhos a sua visão, que é diferente da dos outros. Os comutadores só guardam na RIB os anúncios dos seus vizinhos directos, não os anúncios de todos os comutadores da rede.

Quando recebe um anúncio de um vizinho, um comutador pode actualizar imediatamente a sua FIB pois o processamento do anúncio é fácil. A listagem 16.2 indica uma versão preliminar desse processamento.

Listing 16.2: Processamento de um anúncio pelo algoritmo de Bellman-Ford

```

1 Input:
2   x - node that sent the announcement
3   d - distance or cost to reach x
4   reachability [1..k] - received reachability announcement
5   distance [1..k] - FIB - distance to each node (1..k)
6   nexthop [1..k] - FIB - neighbor selected to reach each node (1..k)
7
8   for i in 1..k do {
9     if ( reachability [i] + d < distance [i] ) {
10       distance [i] = reachability [i] + d
11       nexthop [i] = x
12   }

```

Isto é, “se  $x$  anunciar um destino  $i$ , a uma distância tal que a soma dessa distância com o custo para alcançar  $x$ , for menor que o caminho conhecido actualmente para chegar a  $i$ , então a melhor maneira que se tem de alcançar o destino  $i$  é via  $x$ , e  $i$  passará a esta à distância de chegar a  $x$  mais a distância com que  $x$  consegue chegar a  $i$ ”.

Para se perceber como o algoritmo permite construir dinamicamente as FIBs de todos os comutadores de uma rede vamos admitir, por hipótese, que a rede está integralmente sincronizada por um hipotético relógio global, e que cada nó executa de forma sincronizada um passo do algoritmo sempre que o relógio emite um sinal. Em cada comutador este passo é assim constituído:

Compor um vector de distâncias ou anúncio de visibilidade  
 Enviá-lo a cada um dos vizinhos  
 Receber os vectores enviados pelos vizinhos  
 Processá-los em sequência de acordo com o algoritmo 16.2 e actualizar a FIB

Após a inicialização, cada comutador só conhece o seu próprio destino (por hipótese o seu identificador como na secção anterior) assim como o dos seus vizinhos directos e os custos para os alcançar. Esta informação é fácil de adquirir através, por exemplo, de uma mensagem `hello`. Depois, a cada sinal do relógio ou iteração, todos os nós começam por enviar o seu vector de distâncias aos seus vizinhos directos, lêem os vectores distâncias recebidos, processam-nos e actualizam a sua FIB. O processo acaba quando a rede convergir. Por outras palavras, quando todos os nós ao receberem vectores distâncias, já não “aprenderem” mais nada com os seus vizinhos (ou seja, qualquer novo vector de distâncias recebido não altera nenhuma entrada de nenhuma FIB). A rede usada para exemplificar será a mesma que já usámos na ilustração do funcionamento do algoritmo de Dijkstra.

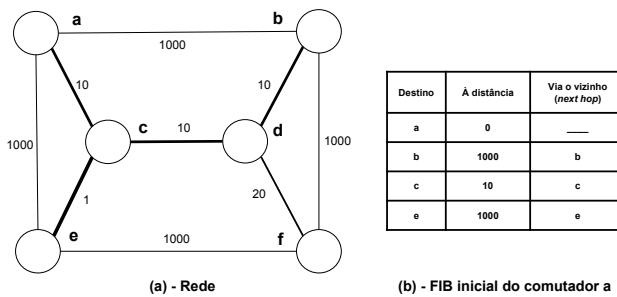


Figura 16.13: Rede usada no exemplo e FIB inicial do comutador a

A Figura 16.13 (a) mostra a rede com as informações necessárias para preencher a FIB de cada comutador. A Figura 16.13 (b) mostra a FIB inicial do comutador a. A Figura 16.14 mostra a execução de um passo do algoritmo do ponto de vista do comutador a. A FIB inicial, Figura 16.14 (a), é a FIB do comutador após a sua inicialização. A Figura 16.14 (b) mostra os vectores de distâncias que lhe são enviados pelos vizinhos, e a Figura 16.14 (c) mostra a mesma FIB depois de esta incorporar o processamento de cada um dos vectores recebidos, pela ordem indicada. As entradas que foram modificadas na FIB do comutador a estão sombreadas se foram alteradas pelo processamento dos anúncios recebidos dos vizinhos.

Com o vector do comutador b, o comutador a passa a conhecer o comutador d à distância 1010 (1000 para b mais 10 deste para d) e o destino f à distância 2000 (1000 para b mais 1000 deste para f). Com o vector do comutador c, o comutador a passa a conhecer o comutador d à distância 20 (10 para c mais 10 deste para d e melhor que os 1010 anteriores) e o destino e passou a estar à distância 11 (10 para c mais 1 deste para e). Com o vector do comutador e, o comutador a não obtém nenhuma informação que o leve a alterar o encaminhamento escolhido.

Neste último caso, poder-se-ia perguntar: mas se a já sabe agora que existe um caminho para e com custo 11, porque não usar essa informação para ir para f via esse caminho à distância de 11 mais o custo com que e conhece um caminho para f, ao invés de considerar a sua distância a e nos cálculos? De facto seria possível fazer uma outra versão do algoritmo que analisasse mais profundamente o conteúdo

da RIB, no entanto, o facto é que se existir uma forma melhor de chegar a um destino, ele aparecerá necessariamente num anúncio posterior do comutador que dá início ao mesmo, como se verá a seguir.

The diagram illustrates the state of the FIB and received distance vectors for each switch during the first iteration:

- (a) - FIB do comutador a antes da primeira iteração:**

Destino	d.	next hop
a	1000	a
b	0	—
c	20	d
d	10	d
e	2000	f
f	30	d

- (b) - Vectores distância recebidos pelo comutador a:**

Destino	À distância	Via o vizinho (next hop)
a	0	—
b	1000	b
c	10	c
e	1000	e

- 1 - vector distância enviado por b:**

Destino	a	b	d	f
Distância	1000	0	10	1000

- 2 - vector distância enviado por c:**

Destino	a	c	d	e
Distância	10	0	10	1

- 3 - vector distância enviado por e:**

Destino	a	c	e	f
Distância	1000	1	0	1000

- (c) - FIB após o processamento dos anúncios:**

Destino	À distância	Via o vizinho (next hop)
a	0	—
b	1000	b
c	10	c
d	1010 / 20	b / c
e	1000 / 11	e / c
f	2000	b

Figura 16.14: (a) FIB inicial do comutador a. (b) vectores distância recebidos durante a primeira iteração. (c) FIB após processamento pela ordem dos vectores distância recebidos.

A Figura 16.15 mostra a evolução da FIB dos restantes comutadores no fim da primeira iteração. Por falta de espaço não se mostra a FIB inicial, nem os vectores de distâncias recebidos por cada comutador dos seus vizinhos directos.

The diagram shows the final FIB of five switches after the first iteration:

- FIB do comutador b:**

Dst	d.	next hop
a	1000	a
b	0	—
c	20	d
d	10	d
e	2000	f
f	30	d

- FIB do comutador c:**

Dst	d.	next hop
a	10	a
b	20	d
c	0	—
d	10	d
e	1	e
f	30	d

- FIB do comutador d:**

Dst	d.	next hop
a	20	c
b	10	b
c	10	c
d	0	—
e	11	c
f	20	f

- FIB do comutador e:**

Dst	d.	next hop
a	11	c
b	2000	a
c	1	c
d	11	c
e	0	—
f	1000	f

- FIB do comutador f:**

Dst	d.	next hop
a	2000	b
b	30	d
c	30	d
d	20	d
e	1000	e
f	0	—

Figura 16.15: FIB dos restantes comutadores no fim da primeira iteração

A Figura 16.16 mostra a evolução da FIB do comutador a durante a segunda iteração. Os vectores recebidos por a já incorporam o que os seus vizinhos aprenderam com os vizinhos deles no final do passo anterior. Com o vector enviado por b, a apenas aprende que existe um caminho para f via b e este ficará à distância 1030. No entanto, com o vector enviado por c é possível concluir que existe um melhor caminho via este comutador e f ficará à distância 40. Com esse mesmo vector é possível concluir que via o mesmo comutador existe um caminho para b e este ficará à distância 30.

The diagram illustrates the Bellman-Ford algorithm iteration process through three tables:

- (a) - FIB do comutador a antes da segunda iteração:** Shows the initial Forwarding Information Base (FIB) for switch a. It has 6 entries: (a, 0, —), (b, 1000, b), (c, 10, c), (d, 20, c), (e, 11, c), and (f, 2000, b). The 'Via o vizinho (next hop)' column is empty.
- (b) - Vectores distância recebidos pelo comutador a:** Shows vectors of distance received from neighbors. There are three vectors corresponding to switches b, c, and e. Each vector has 6 entries: (a, b, c, d, e, f). The 'Via o vizinho (next hop)' column is empty.
- (c) - FIB após o processamento dos anúncios:** Shows the FIB after processing announcements. It has 6 entries: (a, 0, —), (b, 30, c), (c, 10, c), (d, 20, c), (e, 11, c), and (f, 1030 / 40, b / c). The 'Via o vizinho (next hop)' column is present.

1 - vector distância enviado por b						
Destino	a	b	c	d	e	f
Distância	1000	0	20	10	2000	30

2 - vector distância enviado por c						
Destino	a	b	c	d	e	f
Distância	10	20	0	10	1	30

3 - vector distância enviado por e						
Destino	a	b	c	d	e	f
Distância	11	2000	1	11	0	1000

Figura 16.16: (a) FIB do comutador a antes da segunda iteração. (b) vectores distância recebidos dos vizinhos de a. (c) FIB do comutador a após a segunda iteração.

Novas iterações não conduzirão à descoberta de alternativas melhores para chegar aos diferentes destinos e o processo estabilizará. É possível concluir isso executando mais iterações. No entanto, é também possível observar o seguinte: após a inicialização, todos os comutadores conhecem caminhos de comprimento 1 para os seus vizinhos. No fim da primeira iteração passam a conhecer caminhos de comprimento 2 e destes já selecionaram os melhores. Finalmente, no fim da segunda, todos os comutadores já conhecem os melhores caminhos de comprimento 3. Ora como não existem caminhos mais curtos de comprimento maior, os mesmos não podem ser descobertos através de iterações suplementares.

Se considerarmos que durante a inicialização cada comutador apenas se descobre a si próprio, e só passa aos seus vizinhos directos um primeiro vector com a sua própria identificação, então o algoritmo de Bellman-Ford consegue que os comutadores descubram os melhores caminhos para todos os outros comutadores no máximo em tantos passos como o diâmetro da rede, *i.e.*, o comprimento do maior caminho mais curto existente na mesma.

Numa implementação concreta do algoritmo, *i.e.*, num protocolo de encaminhamento, é necessário tomar em consideração que não é fácil introduzir um relógio global que sincronize a execução em todos os comutadores. Também não é possível garantir que todos os anúncios são recebidos num tempo bem definido e que não há atrasos nem perda de mensagens. Assim, as implementações concretas são assíncronas e têm mecanismos para combater a perda de mensagens, *e.g.*, reenviando os anúncios periodicamente. Por outro lado é necessário ter em consideração que a rede muda de estado dinamicamente. Por exemplo, se um canal se avariar, no momento seguinte todos os caminhos que o usavam deixam de ser viáveis e o algoritmo tem também de tomar isso em consideração. Por isso, o vector de distâncias de cada comutador altera-se também sempre que a distância aos seus vizinhos se modifica. Os mais conhecidos protocolos de encaminhamento baseados no algoritmo de Bellman-Ford enviam novos vectores distâncias aos vizinhos após a inicialização, sempre que esses vectores se alteram e ainda periodicamente para compensar eventuais perdas de mensagens e controlar longas avarias de comutadores.

A listagem 16.3 introduz o pseudo-código de uma versão mais completa, mas ainda assim simples, do protocolo. Este funciona de forma assíncrona pois as acções têm lugar na sequência de eventos exclusivamente locais a cada comutador. Adicionalmente,

Listing 16.3: Pseudo código de um protocolo de encaminhamento com base no algoritmo de Bellman-Ford

```

1 Local data including the FIB:
2   distance[1..k] - distance to reach node i
3   nexthop[1..k] - the first hop in the shortest path to reach node i
4   boolean neighbours[1..k] - node i is a direct neighbour
5
6 On event < START >:
7   for i in 1..k do {
8     distance[i] = infinity
9     nexthop[i] = null
10   }
11  distance[myself] = 0
12  nexthop[myself] = myself
13  // probe links, discover neighbours, initialize neighbours[1..k]
14  // and update their distance and nexthop
15  sendReachability(true)
16
17 On event < link to neighbour i becomes INOPERATIONAL >:
18  neighbours[i] = false
19  // update path to all destinations using i as next hop
20  for j in 1..k do if ( nexthop[j] == i ) {
21    nexthop[j] = null
22    distance[j] = infinity
23  }
24  sendReachability(false)
25
26 On event < link to neighbour i returns to OPERATIONAL >:
27  neighbours[i] = true
28  // if we previously knew no other path to i
29  if ( nexthop[i] == null ) {
30    nexthop[i] = i
31    distance[i] = its value
32    sendReachability(true)
33  }
34
35
36 On event <PERIOIC TIMER triggered >:
37  sendReachability(true)
38
39 On event <receive reachability ANNOUNCEMENT (reachability[1..k]) from
40 neighbour x> : // at distance d
41  for i in 1..k do {
42    // if we route to i by x, always update distance
43    if ( nexthop[i] == x ) distance[i] = reachability[i] + d
44    // else try to learn about better alternatives
45    else if ( reachability[i] + d < distance[i] ) {
46      distance[i] = reachability[i] + d
47      nexthop[i] = x
48    }
49  sendReachability(false)
50
51
52 void sendReachability(boolean always) {
53   if (always or FIB has been changed) {
54     send reachability announcements
55     (distance[1..k]) to direct neighbours
56   }

```

os vectores de distâncias são enviados periodicamente para combater eventuais perdas de mensagens. Por outro lado, se algum evento provocar uma alteração da FIB, o novo vector de distâncias é enviado a todos os vizinhos. Nas implementações concretas do algoritmo os anúncios enviados na sequência de alterações das FIBs são designados *triggered updates*. Estes anúncios devem conter necessariamente todos os destinos que foram alterados, mas também podem conter por omissão o vector de distâncias completo. Eles não são enviados imediatamente, mas só depois de um pequeno compasso de espera aleatório, que evita que se forme uma cascata sincronizada de actualizações em todos os comutadores. A listagem também omite o tratamento das mensagens *hello* que permitem a deteção do estado e do custo de atingir os vizinhos.

Um outro aspecto a ter em atenção está relacionado com o facto de que esta versão do algoritmo não usa a memorização dos anúncios recebidos dos vizinhos. Por isso, quando um canal para um vizinho se avaria, todos os destinos que o usavam são colocados à distância infinito, sem recorrer a eventuais alternativas via outros vizinhos. Esta opção é possível, na medida em que mais tarde ou mais cedo aparecerão anúncios de outros vizinhos que restabelecerão a conectividade para esses destinos, caso existam alternativas.

A mesma opção é usada quando o comutador recebe um anúncio de um comutador *i* e esse comutador é usado como vizinho (*next hop*) através de qual se atinge um dado destino *x*. Qualquer alteração da distância a que o vizinho *vê x* é imediatamente repercutida na FIB sem a preocupação, no caso da subida da distância, de procurar imediatamente outras alternativas via outros vizinhos. Tal é impossível pois os seus vectores de distância não foram guardados e pressupõe-se igualmente que essas melhores alternativas serão anunciadas mais tarde ou mais cedo, caso existam.

A seguir ficará mais claro que esta opção, para além de ser mais simples, contribui para não desencadear potenciais efeitos laterais indesejáveis.

O algoritmo de Bellman-Ford baseia-se na troca de informação entre nós vizinhos sobre os destinos que conhecem e as distâncias a que os alcançam.

É fácil de perceber e simples de incorporar num protocolo de encaminhamento, apenas obriga à existência de uma FIB que incorpore a noção de distância pelo que a informação local obrigatória é mínima. O processamento de cada mensagem é simples e linear com a dimensão dos anúncios, os quais são proporcionais ao número de destinos existentes na rede.

Infelizmente, como veremos a seguir, nem sempre o algoritmo tem um comportamento tão benigno.

### Convergência do protocolo

À primeira vista a convergência de protocolos baseados no algoritmo Bellman-Ford é relativamente rápida. Na sequência de uma alteração da FIB de um comutador, este envia novos vectores de distâncias aos vizinhos, estes processam-nos com baixo custo e, caso as suas FIBs se alterem, propagam novos vectores. O número máximo de iterações é, à primeira vista, limitado pelo diâmetro da rede. Com efeito, sempre que a alteração consiste na descida do custo de um canal (*e.g.*, passou de infinito a finito) ou no aparecimento de um novo destino (*e.g.*, foi ligado um novo comutador), o algoritmo converge para a determinação dos caminhos mais curtos, o que foi provado matematicamente pelos seus inventores.

Infelizmente, como veremos a seguir, no caso de a alteração ser no sentido contrário (*e.g.*, um canal passou de um custo finito para infinito porque avariou), a convergência pode ser muito lenta pois os comutadores podem enviar inadvertidamente vectores

de distâncias com informações falsas (*e.g.*, considerar destinos acessíveis quando na verdade já não o estão, só que o comutador em questão ainda não foi disso notificado).

A figura 16.17 permite ilustrar o problema. Na Figura 16.17 (a), à esquerda, o comutador (x) é introduzido na rede. Assim que o canal  $x - c$  ficar operacional e passar de inexistente a existente, o comutador c divulga imediatamente um vector de distâncias contendo o destino x à distância 1 (admitamos que esse é o custo de todos os canais); logo a seguir o comutador b introduz na sua FIB que o destino x é acessível via c à distância 2 e anuncia a a que o destino x está acessível à distância 2; finalmente, o comutador a actualiza a sua FIB para passar a conter a indicação de que o destino x é acessível via b à distância 3. Quaisquer anúncios futuros, num sentido ou outro, não podem introduzir alterações enquanto a configuração da rede se mantiver estável. A convergência deu-se rapidamente num número de passos limitado pelo diâmetro da rede.

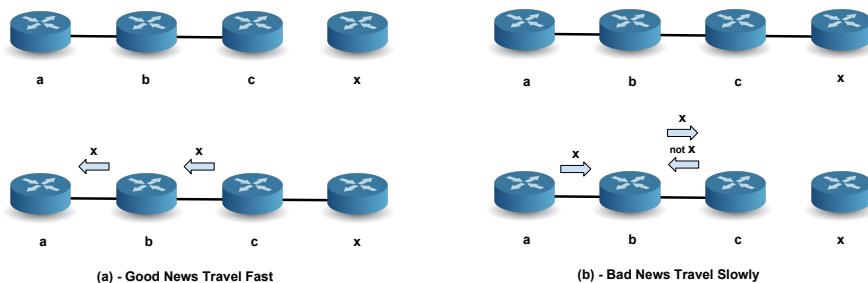


Figura 16.17: Ilustração do fenómeno “as boas notícias correm velozes, mas as más devagar” (“*good news travel fast, while bad news travel slowly*”).

Na Figura 16.17 (b), à direita, ilustra-se a situação inversa: o canal que liga c a x avariou-se e passou de um custo finito a infinito. O processo que se passou no caso (a) pode ter agora lugar no caso (b) e todos os comutadores passam a considerar que o destino x deixou de estar acessível. Infelizmente as coisas não são assim tão simples pois os anúncios não fluem só da direita para a esquerda, mas também no sentido contrário. Por outro lado, o momento em que cada comutador envia anúncios não está sincronizado com os outros. Que acontece se b anuncia a c que conhece um caminho para x com custo 2 antes de receber de c a indicação de que x deixou de estar acessível?

Não há nenhuma razão para que c não passe a acreditar que existe um outro caminho para x com custo 3 via b visto que com o algoritmo Bellman-Ford os anúncios são opacos sobre os caminhos usados para os destinos anunciados. A mesma conclusão poderia ser tomada por c caso usasse uma versão do algoritmo em que guardasse os anúncios recebidos dos outros nós e resolvesse, sempre que um custo de um caminho que está a usar subisse, ir ver se não haveria alternativa melhor via outro vizinho. A partir do momento em que c acreditasse que existe um caminho para x via b, iria anunciar-lo a b, b aumentaria o custo com que conseguia chegar a x e voltava a anunciarlo a c mas com custo maior, e assim sucessivamente. Nada pode parar este ping pong de anúncios excepto quando o custo anunciado for tão alto que os comutadores o equiparem a infinito.

Este fenómeno pode surgir quando um anúncio falso se introduz na rede simplesmente porque o “mentiroso” ainda não se apercebeu de que uma informação que recebeu deixou de ser válida. Quando o problema aparece, aparecem ciclos de pacotes e alguns canais podem deixar de ser utilizáveis pois ficam saturados pelo tráfego, quer de anúncios, quer de eventuais pacotes em ciclo. O fenómeno foi designado por “contagion para o infinito” ou “as boas notícias correm velozes, mas as más

**devagar** (“*good news travel fast, while bad news travel slowly*”).

Foram propostas e são usadas diversas abordagens que mitigam a possibilidade de esta anomalia ter lugar. A primeira destas abordagens consiste em tentar limitar a duração do ping pong. Por exemplo, com o protocolo RIP (Routing Information Protocol), ver o RFC 2453, todos os canais, seja qual for o seu débito, só podem ter custo 1, logo a métrica do custo de um caminho é equivalente ao seu comprimento. No protocolo RIP considera-se que o maior custo de um caminho é 15 pois infinito toma o valor 16. Esta solução limita o número máximo de anúncios em ping pong mas também limita o diâmetro da rede e a métrica do custo.

Uma segunda solução consiste em tentar diminuir a probabilidade de se introduzirem mentiras (ou afirmações falsas mas que um dia foram verdadeiras) e consiste numa abordagem designada *split horizon with poison reverse*. A mesma consiste em o comutador b anunciar com custo infinito a um vizinho c, todos os destinos x tais que b usa c para lá chegar. Na verdade, a informação contrária é redundante e até meio falsa pois não vale a pena estar a anunciar a c que se conhece um caminho para x que passa pelo próprio c. Esta solução resolve o problema na rede da Figura 16.17. No entanto, é fácil reconhecer que não o resolve definitivamente em qualquer rede que tenha um ciclo, como por exemplo a rede ilustrada na Figura 16.18. A sua explicação deixa-se como exercício para o leitor, ver a Secção 16.6 no final do capítulo.

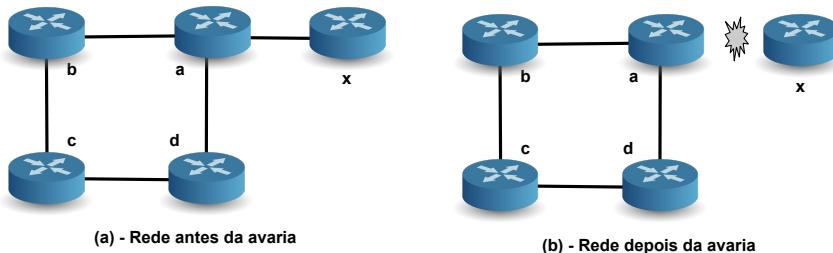


Figura 16.18: Rede onde a solução “*split horizon with poison reverse*” pode não resolver o problema da contagem para o infinito.

A terceira solução consiste em introduzir um mecanismo que tem como efeito ignorar as “boas notícias” durante algum tempo depois de se receberem “máis notícias”. De forma mais concreta, se um comutador recebe um anúncio de que o destino x passou a estar à distância infinito, ignora novos anúncios sobre o destino x durante algum tempo. O alarme que regula este compasso de espera chama-se *hold-down timer*. Esta forma de proceder dá tempo a todos os comutadores de se aperceberem que x deixou de estar acessível e evita que os outros acreditem em falsidades sobre a sua acessibilidade. Para ser efectivo, a duração deste alarme tem de ser maior que o tempo de convergência da rede.

A solução anterior tem contudo um problema, não sendo o tempo de convergência à partida conhecido, e porque se podem perder mensagens, a duração do alarme tem que ser estimada por excesso, podendo resultar num valor demasiado elevado.

O problema suplementar é que o facto de x deixar de ser acessível por um certo caminho, não quer dizer que não continue acessível por outro. Ora esta solução tem o efeito indesejável de nesse caso não permitir que o outro caminho seja usado durante o período de espera. Neste caso, a “prudência” tem um elevado custo pois, apesar de se evitar a existência de ciclos e instabilidade na rede, a convergência é lenta, ficando sempre igual a pelo menos o tempo de espera introduzido pelo *hold-down timer*. A contagem para o infinito pode também surgir quando os comutadores guardam os anúncios dos seus vizinhos e, no caso de um destino deixar de estar acessível, usem

imediatamente outro que tenham memorizado pois este pode corresponder apenas a algo que foi verdadeiro mas que já não é.

Saindo do quadro estrito da utilização de vectores de distâncias, é possível adoptar outras soluções para o problema. Uma das mais comuns é a usada pelo protocolo BGP, ver o Capítulo 17, que consiste em complementar os vectores de distâncias com a indicação, para cada destino, do caminho usado para lá chegar. Os novos vectores dizem-se então vectores de caminhos. Assim, quando um comutador recebe um anúncio de visibilidade, pode analisar o caminho usado pelo vizinho para chegar ao destino. Se o comutador receptor figurar nesse caminho, o caminho deve ser rejeitado pois introduziria um ciclo. Neste caso, o facto de se guardarem os anúncios dos diversos vizinhos não é perigoso pois estes não podem conter ciclos.

Curiosamente, só neste quadro é possível de forma segura passar a usar um protocolo que não envie periodicamente os anúncios, mas apenas quando houver alterações. Repare-se que se os anúncios forem transmitidos de forma fiável apenas quando há alterações, caso um caminho deixe de estar disponível, outro caminho alternativo não será necessariamente anunciado senão quando houver alterações que o afectem. Isso quer dizer que se se trocarem anúncios de forma fiável, então é obrigatório memorizar os anúncios recebidos dos diferentes vizinhos.

Por isso, o problema da contagem para o infinito, a transmissão de anúncios de forma fiável e a memorização dos anúncios recebidos de todos os vizinhos são opções de implementação que estão todas interligadas.

O algoritmo Bellman-Ford serviu de base a quase todos os protocolos de encaminhamento que foram desenvolvidos inicialmente para as redes de pacotes experimentais e mesmo de produção. A sua simplicidade e adequação às capacidades computacionais dos *routers* e canais, então disponíveis, tornaram-no a opção preferida pelos implementadores.

Com a experiência foi-se constatando que o fenómeno designado “**contagem para o infinito**” ou “**as boas notícias correm velozes, mas as más devagar**” (“*good news travel, fast while bad news travel slowly*”) introduz problemas delicados de convergência.

Este problema pode ser minorado introduzindo diversos mecanismos, nomeadamente: limitar o número de *hops* máximo dos caminhos, usar anúncios do tipo *split horizon with poison reverse*, e introduzir *hold-down-timers*. Estes mecanismos minoram o problema sem o resolver definitivamente.

Infelizmente, as soluções definitivas conhecidas anulam a simplicidade do algoritmo. Assim, na prática, sempre que as redes têm alguma complexidade e a sua convergência é mais crítica, os protocolos baseados no estado dos canais são os preferidos.

### Notas históricas

Protocolos baseados em vectores de distâncias foram usados pelas primeiras redes de pacotes experimentais que introduziram encaminhamento dinâmico, como a Arpanet e a Cyclades. Mais tarde a mesma filosofia foi adoptada numa rede experimental desenvolvida no Xerox Parc, onde foi usado pela primeira vez o acrônimo RIP (XNS-RIP).

O XNS-RIP serviu de inspiração para o desenvolvimento da primeira versão de RIP incluída na implementação de TCP/IP do sistema de operação Berkeley UNIX, ver a Secção 1.6. Como o código fonte deste sistema era público e podia ser reutilizado, a sua implementação de TCP/IP, incluindo a implementação de RIP, serviu de base à

implementações de TCP/IP em muitos outros sistemas, e o RIP tornou-se o protocolo de encaminhamento mais popular na Internet da época. Mesmo para ser implementado de novo, necessitava apenas de uma tabela e do processamento de duas mensagens e alguns temporizadores.

A versão inicial, ver o RFC 1058, introduziu uma versão muito simples do protocolo. Mais tarde os diferentes mecanismos descritos acima para combater o problema da contagem para infinito foram introduzidos, e as versões mais recentes, RIPv2, ver os RFCs 2453, e RIPng ou RIP para IPv6, ver o RFC 2080, passaram a inclui-los, nomeadamente *triggered updates*, *split horizon with poison reverse* e *hold-down timers*. No entanto, o protocolo continua a basear-se em UDP e anúncios periódicos *best effort* sem memorização dos anúncios recebidos.

O RIP tem a vantagem adicional de poder ser usado praticamente sem recurso a parametrização nenhuma dos *routers*, o que é mais difícil com outros protocolos. No entanto, as suas características confinam-no a redes simples, de pequena dimensão. Em redes de maior dimensão, geralmente usam-se outros protocolos.

Existem outros protocolos baseados em vectores de anúncios (de visibilidade ou de caminhos) em que o problema da contagem para o infinito foi adequadamente resolvida, como por exemplo os protocolos: EIGRP (Enhanced Interior Gateway Routing Protocol), um protocolo proprietário de um fabricante de comutadores mas transformado em aberto recentemente, ver o RFC 7868 de 2016, e o BGP (Border Gateway Protocol) que será discutido no Capítulo 17.

### **Observações finais**

De um ponto de vista meramente abstracto, *i.e.*, independentemente das opções de implementação concretas, é difícil comparar o encaminhamento baseado em estado dos canais, com o encaminhamento baseado em vectores de distâncias ou caminhos, porque as qualidades finais estão muito dependentes quer das opções concretas de implementação, quer de mecanismos complementares que são importantes em redes reais.

Em redes de pequena dimensão e sem constrangimentos significativos de convergência, não existem razões fortes contra a utilização de RIP. Em redes médias e com constrangimentos de convergência são geralmente utilizados os protocolos baseados em estado dos canais, como o OSPF ou o IS-IS. Em redes muito grandes, geralmente utilizam-se os dois tipos de protocolos mas com base em implementações sofisticadas de protocolos baseados em anúncios de caminhos, nomeadamente o protocolo BGP.

## **16.5 Resumo e referências**

### **Resumo**

O encaminhamento de pacotes é um dos problemas centrais das redes de computadores e também um dos mais complexos. Envolve várias facetas e requisitos: teóricas, para modelização e optimização da rede; algorítmicas, para concretizar essas soluções; de engenharia, para implementar um sistema complexo que responda aos requisitos e minimizar o seu custo; de monitorização, para obter dados sobre o seu funcionamento real. Adicionalmente problemas operacionais e de planeamento, para responder às necessidades do dia a dia e adaptar a rede à continua evolução dos requisitos que lhe são colocados.

Uma primeira solução de compromisso relativamente comum consiste em usar encaminhamento pelo caminho mais curto. Esta solução é aceitável quando os caminhos mais curtos estão bem dimensionados para o tráfego existente. O algoritmo de Dijkstra é um algoritmo centralizado que, dado um grafo com arcos pesados e um nó origem, calcula uma árvore de cobertura de caminhos mais curtos com raiz no nó origem.

Este algoritmo, aplicado repetidamente a cada um dos nós de comutação de pacotes de uma rede, permite determinar caminhos mais curtos de cada nó para todos os outros, ou seja permite determinar tantas árvores de cobertura quantos os nós do grafo. Seja qual for o nó que toma uma decisão de encaminhamento de um pacote, encaminhá-lo-á para o destino por um caminho mais curto.

Para o encaminhamento pelo caminho mais curto foram introduzidas diversas soluções concretas, entre as quais se destacam duas: o encaminhamento com base no **estado dos canais** (*link-state routing*), e a conhecida como **vectores de distâncias** (*distance vectors*) ou **anúncios de visibilidade** (*reachability announcements*).

A técnica de encaminhamento pelo caminho mais curto com base no estado dos canais (*link-state routing*) baseia-se na replicação por todos os comutadores, por difusão fiável, de todas as adjacências existentes na rede. Desta forma, cada comutador adquire a visão do grafo que caracteriza a rede e, usando o algoritmo de Dijkstra, pode calcular os melhores caminhos para todos os destinos existentes e assim construir a sua FIB.

Sempre que têm lugar modificações do estado de qualquer componente da rede, os comutadores que as detectam difundem actualizações, também designadas SLAs (*Link State Announcements* ou *Adjacency Announcements*) e, logo que a difusão fiável se completar, as actualizações são repercutidas nas FIBs de todos os comutadores e a rede converge e volta a um estado estável.

O encaminhamento por vectores de distâncias, ou usando o algoritmo de Bellman-Ford, consiste na troca entre comutadores vizinhos de vectores de distâncias ou anúncios de visibilidade. Estas mensagens indicam aos vizinhos, os destinos e as distâncias que cada comutador conhece. Ao receber anúncios dos vizinhos, cada comutador pode escolher entre as diversas alternativas aquelas que correspondem aos caminhos mais curtos.

O algoritmo é fácil de perceber e simples de incorporar num protocolo de encaminhamento, apenas obriga à existência de uma FIB que inclua a noção de distância pelo que a informação local obrigatória é mínima. O processamento de cada mensagem é simples e linear com a dimensão dos anúncios, os quais são proporcionais ao número de destinos diferentes existentes na rede.

Com a experiência foi-se constatando que o fenómeno designado “**contagem para o infinito**” ou “**as boas notícias correm velozes, mas as más devagar**” (“*good news travel fast while bad news travel slowly*”) introduz problemas delicados de convergência no algoritmo de Bellman-Ford. Este fenómeno pode ser minorado introduzindo diversos mecanismos, nomeadamente: limitar o número de *hops* máximo dos caminhos, usar anúncios do tipo *split horizon with poison reverse*, e introduzir *hold-down-timers*. Estes mecanismos minoram o problema sem o resolver definitivamente.

Soluções mais completas anulam a simplicidade do algoritmo. Assim, na prática, sempre que as redes têm alguma complexidade e a sua convergência é mais crítica, os protocolos baseados no estado dos canais são os preferidos em redes de média dimensão.

Em redes de pequena dimensão e sem constrangimentos significativos de convergência, não existem razões fortes contra a utilização de RIP, um protocolo simples baseado no algoritmo de Bellman-Ford. Em redes médias e com constrangimentos de convergência são geralmente utilizados os protocolos baseados em estado dos canais, como o OSPF ou o IS-IS. Em redes muito grandes, geralmente utilizam-se os dois tipos de protocolos mas com base em implementações sofisticadas de protocolos baseados em anúncios de caminhos, como o protocolo BGP, ver o Capítulo 17.

Alguns dos termos introduzidos no capítulo são a seguir passados em revista. Entre parêntesis figura a tradução mais comum em língua inglesa.

### Principais conceitos

**Custo de um caminho** (*path cost*) Define-se custo de um caminho como sendo a soma dos custos dos diferentes canais atravessados por esse caminho.

**Distância** (*distance*) Define-se distância entre dois nós como sendo o custo do um caminho de menor custo entre esses nós. A distância é relativa a um conjunto de caminhos conhecidos.

**Caminho mais curto** (*shortest path*) Define-se caminho mais curto entre dois nós como sendo um caminho mais curto entre todos os caminhos que os ligam.

**Algoritmo de Dijkstra** (*Dijkstra algorithm*) é um algoritmo centralizado que, dado um grafo com arcos pesados e um nó origem, calcula uma árvore de cobertura de caminhos mais curtos com raiz no nó origem.

**Comutador inteligente de pacotes** (*packet router*) É um comutador de pacotes que executa um protocolo de encaminhamento, *i.e.*, um algoritmo distribuído de troca de informação de encaminhamento entre os comutadores, que permite aos mesmos adaptarem-se dinamicamente ao estado da rede e realizarem o encaminhamento dos pacotes em função de uma estratégia de optimização como por exemplo encaminhar pelo caminho mais curto.

**Métrica de um protocolo** (*routing protocol metric*) Numa rede real cada canal tem um custo. A métrica de um protocolo é a forma concreta que este usa para determinar o custo dos canais. Uma métrica simples é aquela que associa um custo constante a cada canal, por exemplo 1. As métricas mais sofisticadas tentam associar a um canal um custo função do tempo de trânsito dos pacotes que o atravessam. Neste caso uma métrica pode ser inversamente proporcional ao débito, proporcional ao tempo de propagação, proporcional à intensidade do tráfego ou ainda proporcional ao custo monetário do canal.

**Encaminhamento pelo caminho mais curto** (*shortest path routing*) Forma de encaminhamento de pacotes numa rede que consiste em escolher sempre caminhos mais curtos para os pacotes. Esta solução é aceitável quando os caminhos mais curtos estão bem dimensionados para o tráfego existente na rede.

**FIB** (*Forwarding Information Base*) Estrutura de dados que contém a informação necessária para um comutador de pacotes escolher qual a interface pela qual tem de transmitir um pacote a caminho do seu destino. Geralmente, a FIB implementa um simples mapeamento de endereços de destino em interfaces e distâncias e é independente dos protocolos de encaminhamento usados.

**RIB** (*Routing Information Base*) Estrutura de dados de um comutador associada a um protocolo de encaminhamento específico, que contém a informação, necessária para a computação de uma FIB, obtida por esse protocolo.

**Encaminhamento com base no estado dos canais** (*link state routing*) Estratégia de encaminhamento que consiste em os comutadores trocarem informação, através de um protocolo de difusão fiável, que lhes permite adquirirem todas as adjacências de cada comutador. Desta forma, cada comutador adquire uma visão completa do estado global da rede. Esse estado é idêntico em todos os comutadores e permite a cada um deles, usando por exemplo o algoritmo de Dijkstra, computar a sua FIB.

**LSA** (*Link State Announcement*) São as mensagens com a informação de adjacência transmitidas por cada comutador, que são replicadas por difusão fiável e que servem de base aos protocolos de encaminhamento com base no estado dos canais. Os LSAs podem ser de vários tipos e servirem para transportar vários tipos de informações entre os comutadores da rede.

**LSDB** (*Link State Data Base*) Estrutura de dados de um comutador que contém o conjunto dos LSAs recebidos por cada comutador a participar em encaminhamento com base no estado dos canais.

**Encaminhamento com vectores de distâncias** (*distance vector routing*) Estratégia de encaminhamento que consiste em pressupor que cada comutador conhece

uma forma de encaminhar pacotes dirigidos a alguns destinos e que divulga a mesma aos seus vizinhos directos através de vectores ou anúncios de distâncias. Estes anúncios consistem num conjunto de pares de (destino, distância), que permitem ao receptor, sabendo a que distância está o vizinho emissor do anúncio, saber se o deve seleccionar para encaminhar pacotes para aquele destino por um caminho mais curto.

**Algoritmo de Bellman-Ford** Algoritmo que está na base do encaminhamento com vectores de distâncias, designado com base no nome dos seus inventores.

**Anúncios de visibilidade** (*reachability announcements*) É outra designação dada aos vectores de distâncias na forma de encaminhamento designada como encaminhamento com base em vectores de distâncias.

**Convergência** (*convergence*) É o tempo necessário para, na sequência de uma qualquer alteração de estado dos nós ou dos canais de uma rede, todos os nós de comutação tomarem em consideração essas alterações, reconfigurarem a rede, e voltarem a colocá-la em funcionamento de acordo com a nova configuração. Os protocolos com base no estado dos canais convergem rapidamente. Os protocolos com base em vectores de distância convergem também rapidamente se a alteração de estado consistir numa diminuição de distância. No entanto, quando se dá um aumento da distância, em particular quando um destino deixa de ser acessível, pode produzir-se uma situação anómala em que alguns comutadores introduzem na rede uma informação que já não é válida, mas que outros tomam como certa. Em casos particulares, isso pode dar uma situação oscilatória e transitória em que se formam ciclos de encaminhamento e que é designada por contagem para o infinito (*counting to infinity*).

**Split horizon with poison reverse** Num protocolo de encaminhamento com base em vectores de distâncias, os comutadores anunciam aos seus vizinhos um destino à distância infinito caso usem esse vizinho para atingir o destino. Esta técnica permite minorar o aparecimento da anomalia contagem para o infinito. No entanto, não a resolve definitivamente.

**Hold-down-timers** Num protocolo de encaminhamento com base em vectores de distâncias, os comutadores colocam em quarentena os destinos que passaram a estar à distância infinito e não aceitam anúncios sobre esses destinos durante o período de tempo *hold-down timer*. O objectivo é permitir a todos os comutadores receberem o anúncio com as “más notícias” e não introduzirem, por inadvertência, um anúncio falso sobre a possibilidade de enviarem pacotes para o destino que de facto se tornou inacessível. Este mecanismo resolve a anomalia contagem para o infinito à custa de cortar o acesso durante o período de tempo *hold-down*.

**RIP** (*Routing Information Protocol*) Um dos primeiros protocolos de encaminhamento introduzidos foi o protocolo RIP. Trata-se de uma protocolo do tipo encaminhamento com base em vectores de distâncias que implementa a forma mais simples de realizar esta estratégia de encaminhamento. Tem como vantagem a simplicidade mas só é realista em redes de pequena dimensão e onde se tolerem tempos de convergência elevados.

**IS-IS** (*Intermediate System - Intermediate System*) Primeiro protocolo de encaminhamento que usou o método de encaminhamento designado como encaminhamento com base no estado dos canais. Pode ser usado em inúmeros contextos e diferentes redes pois dispõe de um sistema extensível de LSAs que pode adaptar-se à difusão de vários tipos de informações em diversos tipos de redes.

**OSPF** (*Open Shortest Path First*) Protocolo de encaminhamento inspirado do IS-IS mas especializado no encaminhamento em redes TCP/IP.

**EIGRP** (*Enhanced Interior Gateway Routing Protocol*) Protocolo de encaminhamento com base em vectores de distâncias que inclui métodos sofisticados de evitar a anomalia contagem para o infinito, assim como suporta várias métricas de encaminhamento. Trata-se de um protocolo de encaminhamento proprietário de um fabricante de comutadores que só recentemente foi tornado público.

**BGP** (*Border Gateway Protocol*) Protocolo de encaminhamento com base em vectores de caminhos que transmite anúncios de visibilidade que incluem o caminho completo para chegar ao destino. Por essa razão, previnem a introdução de ciclos de encaminhamento e anulam a contagem para o infinito. Para além disso, os anúncios são memorizados pelos comutadores e transmitidos de forma fiável sobre TCP, pelo que só são transmitidas as alterações quando estas têm lugar. Trata-se de um protocolo complexo que só é usado em redes de grande dimensão de operadores ou em redes de acesso ligadas a diferentes operadores.

## Referências

Os livros [Bellman, 1957; Ford and Fulkerson, 1962] são hoje em dia dois clássicos e ambos apresentam o algoritmo Bellman-Ford. O primeiro numa versão centralizada e o segundo numa versão distribuída. O artigo [Dijkstra, 1959] apresenta o algoritmo de Dijkstra para cálculo de caminhos mais curtos.

O livro [Huitema, 1995] é uma referência clássica que apresenta uma introdução aos protocolos RIP e OSPF e discute as suas bases algorítmicas. Todos os livros bem conhecidos de introdução às redes de computadores cobrem os algoritmos cobertos neste capítulo, nomeadamente [Tanenbaum and Wetherall, 2011; Peterson and Davies, 2012; Kurose and Ross, 2013]. O problema da convergência do encaminhamento com base no estado dos caminhos em redes complexas é analisado em detalhe em [Francois et al., 2005].

Existem inúmeros livros com descrições dos protocolos citados neste capítulo. O livro [Doyle and Carroll, 2006] é considerado um dos melhores, que inclui também indicações concretas sobre como os parametrizar em comutadores de pacotes populares.

## Apontadores para informação na Web

- <http://en.wikipedia.org> – Os artigos sobre RIP, OSPF, IS-IS e BGP disponíveis na wikipedia na versão inglesa constituem boas introduções, relativamente sintéticas, a estes protocolos. Têm a vantagem adicional de listar todos os RFCs relevantes.
- <http://ietf.org/rfc.html> – É o repositório oficial dos RFCs, nomeadamente dos citados neste capítulo: RFC 1058 (RIPv1), RFC 2080 (RIPng - IPv6), RFC 2328 (OSPF IPv4), RFC 2453 (RIPv2), RFC 5340 (OSPF IPv6) e RFC 7868 (EIGRP).

## 16.6 Questões para revisão e estudo

1. Defina encaminhamento estático e encaminhamento dinâmico.
2. Indique que factores se devem tomar em consideração para conceber um protocolo ideal de encaminhamento dinâmico.
3. Defina encaminhamento pelo melhor caminho.
4. Diga se é verdade ou mentira ou comente:
  - (a) Um protocolo de encaminhamento do tipo vector de distâncias pode tratar os canais ponto-a-ponto e os canais multi-ponto exactamente da mesma forma.

- (b) Um protocolo de encaminhamento do tipo estado dos canais trata os canais ponto-a-ponto e os multi-ponto exactamente da mesma forma.
  - (c) Um protocolo de encaminhamento do tipo estado dos canais trata a reparação de uma partição de uma rede como um caso normal de anúncio do estado do canal que acabou de ser reparado.
  - (d) Um protocolo de encaminhamento do tipo estado dos canais suporta bem redes de grande dimensão, com muitos canais e com muitos destinos possíveis.
  - (e) Um protocolo de encaminhamento do tipo estado dos canais converge rapidamente quando há canais que se avariam.
  - (f) Um protocolo de encaminhamento do tipo estado dos canais converge rapidamente quando há canais avariados que são reparados.
  - (g) Um protocolo de encaminhamento do tipo estado dos canais assegura que um anúncio de um novo destino é propagado a todos os comutadores num período de tempo proporcional ao número de comutadores da rede.
  - (h) Um protocolo de encaminhamento do tipo estado dos canais assegura que um anúncio de um novo destino é propagado a todos os comutadores num período de tempo proporcional ao número de canais da rede.
  - (i) Um protocolo de encaminhamento do tipo estado dos canais assegura que um anúncio de um novo destino é propagado a todos os comutadores num período de tempo proporcional ao diâmetro da rede.
  - (j) Se vários canais ficarem inoperacionais, a rede pode decompor-se em várias partições. Quando uma partição de rede se repara, pode surgir o problema contagem para o infinito associado a um protocolo de encaminhamento de tipo vector de distâncias.
  - (k) Um protocolo de encaminhamento do tipo vector de distâncias suporta bem redes de grande dimensão e com muitos destinos possíveis.
  - (l) Um protocolo de encaminhamento do tipo vector de distâncias converge rapidamente quando há canais que se avariam.
  - (m) Um protocolo de encaminhamento do tipo vector de distâncias converge rapidamente quando há canais avariados que são reparados.
  - (n) Um protocolo de encaminhamento do tipo vector de distâncias assegura que um anúncio de um novo destino é propagado a todos os comutadores num período de tempo proporcional ao número de comutadores da rede.
  - (o) Um protocolo de encaminhamento do tipo vector de distâncias assegura que um anúncio de um novo destino é propagado a todos os comutadores num período de tempo proporcional ao número de canais da rede.
  - (p) Um protocolo de encaminhamento do tipo vector de distâncias assegura que um anúncio de um novo destino é propagado a todos os comutadores num período de tempo proporcional ao diâmetro da rede.
  - (q) Se vários canais ficarem inoperacionais, a rede pode decompor-se em várias partições. Quando uma partição de rede se repara, pode surgir o problema contagem para o infinito associado a um protocolo de encaminhamento de tipo vector de distâncias.
5. Diga se são verdade ou mentira as seguintes afirmações produzidas no contexto de uma rede que está a usar um protocolo de encaminhamento do tipo estado dos canais. Justifique a sua resposta.
- (a) Se juntarmos à rede mais nós, o tempo de execução do algoritmo de Dijkstra aumentará apenas nos nós vizinhos dos novos nós.

- (b) Cada comutador inunda a rede para enviar para todos os outros a topologia de toda a rede que conhece.
  - (c) Cada comutador inunda a rede para enviar para todos os outros o estado dos canais a que está ligado directamente.
  - (d) Cada comutador calcula árvores de menores custos entre si e todos os outros comutadores.
  - (e) O algoritmo de Dijkstra pode introduzir o problema designado contagem para o infinito.
  - (f) Quando há uma alteração do estado de um canal, o comutador que lhe está ligado espera 120 segundos antes de decidir enviar um anúncio de mudança de estado pois desta forma agrupa anúncios, poupa mensagens e não inunda a rede com alarmes inúteis.
6. O encaminhamento diz-se simétrico se os canais atravessados de A para B são os mesmos, mas por ordem inversa, que os canais usados de B para A. Suponha que numa dada rede só existe um caminho mais curto entre cada dois comutadores. Diga se é verdade ou mentira:
- (a) Nessa rede, caso a métrica do protocolo de encaminhamento seja o número de *hops*, então todo o encaminhamento é simétrico.
  - (b) Nessa rede o encaminhamento é simétrico independentemente da métrica do protocolo de encaminhamento.
  - (c) Nessa rede, caso a métrica do protocolo de encaminhamento seja o inverso do débito dos canais, existe uma situação particular em que o encaminhamento é simétrico.
7. Considere uma rede baseada em canais ponto-a-ponto que interligam vários comutadores. Imagine que um dos canais tem um problema que faz com que esteja constantemente a transitar do estado operacional (capaz de transmitir dados) para o estado inoperacional (sem poder transmitir dados) e vice-versa, a um ritmo relativamente rápido (*e.g.*, a cada 5 segundos).
- (a) De que forma uma transição de estado do canal é tomada em consideração por um protocolo de encaminhamento do tipo estado dos canais?
  - (b) De que forma uma transição de estado do canal é tomada em consideração por um protocolo de encaminhamento do tipo vector de distâncias?
  - (c) Imagine uma forma de minorar os custos induzidos por esta instabilidade em ambos os protocolos.
8. Um ciclo no encaminhamento (*routing loop*) ocorre, por exemplo, quando um comutador A ligado directamente a um comutador B julga que o melhor caminho para atingir o destino D é via B e B julga que o melhor caminho para atingir o mesmo destino é via A.
- (a) Quais os inconvenientes de um ciclo no encaminhamento?
  - (b) A situação indicada acima é a única em que um ciclo no encaminhamento pode ocorrer?
  - (c) Que mecanismo existe por omissão nos cabeçalhos dos pacotes para evitar os inconvenientes desta anomalia quando ela ocorre?
  - (d) Em que situação pode ocorrer um ciclo no encaminhamento quando se usa um protocolo de encaminhamento do tipo estado dos canais?

- (e) Em que situação pode ocorrer um ciclo no encaminhamento quando se usa um protocolo de encaminhamento do tipo vector de distâncias?
9. Considere a rede da Figura 16.18 e indique uma sequência de eventos que pode levar ao aparecimento de uma anomalia do tipo contagem para o infinito se o canal que liga os nós **x** e **a** se avariar.
10. Existe uma situação particular em que dois comutadores a usarem um protocolo baseado no estado dos canais têm de sincronizar as respectivas LSA Data Bases. Qual é esse caso?
11. Considere a rede da Figura 16.19. O encaminhamento é do tipo vector de distâncias com anúncios periódicos.

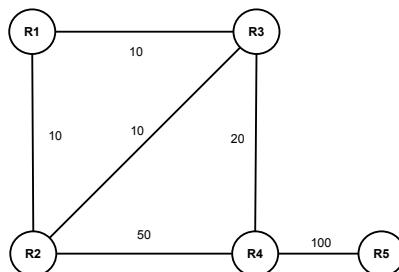


Figura 16.19: Rede do exercício 11

- (a) Indique as FIBs (tabelas de encaminhamento) dos comutadores R2, R3, R4 e R5.
- (b) A rede já convergiu e todos os canais estão operacionais. Descreva os anúncios que R2, R3, R4 e R5 enviam periodicamente sabendo que o protocolo não usa *split horizon with poison reverse*.
- (c) A rede já convergiu e todos os canais estão operacionais. Descreva os anúncios que R4 e R5 enviam periodicamente sabendo que o protocolo usa agora *split horizon with poison reverse*.
- (d) A rede não usa *hold down timers*. Que anúncios fará R4 quando o canal para R5 falhar se os comutadores estiverem a usar *split horizon with poison reverse*.
- (e) A rede não usa *hold down timers*. Descreva uma situação em que pode surgir uma anomalia de contagem para o infinito quando o canal para R5 falhar e os comutadores estiverem a usar *split horizon with poison reverse*.
- (f) Se a rede usar *hold down timers* que acontece quando o canal de R4 para R5 falhar?
12. Considere a rede da Figura 16.20. O encaminhamento é do tipo vector de distâncias com anúncios periódicos. O custo de um canal com a capacidade de 10 Mbps é 100, e o de um com a capacidade de 100 Mbps é 10. O tempo de propagação de extremo a extremo de cada canal é de 100 ms.
- (a) O protocolo já convergiu. Qual o custo de enviar pacotes de R1 para R5 do ponto de vista do algoritmo de encaminhamento?
- (b) Como é o vector de distâncias que contém um anúncio de R4 dirigido a R5 quando todos os canais estão operacionais ?

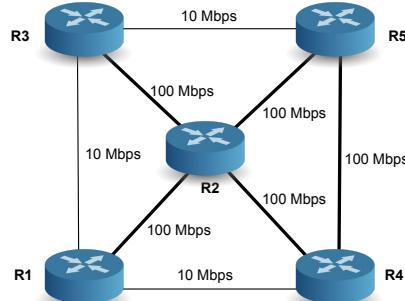


Figura 16.20: Rede do exercício 12

- (c) Admitindo que os anúncios são transmitidos instantaneamente sem demora ou tempo de transmissão relevante, e que o tempo de processamento dos mesmos é desprezável, qual o tempo máximo que levam todos os routers a ficar a conhecer o novo encaminhamento quando o canal que liga R5 a R4 for abaixo?
- (d) Idem quando o canal avariado for o o canal que liga R1 a R4.
13. Considere a rede da Figura 16.21 que interliga os comutadores indicados com os custos assinalados nos canais. A rede usa um protocolo de encaminhamento baseado no algoritmo Bellman-Ford com anúncios periódicos e sem memorização dos anúncios.

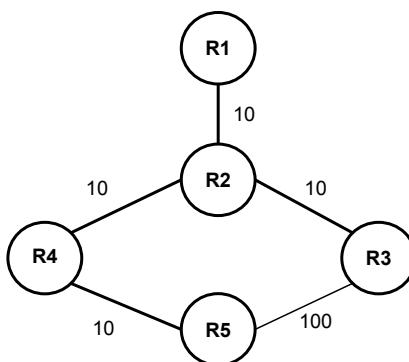


Figura 16.21: Rede do exercício 13

- (a) A rede já convergiu e todos os canais estão operacionais. Descreva os anúncios que R3 e R5 enviam aos seus vizinhos sabendo que o protocolo não usa *split horizon with poison reverse*.
- (b) A rede já convergiu e todos os canais estão operacionais. Descreva os anúncios que R3 e R5 enviam aos seus vizinhos sabendo que o protocolo usa *split horizon with poison reverse*.
- (c) A rede já convergiu mas mais tarde o canal entre R1 e R2 avaria e R2 faz uma anúncio de que a distância a R1 passou infinito. Sabendo que

- o protocolo não usa *split horizon with poison reverse*, é possível entrar-se num ciclo de contagem para o infinito nesta rede? Justifique.
- (d) A rede já convergiu mas mais tarde o canal entre R1 e R2 avaria e R2 faz um anúncio de que a distância a R1 passou infinito. Sabendo que o protocolo usa *split horizon with poison reverse*, é possível entrar-se num ciclo de contagem para o infinito nesta rede? Justifique.
  - (e) Se a rede usar *hold down timers* que acontece quando o canal de R1 para R2 falha?
  - (f) Se a rede usasse um protocolo do tipo estado dos canais seria possível acontecer a anomalia contagem para o infinito? Seria necessário usar *hold down timers*?
14. Considere uma rede baseada num protocolo de encaminhamento do tipo vector de distâncias.
- (a) Como é que um comutador pode tentar atrair para si todos os pacotes da rede?
  - (b) De que forma um comutador pode tentar obrigar a rede a parar de encaminhar pacotes para os diferentes destinos?
  - (c) Quais as implicações dos ataques anteriores do ponto de vista da segurança?
15. Considere uma rede baseada num protocolo de encaminhamento do tipo estado dos canais.
- (a) Como é que um comutador pode tentar atrair para si todos os pacotes da rede?
  - (b) De que forma um comutador pode tentar obrigar a rede a parar de encaminhar pacotes para os diferentes destinos?
  - (c) Quais as implicações dos ataques anteriores do ponto de vista da segurança?
16. Considere um protocolo de encaminhamento que usa uma métrica exclusivamente baseada no tempo de trânsito. No momento de tomar a decisão de encaminhamento, o comutador considera o tempo de trânsito assinalado na FIB mas também o tamanho da fila de espera dos canais pelos quais pode transmitir os pacotes.
- (a) Descreva o processo de tomada de decisão de encaminhamento pelo comutador.
  - (b) Numa dada rede dois comutadores estão ligados directamente por dois canais paralelos com o mesmo tempo de trânsito. Como realiza o comutador o encaminhamento quando o *next hop* é esse vizinho e as filas de espera envolvidas têm comprimentos distintos?
  - (c) Questão idêntica à anterior mas as filas de espera envolvidas estão sempre vazias?
  - (d) Que efeito tem esta forma de encaminhamento sobre a ordem com que os pacotes são entregues ao destino? Acha isso bom ou mau?
  - (e) Existem outras razões para não se adoptar esta métrica?

17. Numa rede de computadores, organizada como uma malha com ciclos, existem vários caminhos possíveis de cada comutador até cada destino. Se a rede tiver muitos ciclos e um grau elevado de redundância, existem muitos desses caminhos e alguns terão custos idênticos. Um protocolo de encaminhamento capaz de utilizar simultaneamente vários caminhos para o mesmo destino com o mesmo custo, diz-se um protocolo de encaminhamento multi-caminho com custo igual (*equal cost multi-path*).
- Indique uma vantagem de no caso particular em que existem vários caminhos com o mesmo custo explorar alternativamente esses vários caminhos.
  - Algumas métricas consideram que cada salto (*hop*) entre dois comutadores tem sempre custo 1 seja qual for a capacidade da ligação e não reflectem de forma proporcional o tempo de trânsito de um caminho. Por exemplo, um caminho entre os computadores A e B tem tempo de trânsito de extremo a extremo de 10 milissegundos e outro, entre os mesmos computadores, tem tempo de trânsito de 100 milissegundos. No entanto, se ambos os caminhos tiverem o mesmo número de *hops*, a função de custo indicará que têm o mesmo custo e poderiam ser usados para encaminhamento multi-caminho. Qual o resultado dessa utilização?
  - Suponha que uma aplicação multimédia interactiva sobre UDP entre os computadores A e B está a utilizar uma rede com as características indicadas na alínea anterior e que os pacotes entre A e B estão a seguir caminhos alternativos: o 1º pacote vai por um caminho, o 2º por outro, o 3º pelo caminho do 1º, o 4º pelo caminho do 2º, etc. Que parâmetro dessa aplicação teria de ser regulado para que não houvesse uma repercussão demasiado negativa do encaminhamento multi-caminho neste cenário?
18. No protocolo OSPF, os LSAs são passados aos vizinhos de forma fiável (com um protocolo do tipo *stop & wait*) que implica que as relações entre vizinhos sejam sempre ponto-a-ponto, mesmo quando o canal que os liga é multi-ponto. No protocolo RIP, os anúncios são passados aos vizinhos usando mensagens multi-ponto quando o canal é multi-ponto. O protocolo RIP introduz fiabilidade na transmissão dos anúncios repetindo-os periodicamente. Analise se seria preferível o OSPF adoptar uma solução semelhante ao RIP, e o RIP adoptar uma solução semelhante ao OSPF, sempre que os vizinhos estão ligados por canais em difusão.
19. É possível modificar o algoritmo Bellman-Ford para passar a incluir nos anúncios o caminho usado para o destino (os vectores de distâncias passam a chamar-se vectores de caminhos). Por exemplo, o anúncio inclui destinos, distâncias e os caminhos (uma lista de identificadores de comutadores) usados para os alcançar. Inicialmente, um comutador apenas consegue anunciar os caminhos para os destinos a que dá acesso directamente. Mas à medida que vai recebendo anúncios, não só enriquece a sua RIB com mais destinos e caminhos, como os envia aos seus vizinhos. Existem na rede  $n$  destinos,  $c$  comutadores e  $k$  canais. O diâmetro máximo também é conhecido e é  $d_{max}$ .
- Indique uma estrutura de dados adequada para suportar a RIB.
  - Indique uma estrutura de dados adequada para suportar um anúncio.
  - Indique a complexidade espacial da RIB e compare-a com a complexidade espacial da FIB do algoritmo de Bellman-Ford.
  - Quais as vantagens e defeitos desta variante do algoritmo de encaminhamento?

20. Um ponto de partida para conceber um algoritmo de encaminhamento para uma rede com ciclos, que só tem canais ponto-a-ponto, poderia ser o seguinte: cada vez que um comutador recebe um pacote, encontra no seu cabeçalho o endereço do emissor inicial, assim como o número de *hops* percorridos desde a origem e sabe qual a interface pela qual o pacote chegou. Com estas informações o comutador pode ir enriquecendo a sua FIB.
- Indique como se poderia chamar essa técnica por analogia com outro protocolo de encaminhamento.
  - Como actualiza a FIB um comutador quando recebe um pacote?
  - O comutador acabou de receber um pacote dirigido a um destino para o qual ainda não existe nenhuma entrada na FIB. Como deve processar o pacote? Que requisitos é necessário garantir para que a forma de proceder indicada seja realista?
  - Esta técnica pressupõe ou não que a rede tem simetria, *i.e.*, que o melhor caminho de A para B é simétrico do melhor caminho de B para A?
  - Admita que a carga ou a configuração da rede vai variando. As soluções indicadas nos parágrafos anteriores ainda são adequadas? Como se poderia o algoritmo adaptar a essas alterações?
21. No quadro do encaminhamento baseado no estado dos canais, um comutador recebeu de um vizinho um LSA que o vizinho está a difundir por inundação. Essa mensagem, da classe LSA, pode ser manipulada pelos seguintes métodos:
- `int getRouter()` – devolve o identificador do comutador que gerou originalmente este LSA
  - `int getSeq()` – devolve o número de sequência do LSA
  - `void flood()` – faz o *flooding* do LSA para os outros comutadores
  - `void sendACK()` – envia um ACK ao vizinho que enviou o LSA
  - `void sendNACK(LSA newerLSA)` – envia um NACK ao vizinho que enviou o LSA com a versão mais recente do mesmo LSA.

A tabela lsadb, da classe LSADatabase, contém em cada comutador o estado da rede *i.e.*, o conjunto dos LSAs mais recentes recebidos. Esta tabela pode ser manipulada pelos seguintes métodos:

- `LSA getLSA(int c)` – devolve o mais recente lsa gerado pelo comutador c ou `null` se não existe nenhum
- `void putLSA( LSA lsa)` – insere um lsa na tabela e assegurando que existe apenas o lsa mais recente associado a cada comutador.

Escreva o código do método `processReceivedLSA` executado pelos comutadores quando recebem um LSA de um vizinho para assegurarem a difusão fiável dos LSAs por toda a rede.

```
void processReceivedLSA ( LSA lsa, LSADatabase lsadb ) { ... }
```

22. Numa rede, os comutadores têm vários vizinhos, a cada um dos quais estão ligados directamente por um canal distinto. Por hipótese, todos os destinos existentes na rede e todos os comutadores são identificados por uma letra: A,

B, C, ..., Z. As interfaces que ligam um comutador aos vizinhos têm por identificador o identificador do vizinho.

Um comutador envia um pacote `p` para o vizinho X usando a função `sendTo( p, X )`. Um comutador tem acesso ao seu identificador através do método `myID()`. Um pacote `p` tem os seguintes campos: `p.source`, `p.destination`, `p.ttl` e `p.data` com os significados óbvios.

Um comutador tem uma FIB `fib` que suporta vários métodos, entre os quais:

- `fib.getGateway( destination )` - devolve o vizinho que encaminha para `destination` ou `null` se `fib` desconhece `destination`
- `fib.getDistance( destination )` - devolve o custo do caminho deste comutador até `destination` ou `infinity` se `fib` desconhece `destination`
- `fib.setFIBEntry( destination, distance, gateway )` - actualiza a entrada de `fib` referente a `destination`.

- (a) Escreva o pseudo código usado pelo comutador para processar um pacote `p` que acabou de receber. Se o comutador não sabe ou não pode processar o pacote, deve ignorá-lo.
  - (b) Escreva o pseudo código usado pelo comutador para processar um pacote `p` que acabou de receber do vizinho V. Se o comutador não conseguir processar o pacote ignora-o, mas gera um pacote `error` dirigido à origem do pacote `p`. Como deveria este pacote de erro ser processado por V?
  - (c) Na rede usa-se o algoritmo Bellman-Ford para actualizar as tabelas de encaminhamento. O comutador usa a versão *split horizon with poison reverse* do algoritmo. Escreva o pseudo código usado pelo comutador para gerar o anúncio que vai enviar ao vizinho V.
  - (d) Na rede usa-se o algoritmo Bellman-Ford para actualizar as tabelas de encaminhamento. O comutador usa a versão *split horizon with poison reverse* do algoritmo. Escreva o pseudo código usado pelo comutador para processar um anúncio que acabou de receber do vizinho V à distância `Vcost`.
23. Considere uma rede com  $n$  comutadores e  $c$  canais ponto-a-ponto. Existem duas opções para protocolo de encaminhamento para essa rede: RIP ou OSPF. Utilize nas questões a seguir a notação  $O(n)$  para indicar que uma grandeza é proporcional a  $n$ .
    - (a) Compare em cada uma das opções a memória ocupada em cada comutador.
    - (b) Compare em cada uma das opções a carga da CPU em cada comutador sempre que se liga um novo comutador e quando um canal se avaria.
    - (c) Compare em cada uma das opções a carga da rede em número de mensagens quando a rede está estável.
    - (d) Compare em cada uma das opções a carga da rede em número de mensagens sempre que se liga um novo comutador e quando um canal se avaria.



## *Capítulo 17*

---

### *Interligação de redes - protocolo IP*

---

*An expert is a man who made all the mistakes, which can be made, in a very narrow field.*

– Autor: Niels Bohr

Nos capítulos anteriores vimos como funcionam internamente algumas redes particulares, nomeadamente redes baseadas em canais de difusão, redes que usam protocolos de encaminhamento baseados em inundação e algoritmos e protocolos para encaminhamento pelo melhor caminho. Nenhuma dessas soluções particulares tomada isoladamente é adequada à Internet como um todo. Com efeito, a Internet é formada pela interligação de um grande conjunto de redes particulares: redes domésticas, redes institucionais, redes de acesso e vários tipos de redes de trânsito, ver a Secção 1.3 e ver a Secção 4.4.

Os bons princípios de desenho de um sistema complexo, ver a Secção 4.1, recomendam a sub-divisão do mesmo em partes separadas e isoladas, e a definição de interfaces entre as mesmas que permitam a sua independência. Do ponto de vista do encaminhamento, a Internet é sub-dividida em diferentes redes, que formam uma partição da mesma, *i.e.*, essas diferentes redes são independentes e sem sobreposições. No entanto, essas redes estão interligadas através de múltiplas ligações, ver a Secção 4.4. O termo técnico usado para um conjunto de redes interligadas é um *internetwork*.

A independência das diferentes partes interligadas de um sistema depende em larga medida da correcta definição de interfaces. Na Internet, do ponto de vista do encaminhamento, as principais interfaces são o protocolo IP, que define o endereçamento, o formato do cabeçalho dos pacotes e o significado e processamento dos seus diferentes campos, assim como os protocolos de encaminhamento entre as diferentes redes. Adicionalmente, os bons princípios recomendam a independência das partes, o que implica que os protocolos de encaminhamento entre as diferentes redes deve deixar liberdade às mesmas para esconderem as soluções de encaminhamento que usam internamente<sup>1</sup>.

Neste capítulo vamos ver como são definidas e funcionam as interfaces entre as redes que formam a Internet. Começaremos por analisar como é definido o sistema de endereçamento da Internet. Uma faceta fundamental que abordaremos é a interligação entre endereçamento e encaminhamento e como o endereçamento condiciona as

<sup>1</sup> Este princípio também recomendaria que o endereçamento e as interfaces usadas internamente pelas partes também fosse escondida do exterior, ver [Day, 2008] por exemplo. No entanto, esta faceta foi ignorada na definição do protocolo IP.

formas de encaminhamento usadas. Esta relação é delicada e merece uma explicação relativamente longa.

A seguir vamos estudar o protocolo IP. Este protocolo é a principal interface entre os computadores e as redes e entre as diferentes redes interligadas. No entanto, em muitas redes, o protocolo IP é também a principal interface entre as diferentes componentes internas das mesmas. Veremos então que o conceito de rede é na verdade recursivo.

Para que uma rede baseada no protocolo IP funcione, são necessários um conjunto de protocolos auxiliares. A sua apresentação será feita mais adiante.

Existem actualmente duas versões do protocolo IP. A versão 4 (IPv4), que é a versão ainda mais popular no momento da escrita deste capítulo, e a versão 6 (IPv6), que é a versão que começa a ser também usada e que tenderá a tornar-se dominante. Sempre que for relevante serão postos em evidência os aspectos em que a nova versão difere da antiga.

## 17.1 A Internet e o endereçamento IP

Os endereços das componentes de um sistema, e em particular os endereços das interfaces ligadas a uma rede, estão envolvidos em vários problemas, nomeadamente: como gerá-los e afectá-los, e como chegar às entidades a que estão associados.

A maneira mais simples de gerar e afectar endereços consiste em gerá-los de forma descentralizada e aleatória. Com um número suficiente de bits (*e.g.*, 128, 192, 256, ...) a probabilidade de colisão seria da mesma ordem de grandeza que a de descobrir por força bruta uma chave criptográfica com o mesmo número de bits. No entanto, este método também tem vários inconvenientes: a necessidade de usar um grande número de bits (seriam 128 suficientes?) e a necessidade de usar geradores pseudo-aleatórios com diferentes sementes e de qualidade suficiente para evitar colisões. No entanto, o grande inconveniente desta solução tem a ver com o encaminhamento: como localizar um destes endereços na rede global?

A Internet tem hoje em dia vários milhares de milhões de utilizadores. No primeiro trimestre de 2016 só a Akamai ([Akamai Technologies, 2016]) conseguiu recensear cerca de um milhar de milhões de endereços IP diferentes em pacotes que chegaram à sua infra-estrutura global. Como realizar então encaminhamento numa rede com esta dimensão se usássemos endereços aleatórios? Necessariamente, algumas entradas nas tabelas de encaminhamento ou comutação dos comutadores das redes de trânsito teriam de ter muitos milhões de entradas e a localização passaria por métodos sofisticados, envolvendo directórios de grande dimensão<sup>2</sup>.

Uma solução mais simples passa por introduzir alguma forma de agregação ou hierarquia no endereçamento. Assim, os endereços IP estão agregados em conjuntos disjuntos, discriminados por um certo número de bits de prefixo, que se chamam *prefixos IP*. Os endereços de cada um desses conjuntos estão afectados e localizados numa das redes que forma a Internet e portanto há uma correspondência entre prefixos e redes que facilita a localização. Quando um sistema de endereçamento está dependente da localização, diz-se que o mesmo tem **carácter topológico**. Os endereços e os prefixos IP têm uma organização hierárquica e topológica.

Esta problemática da agregação de endereços já foi de alguma forma abordada quando discutimos o endereçamento IEEE 802 (originalmente concebido para as redes Ethernet). Como estas redes se baseiam em difusão, e o encaminhamento repousa em difusão e inundação, elas podem usar quaisquer endereços, desde que distintos uns dos

---

<sup>2</sup> Este problema tem de ser hoje em dia solucionado nas rede celulares de milhões de terminais móveis e existe também uma linha de investigação semelhante em redes de conteúdos, que são identificados com identificadores com um grande número de bits.

outros, pois a problemática do encaminhamento está *a priori* resolvida. A geração e gestão dos endereços de nível MAC ficou a cargo dos fabricantes das interfaces. Por isso estes endereços têm prefixos por fabricante, o que facilita a sua geração e gestão, mas esses prefixos não têm carácter topológico pois são independentes da localização.

Os endereços IP também têm prefixos, mas estes são topológicos e relacionados com a localização numa dada rede e por isso não podem ser afectados pelos fabricantes. Na verdade, esta dependência da localização implica que os mesmos sejam afectados pelos gestores das diferentes redes constituintes da Internet.

Aqui chegados, poderíamos colocar a seguinte questão: mas se as interfaces já têm de ter endereços IP, porque razão são necessários os endereços de nível MAC? A resposta passa por constatar que os endereços MAC são bastante mais simples de gerir e afectar, podem ser usados para comunicar antes de as interfaces terem endereços IP definidos e, adicionalmente, podem haver redes que não utilizam sequer endereços IP. De qualquer forma, como veremos a seguir, as redes IPv6 podem usar os endereços de nível MAC para facilitar a gestão e afectação de endereços IPv6.

Devido à escala da Internet, onde existem biliões de interfaces globalmente endereçáveis, e a necessidade de tornar o encaminhamento mais simples, os endereços IP são de carácter topológico e hierárquico, são afectados por prefixos relacionados com a localização das diversas redes constituintes da Internet e são depois geridos e afectados pelos gestores das mesmas.

### Endereços IPv4

Os endereços IPv4 têm 32 bits. Esses 32 bits escritos como um número decimal inteiro sem sinal são difíceis de memorizar. Como a representação binária também é difícil de manipular pelos humanos, os endereços IPv4 são representados (para utilização por humanos) usando a designada ***quad notation***, que consiste em representar em notação decimal sem sinal cada um dos 4 (quad) bytes que formam o endereço, separados por um ponto, ver a Figura 17.1.

3 246 947 883 — Notação decimal								
11000001100010001111000101011 — Notação binária								
<b>193.136.126.43 — Quad Notation</b>								
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">11000001</td> <td style="padding: 2px;">10001000</td> <td style="padding: 2px;">01111110</td> <td style="padding: 2px;">00101011</td> </tr> </table>	11000001	10001000	01111110	00101011				
11000001	10001000	01111110	00101011					
<table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%; text-align: center;">↓</td> </tr> <tr> <td style="width: 25%; text-align: center;">193</td> <td style="width: 25%; text-align: center;">136</td> <td style="width: 25%; text-align: center;">126</td> <td style="width: 25%; text-align: center;">43</td> </tr> </table>	↓	↓	↓	↓	193	136	126	43
↓	↓	↓	↓					
193	136	126	43					

Figura 17.1: Representação do mesmo endereço IPv4 em decimal, binário e usando a *quad notation*.

Dado o carácter topológico do endereçamento IP, os endereços estão divididos em conjuntos, identificados por um conjunto de bits mais significativos, chamado o prefixo do endereço. O conjunto de endereços identificado por um prefixo tem uma cardinalidade que é uma potência de dois<sup>3</sup>. Daqui para a frente chamaremos a estes conjuntos de endereços **prefixos IP ou simplesmente prefixos**. Cada prefixo é

<sup>3</sup> Como foram concebidos para serem manipulados por humanos, os números de telefone utilizam prefixos telefónicos cuja dimensão é uma potência de dez, i.e., são codificados num dado número de algarismos decimais.

identificado pelo conjunto de bits mais significativo que o caracteriza. Repare-se, no entanto, que a dimensão em bits de diferentes prefixos não tem de ser a mesma.

Para denotar um prefixo, utiliza-se também a *quad notation*, complementada com a dimensão em bits do prefixo e separada do mesmo com uma barra. Com esta representação é relativamente mais fácil manipular os endereços e é também mais fácil reconhecer se vários endereços IP pertencem ou não ao mesmo prefixo.

Devido à escala da Internet, onde existem biliões de interfaces globalmente endereçáveis, e a necessidade de tornar o encaminhamento mais simples, os endereços IP são de carácter topológico e hierárquico, são afectados por **prefixos** relacionados com a localização das diversas redes constituintes da Internet e são depois geridos e afectados pelos gestores das mesmas.

Um prefixo IP é representado por um endereço escrito em *quad notation*, com zeros nos bits à direita do prefixo, seguido de uma barra e da dimensão do prefixo.

Admitindo que o prefixo de rede a que o endereço 193.136.124.43 pertence tem 24 bits de comprimento, esse prefixo é representado por 193.136.124.0/24. A notação usada indica que este prefixo tem 24 bits de comprimento e por isso é distinto do prefixo 193.136.124.0/28 que tem 28 bits de comprimento. O exemplo também põe em evidência que um prefixo pode estar contido noutro. Por outro lado, é fácil de reconhecer que os endereços 193.136.124.10 e 193.136.124.129 estão ambos contidos dentro do prefixo 193.136.124.0/24 pois os bits em que os endereços diferem estão todos no quarto byte, enquanto que o prefixo é caracterizado pelo valor dos primeiros três bytes. No entanto, enquanto que o endereço 193.136.124.10 está contido no prefixo 193.136.124.0/28 (que corresponde intervalo de endereços 193.136.126.0 – 193.136.126.15), o endereço 193.136.124.129 não está. O número a seguir à barra que se segue ao endereço em *quad notation* é o número de bits mais significativos que caracterizam o prefixo. Quanto maior for esse número, menor é a cardinalidade do conjunto de endereços que o prefixo denota.

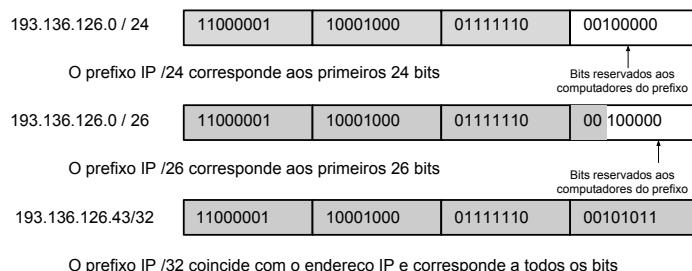


Figura 17.2: Prefixos (a sombreado) e endereços

A Figura 17.2 ilustra o papel dos prefixos dos endereços. Um prefixo IP caracteriza um conjunto de endereços contíguos, alinhado com intervalos da dimensão de uma potência de 2, e associado a uma (sub-)rede.

Cada prefixo IP activo está associado a uma (sub-)rede. Os bits do endereço à direita dos bits do prefixo, *i.e.*, os bits menos significativos, correspondem ao endereço IP da interface dentro do intervalo do prefixo, *i.e.*, dentro da (sub-)rede. Um endereço activo dentro do conjunto do prefixo está necessariamente associado a uma interface que está acessível dentro da (sub-)rede associada ao prefixo.

Um prefixo IP denota um conjunto de endereços IP distintos mas contíguos, e que têm em comum o conjunto de bits mais significativos. O prefixo identifica a (sub-)rede associada do ponto de vista do encaminhamento. Por outro lado, os prefixos têm entre si uma relação hierárquica, ou bem que um está completamente contido no outro, e nesse caso a sua intercepção corresponde ao prefixo mais longo, *i.e.*, o de menor cardinalidade, ou então são disjuntos.

Dado os prefixos serem formados por conjuntos associados a conjuntos mais significativos de bits, das duas uma: ou bem que as duas (sub-)redes são completamente disjuntas, ou bem que uma é um sub-conjunto da outra. A Figura 17.3 mostra vários prefixos IP e ilustra a relação entre os mesmos. Na mesma consideram-se dois prefixos disjuntos ( $10.0.0.0/8$  e  $193.0.0.0/8$ ), assim como vários sub-prefixos do segundo, contidos uns nos outros.

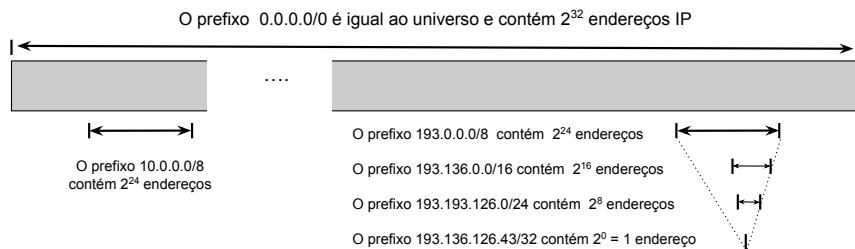


Figura 17.3: Prefixos IP e ilustração da relação entre diferentes prefixos (a dimensão dos prefixos na figura não está à escala).

Existe uma relação estreita entre os prefixos IP e os endereços. Ao nível global da Internet os endereços IP estão associados a interfaces e são opacos aos prefixos a que pertencem pois analisando isoladamente um endereço IP (*e.g.*, 193.136.124.10), não é possível saber a que prefixos activos pertence.

No entanto, o encaminhamento na Internet é realizado em direção a prefixos IP pois cada uma das (sub-)redes da Internet tem um ou mais prefixos associados. Por outro lado, numa primeira aproximação, os sub-prefixos de um dado prefixo estão associados a redes subordinadas à que está associada ao prefixo envolvente. Mesmo tratando-se de uma visão simplificada do processo, o facto é que se torna claro que os prefixos IP são essenciais para tornar o encaminhamento global na Internet mais simples e escalável. As entradas das FIBs (tabelas de encaminhamento ou comutação) dos comutadores IP são indexadas por prefixos IP.

Os diferentes prefixos IP existentes têm entre si relações hierárquicas. Todos os prefixos são sub-prefixos do universo, denotado  $0.0.0.0/0$ , o prefixo de comprimento nulo. Diferentes sub-prefixos são depois afectados às redes que formam a Internet e sub-prefixos desses são depois afectados a redes subordinadas às mesmas.

Esta hierarquia não é estrita mas desempenha um papel fundamental no encaminhamento pois o encaminhamento no mundo IP consiste em encaminhar os pacotes para os prefixos a que pertencem. É esta característica que está por detrás da escalabilidade do encaminhamento na Internet pois as tabelas de encaminhamento contém prefixos IP e não endereços IP isolados.

### Notas históricas sobre os prefixos IP

Quando o protocolo IP foi introduzido, acreditou-se que os 32 bits seriam mais do que suficientes para todos os endereços que viessem a ser necessários, e também se arranjou um esquema de, dado um endereço IP, poder identificar imediatamente a que prefixo ele pertencia. Isso dispensava o uso de comprimento do prefixo pois este estava implícito.

Assim, se o primeiro bit do endereço valesse '0', i.e., se o valor do primeiro byte estivesse entre 0 e 127 (50% do espaço de endereçamento), o prefixo era o /8 e considerava-se que a (sub-)rede a que o mesmo estava associado era uma rede classe A, que era identificada pelo valor do primeiro byte do endereço. Se os dois primeiros bits do endereço valessem '10', i.e., se o valor do primeiro byte estivesse entre 128 e 191(25% do espaço de endereçamento), o prefixo era o /16 e considerava-se que a (sub-)rede a que o mesmo estava associado era uma rede classe B, que era identificada pelo valor dos primeiros dois bytes do endereço. Finalmente, Se os três primeiros bits do endereço valessem '110', i.e., se o valor do primeiro byte estivesse entre 192 e 223, o prefixo era o /24 e considerava-se que a (sub-)rede a que o mesmo estava associado era uma rede classe C, que era identificada pelo valor dos primeiros três bytes do endereço. Este esquema revelou-se demasiado rígido e provocava um grande desperdício do espaço de endereçamento (e.g., em 50% do espaço de endereçamento apenas podiam existir 127 (sub-)redes) e por isso foi abandonado.

O novo esquema, que é o introduzido acima e que continua em utilização, foi batizado por **endereçamento sem classes ou Classless Interdomain Routing (CIDR)**. Em muitos livros de texto continua-se a introduzir a forma antiga (por classes) e só depois se introduz o conceito de CIDR. No entanto, mais de 20 anos depois da sua introdução e do mesmo se ter tornado íntimo e imprescindível ao esquema de endereçamento IP, parece-nos inútil continuar a introduzir o conceito de prefixo IP com base na sua origem histórica, já caída em desuso e para mais inaplicável em IPv6. Esta nota histórica justifica-se apenas para o caso de o leitor tropeçar no termo CIDR e na noção de classe de um endereço IP.

Um outro conceito que caiu em desuso foi o de **máscara de rede IP (IP network mask)** que é uma outra forma de indicar a que prefixo um endereço IP pertence. Uma máscara IPv4 é uma sequência de 32 bits que começa com uma sequência de bits a 1, seguida de uma sequência de bits a 0. A máscara correspondente aos endereços classe A, i.e., prefixos da forma /8, é 255.0.0.0 (primeiro byte com o valor '1111111'), a máscara correspondente aos endereços classe B, i.e., prefixos da forma /16, é 255.255.0.0, a máscara correspondente aos endereços classe C, i.e., prefixos da forma /24, é 255.255.255.0, a máscara correspondente ao prefixo /26 é 255.255.255.192, etc.

Para todo os efeitos, a máscara é uma forma de indicar em 4 bytes o comprimento do prefixo, ao invés de em um único byte, mas com a vantagem de permitir calcular imediatamente o prefixo através de uma operação de *and* bit a bit entre a máscara e o valor do endereço. Modernamente a tendência é para deixar ao software o encargo de calcular a máscara e deixar os humanos usarem a notação '/tamanho do prefixo' que é mais confortável. No entanto, se o leitor cair sobre uma máscara IPv4 já sabe do que se trata. A mesma notação em IPv6 também é aplicável mas é raramente utilizada.

Antes de penetrarmos de forma mais detalhada na relação entre prefixos IP e encaminhamento, vamos ver a seguir como são afectados os prefixos IP às (sub-)redes.

### Como são afectados os prefixos IP

A entidade responsável por gerir o espaço de endereçamento (Versão 4 e Versão 6) da Internet é a **Internet Corporation for Assigned Names and Numbers (ICANN)**, uma organização sem fins lucrativos. A ICANN gera o espaço de endereçamento IP desempenhando o papel de **Internet Assigned Numbers Authority**

**(IANA)**. A ICANN também gera o domínio **root**, e delega a gestão dos seus sub-domínios directos (os TLDs ou *Top Level Domains*). Dado o carácter de monopólio supra-nacional que a ICANN exerce sobre a Internet, esta empresa é gerida segundo um princípio *multistakeholder*, i.e., uma gestão por consenso envolvendo todas as partes a quem os processos dizem respeito.

A gestão dos endereços também é hierarquizada. A IANA afecta prefixos IP globais a diversos usos. A maioria desses prefixos, são prefixos /8 de endereços IP públicos que são afectados às **Regional Internet Registries (RIRs)**. Estas organizações são 5 a nível mundial e repartem entre si a responsabilidade de gestão sobre as diversas regiões do mundo. Por exemplo, a associação **Réseaux IP Européens Network (RIPE)** é responsável pela Europa, Rússia, Médio Oriente e Ásia Central. As RIRs são associações sem fins lucrativos formadas pelos operadores de redes da Internet (ISPs)<sup>4</sup> da sua região e afectam-lhes sub-prefixos dos prefixos que recebem da IANA.

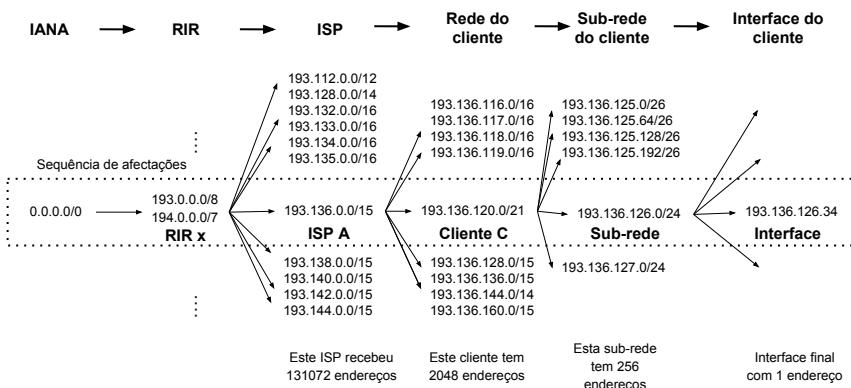


Figura 17.4: Sequência de feactações de prefixos IP: à RIR x, ao ISP A, ao cliente C etc. que permitem a afectação de um endereço a uma interface na rede do cliente C.

A dimensão do prefixo afectado a cada ISP depende da dimensão e importância da sua rede, e indirectamente do valor da sua quota anual para a RIR da sua região. Conforme vai evoluindo e necessita de mais espaço de endereçamento, um ISP pode ir recebendo mais prefixos IP, mas eventualmente disjuntos.

Os ISPs afectam espaço de endereçamento aos seus clientes, i.e., sub-prefixos dos prefixos IP que lhes foram afectados pela sua RIR. A Figura 17.4 ilustra este processo: um prefixo /8 foi afectado a uma RIR, esta afecta um prefixo /15 a um operador (ISP) e este afecta sub-prefixos aos seus clientes, que por sua vez afectam prefixos às suas sub-redes e endereços aos computadores ligados às mesmas.

### Prefixos IP e encaminhamento

O encaminhamento na Internet é realizado movendo os pacotes em direção às redes associadas aos prefixos IP que a que pertencem os seus endereços de destino. No entanto, nem todos os prefixos têm de ser visíveis no sistema de encaminhamento global da Internet. Os prefixos afectados pela IANA aos RIRs são meramente administrativos pois não são visíveis pelo sistema de encaminhamento visto que não há “encaminhamento para África”, nem “encaminhamento para a América do Norte”, i.e., o encaminhamento IP não é baseado na localização geográfica. O primeiro nível

<sup>4</sup> Internet Service Providers.

visível é o do encaminhamento para os prefixos dos operadores, pois estes têm de estar necessariamente visíveis para efeitos de encaminhamento. A seguir coloca-se a questão: e os prefixos dos clientes dos ISPs são visíveis no sistema de encaminhamento da Internet? A resposta é “depende”.

As redes domésticas estão ligadas às redes de acesso dos operadores. O comutador de pacotes que liga cada uma delas à Internet tem um endereço ou um prefixo que apenas é conhecido dentro da rede do operador. Isto é, na Internet estes destinos são alcançados entregando os pacotes que lhes são dirigidos aos comutadores da fronteira da rede do operador que os serve. Só dentro da rede do operador os prefixos mais longos das sub-redes dos clientes do operador, assim como os endereços finais dos clientes domésticos, são conhecidos.

Veremos na Secção 17.3 que muitas dessas redes domésticas têm mais do que um computador ligado, cada um com um endereço diferente, mas todos esses computadores usam um endereço IP partilhado, pelo que em IPv4, o prefixo que está associado a uma rede doméstica é geralmente de comprimento /32, i.e., contém apenas um endereço. Os terminais móveis ligados às redes dos operadores celulares também têm um único endereço IP, contido num prefixo IP do operador celular.

Muitos clientes institucionais (universidades, empresas, etc.) recebem prefixos dos seus operadores e muitas dessas redes também não estão visíveis no sistema de encaminhamento global. Para que um pacote lhes chegue, primeiro determina-se o prefixo IP a que o endereço de destino pertence, depois faz-se chegar o pacote à rede do operador à qual o prefixo está associado. Posteriormente, dentro da rede do ISP é realizado o encaminhamento para o prefixo da rede institucional do cliente.

O que acontece se um cliente do ISP A pretender mudar de operador? Se tudo continuar na mesma do ponto de vista dos endereços e dos prefixos anunciados na Internet global, a nova ligação é inútil. De facto ela não será usada, visto que na Internet global a mesma não é conhecida porque o prefixo do cliente continua contido (e escondido) no prefixo do antigo operador. Para ser usada, o prefixo específico do cliente tem de ser anunciado pelo novo operador para que o encaminhamento para o cliente possa ser realizado via o novo operador (e.g., ISP B), ao invés de via o antigo operador (ISP A).

Fazendo referência à Figura 17.4, o anúncio 193.136.0.0/15 acessível via A é anunciado pelo ISP A, o que é suficiente para enviar pacotes para o cliente C com o prefixo subordinado 193.136.120.0/21. Se o cliente C quiser mudar para o ISP B, das duas uma, ou muda de endereços, o que pode ser complexo e dispendioso, ou deixa de ser acessível na Internet pois os pacotes serão enviados para uma rede (a do ISP A) sem ligação à sua. A outra alternativa, sem mudar de endereços, é o ISP B passar a fazer um anúncio 193.136.120.0/21 acessível via B. Só que agora os pacotes dirigidos ao cliente C (por exemplo 193.136.126.43) pertencem a dois prefixos nas tabelas de encaminhamento: 193.136.0.0/15 via A e 193.136.120.0/21 via B. Se for escolhido o prefixo mais curto (/15) o pacote irá parar a um beco sem saída pois o ISP A deixou de ter ligação a C.

Para que o encaminhamento funcione correctamente é aplicada a regra **o encaminhamento pelo prefixo mais longo é o melhor (the longest prefix is the best)**. Com esta regra, o anúncio via B tem precedência e os pacotes dirigidos a 193.136.126.43 vão sempre via o ISP B, que é o que se pretendia.

Supondo agora que o desejo do cliente C era assegurar conectividade à Internet via os dois operadores (ISP A e ISP B), então teriam de ser feitos dois anúncios para além dos anúncios dos prefixos dos operadores A e B: prefixo 193.136.120.0/21 acessível via A e acessível via B. Se aparecesse apenas o anúncio do prefixo do cliente C via o ISP B, então a ligação ao ISP A seria inútil porque nunca seria usada<sup>5</sup>. Quando um cliente pretende estar ligado a mais do que um ISP, esse cliente diz-se *multi-homed*.

---

<sup>5</sup> A outra alternativa seria não fazer nenhum anúncio específico do prefixo do cliente C. No entanto, como veremos no Capítulo 18, essa solução não seria muito adequada.

Devido à regra *the longest prefix is the best*, um cliente pode mudar de operador sem mudar de prefixo IP, mas os operadores, por razões comerciais e também técnicas, começaram a restringir a hipótese de os clientes usarem prefixos IP dos seus blocos sem lhes estarem ligados, ou seja sem estarem ligados ao “dono” dos endereços. Quando um prefixo de um cliente está contido no prefixo de um operador, esse prefixo diz-se pertencer ao **espaço de endereçamento dependente de provedores** (*provider dependent address space*).

Para os clientes com dimensão razoável que prevêem que mais tarde ou mais cedo desejem mudar de ISP e não queiram mudar de endereços nessa altura, ou desejem *a priori* estar ligados a vários ISPs, as RIRs dispõem também de blocos de endereços que não estão afectados a nenhum ISP e funcionam como um ISP virtual para efeitos de administração de endereços, pois afectam directamente prefixos específicos a clientes finais de dimensão razoável. Um prefixo desse tipo diz-se que pertence ao **espaço de endereçamento independente** (*provider independent address space*) e tem de ser anunciado directamente na Internet por um ou mais ISPs pois o prefixo global em que está contido não é anunciado por nenhum operador.

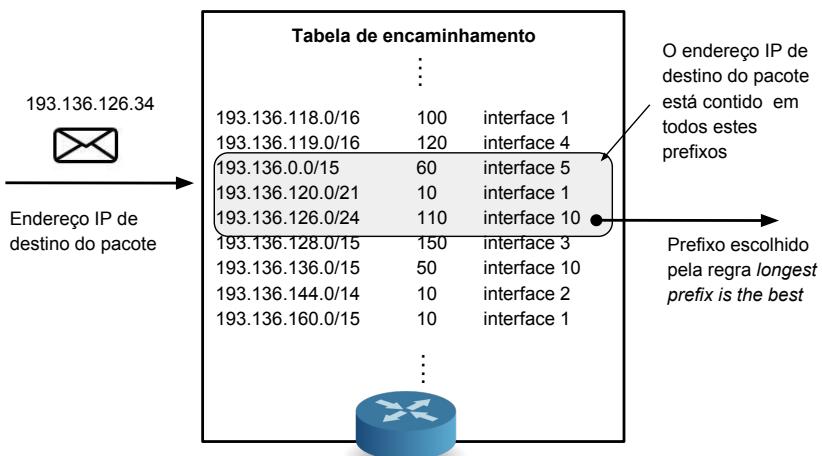


Figura 17.5: Tabela de encaminhamento IP e a regra *longest prefix is the best*

Figura 17.5 mostra um trecho de uma tabela de encaminhamento global (*i.e.*, da FIB de um comutador que conhece os prefixes IP encaminhados na Internet global<sup>6</sup>) e como a regra *longest prefix is the best* é usada para seleccionar a interface pela qual o pacote deve seguir. A flexibilidade permitida por esta regra, e o conjunto dos mecanismos de afectação de endereços IP que foram introduzidos, permite o máximo de concorrência do ponto de vista comercial entre ISPs e também todas as formas de interligação que se revelarem desejáveis por razões de desempenho ou segurança.

No entanto, os mesmos têm dois efeitos nefastos. O primeiro tem a ver com o facto de que a consulta das FIBs deixou de ser possível usando simplesmente uma estrutura de dados do tipo tabela de dispersão (*hash tables*) e passou a ser necessária um estrutura mais sofisticada, caracterizada como uma árvore de prefixes. A raiz dessa árvore está associada ao prefixo 0.0.0.0/0. Os filhos da raiz são os sub-prefixos de topo dos operadores e os correspondentes aos prefixes pertencentes aos *provider independent address spaces*. A seguir, qualquer outro prefixo conhecido pelo comutador, tem de ser

<sup>6</sup> A Internet global é também designada por *Default-Free Zone* (DFZ) por razões que serão mais claras a seguir.

filho do maior prefixo que o contém. A gestão desta estrutura é muito mais pesada que a simples gestão de tabelas de dispersão, o que encarece a construção e a operação dos comutadores de alta gama.

O segundo efeito nefasto tem a ver com a explosão do número de prefixes IP específicos que têm de ser conhecidos na Internet global. No limite, se cada rede doméstica tivesse endereços no espaço *provider independent address space*, as tabelas de encaminhamento globais teriam de ter biliões de entradas com prefixes de comprimento /32. É por esta razão que existe pressão para só anunciar globalmente prefixes curtos (*i.e.*, com um pequeno número de bits), pois isso torna as tabelas de encaminhamento globais menos detalhadas e portanto de menor dimensão.

No início do ano de 2016 a tabela de prefixes IP visíveis globalmente, *i.e.*, a tabela de encaminhamento dos comutadores de pacotes do “coração” da Internet, continha mais de 600.000 entradas (prefixos distintos que exigem encaminhamento específico). Sempre que os comutadores se tornaram mais lentos, ou mesmo incapazes de lidarem com tabelas da dimensão e complexidade necessária, porque a tecnologia do momento não o permitia, muitos operadores Tier 1 e 2, ver o Capítulo 4, os que mais sofreram com o problema, começam a não aceitar os anúncios de prefixes muito longos (*i.e.*, com muitos bits), o que limita a flexibilidade do encaminhamento para os clientes finais e tem implicações comerciais de exercício de poder ligado à dimensão da rede.

A Figura 17.6 mostra um gráfico da evolução do número de prefixes visíveis na Internet global ao longo dos anos.

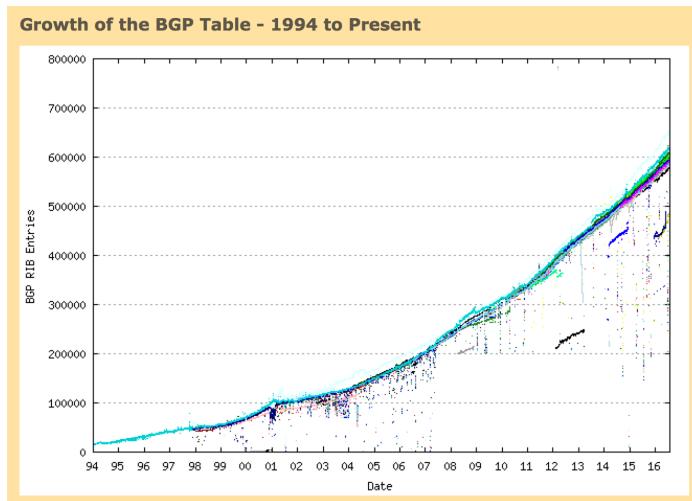


Figura 17.6: Evolução do número de prefixes IP visíveis nas tabelas de encaminhamento globais da Internet segundo o site <http://bgp.potaroo.net>. Consulta realizada em Julho de 2016.

O encaminhamento na Internet global, a chamada *Default-Free Zone*, é realizado com base nos prefixos IP anunciados pelas diferentes redes interligadas. Para evitar que uma hierarquização total dos prefixos redundasse num sistema demasiado rígido, e potencialmente com as características de um monopólio dos operadores Tier1, são anunciados prefixos correspondentes a todas as redes, mesmo de clientes finais, desde que estas tenham a necessidade de controlar de forma mais fina o encaminhamento dos pacotes que lhes são dirigidos.

Os prefixos afectados pelos operadores aos seus clientes fazem parte do espaço *provider dependent address space*. Esses clientes, se não forem *multi-homed*, estão num espaço de endereçamento que também se pode chamar *provider-based aggregation address space* porque poupa entradas nas tabelas de encaminhamento. Ao contrário, os prefixos das redes clientes mas independentes do fornecedor, têm prefixos no *provider independent address space* ou *without aggregation address space* e expandem necessariamente as tabelas de encaminhamento globais.

A regra *longest prefix matching is the best* na escolha das entradas das tabelas de encaminhamento dos comutadores funciona como garante desta flexibilidade. O controlo fino do encaminhamento para uma rede, e as necessidades de *multi-homing* levam à subida incessante do número de prefixos IP visíveis no “coração” da Internet.

### Prefixos IPv4 especiais

As regras de afectação de endereços IPv4 comportam diversos prefixos IP especiais que não podem figurar nas tabelas de encaminhamento globais e que têm utilizações especiais, nomeadamente os da Tabela 17.1, onde figuram os principais blocos de endereços desse tipo. Os blocos de endereços IPv4 privados são discutidos no fim da Secção 17.3. O bloco de endereços IP multicasting são usados para grupos de IP multicasting, ver a Secção 5.2. O endereço 127.0.0.1 é o endereço de *loopback*, *i.e.*, o endereço de uma interface virtual existente em todos os dispositivos que comunicam em IP, e que denota localmente o próprio dispositivo. Este endereço também é conhecido pelo nome **localhost**, ver a Secção 5.1. Os *link local IP addresses* são endereços IP que podem ser afectados localmente pelo software IP mas que só são válidos para comunicar com computadores ou outros dispositivos ligados ao mesmo canal. ver a Secção 17.3

### A quem pertencem e onde estão os endereços IP

Dado um endereço IP é possível consultar as bases de dados das RIRs e obter informação sobre a sua afectação. Este serviço está disponível via o comando **whois**. O exemplo abaixo mostra parte do resultado da execução da interrogação **whois -h whois.ripe.net 193.136.126.43** à base de dados do RIPE. O resultado mostra que o endereço 193.136.126.43 pertence ao prefixo IP 193.136.120.0/21 que foi afectado à Faculdade de Ciencias e Tecnologia da Universidade Nova de Lisboa (FCT/UNL) em 30 de Novembro de 1993 pelo serviço IP da UNL e que se trata de um prefixo do *PA - Provider Aggregated address space*. Com efeito este prefixo faz parte de um prefixo maior afectado pela Rede das Universidades Portuguesas à UNL.

Tabela 17.1: Principais prefixos IPv4 reservados, ver o RFC 6890

Nome	Intervalo ou endereço	# endereços disponíveis	Prefixo IP	RFC
24-bit block private IP addresses	10.0.0.0 a 10.255.255.255	16 777 216	10.0.0.0/8	RFC 1918
20-bit block private IP addresses	172.16.0.0 a 172.31.255.255	1 048 576	172.16.0.0/12	RFC 1918
16-bit block private IP addresses	192.168.0.0 a 192.168.255.255	65 536	192.168.0.0/16	RFC 1918
multicast IP addresses	224.0.0.0 a 239.255.255.255	268 435 456	224.0.0.0/4	RFC 1122
loopback IP addresses	127.0.0.1 a 127.255.255.255	16 777 216	127.0.0.0/8	RFC 1122
loopback IP address	127.0.0.1	1	127.0.0.1/32	RFC 1122
link local IP address	169.254.0.0 a 169.254.255.255	65 536	169.254.0.0/16	RFC 3927

```
$ whois -h whois.ripe.net 193.136.126.43
% This is the RIPE Database query service.

...
193.136.120.0 - 193.136.127.255
PT-FCT-UNL-1
Faculdade de Ciencias e Tecnologia da Universidade Nova ... PT
RCUN1-RIPE
ASSIGNED PA
ORG-UNDL3-RIPE
SERVIP-UNL
created 19931130
```

Uma outra faceta interessante dos endereços IP tem a ver com o facto de que hoje em dia é possível obter informação sobre a localização geográfica dos endereços IP, como já foi referido na Secção 13.2.

### Endereços IPv6

Os endereços da versão 6 do protocolo IP, ou IPv6, têm 128 bits, que acomodam  $2^{128} = 340\,282\,366\,920\,938\,463\,463\,374\,607\,431\,768\,211\,456$  endereços distintos.

Para a representação legível destes endereços foi introduzida uma notação designada **hexadecimal colon notation**: cada endereço é escrito usando 8 grupos de 4 dígitos hexadecimais separados pelo símbolo dois pontos. Cada grupo de dígitos tem 4 símbolos e representa 16 bits. Um exemplo de utilização desta notação é o endereço: **2001:0:0:0:0:45bf:ff23:8b4f**.

A notação admite igualmente que qualquer sequência de grupos só com zeros possa ser omitida, sendo substituída por dois símbolos dois pontos seguidos. Em cada endereço só pode haver uma destas substituições, senão resultaria numa ambiguidade (excepto no caso particular do endereço "0::0" que pode ser abreviado para "::"). Aplicando esta regra ao exemplo anterior resulta numa representação mais compacta do mesmo endereço: **2001::45bf:ff23:8b4f**.

Os endereços IPv6 são afectados com a mesma lógica que os endereços IPv4, nomeadamente usando prefixos IPv6. Existem igualmente diversos prefixos IPv6 reservados. A Tabela 17.2 apresenta os principais.

Tabela 17.2: Principais prefixos IPv6 reservados, ver o RFC 6890

Nome	Intervalo ou endereço	# endereços disponíveis	Prefixo IP	RFC
reserved for special purposes	0:0:0:0:0:0:0 a 00ff:ffff:ffff:...ffff:ffff	$2^{120}$	::/8	RFC 4291
unspecified IP address	0:0:0:0:0:0:0	1	::/128	RFC 4291
loop-back IP address	0:0:0:0:0:0:1	1	::1/128	RFC 4291
multicast IP address	ff00:0::0 a ffff:ffff:...:ffff:ffff	$2^{120}$	ff00::/8	
link-local IP addresses	fe80:: a febf:ffff:...:ffff:ffff	$2^{118}$	fe80::/10	RFC 4291
site-local IP addresses	fec0:: a fecf:ffff:...:ffff:ffff	$2^{118}$	fec0::/10	RFC 4193

Os endereços que começam com 8 bits a zero (o prefixo ::/8) são reservados para funções especiais. O endereço "::0" é um endereço que significa ausência de endereço e que só pode ser usado por uma interface antes de ter um endereço próprio, para enviar um pacote em difusão à procura de quem lhe afecte um endereço. O endereço "::1" é o endereço *loopback* que corresponde ao nome `localhost` em IPv6 e é equivalente ao endereço 127.0.0.1 em IPv4. Nesta gama de endereços estão também previstas formas especiais de representar em IPv6 endereços IPv4. Esses endereços começam com 96 bits a zero seguidos dos 32 bits do endereço IPv4 ou com 80 bits a zero, seguidos de 16 bits a um, seguidos dos 32 bits do endereço IPv4.

Os endereços que começam com 8 bits a um são endereços IPv6 multicasting. Os endereços *link-local IP addresses* são endereços IP que podem ser afectados localmente pelo software IP mas que só são válidos para comunicar em IP com computadores ou outros dispositivos ligados ao mesmo canal. Os endereços *site-local IP addresses* têm em IPv6 o mesmo papel que os endereços privados em IPv4.

Actualmente a ICANN está a afectar blocos de endereços IPv6 às RIRs no prefixo 2000::/3 (com 001 nos primeiros 3 bits do endereço) para serem distribuídos como endereços unicast. As afectações de prefixes IPv6 estão também registadas pelos RIR. Abaixo mostra-se parte do resultado de executar o comando `whois -h whois.ripe.net` sobre o prefixo 2001:690::/32. Através do resultado constata-se que se trata de um prefixo afectado pela Fundação para a Ciéncia e a Tecnologia à rede Portuguese National Research & Education Network como sub-prefixo do seu prefixo 2001:690::/29.

```
$ whois -h whois.ripe.net 2001:690::/32
% Information related to '2001:690::/29'

inet6num: 2001:690::/29
netname: PT-RCCN-20000623
country: PT
org: ORG-FpaC1-RIPE
admin-c: JNF1-RIPE
status: ALLOCATED-BY-RIR
.....
created: 2013-11-22T07:46:28Z
source: RIPE

organisation: ORG-FpaC1-RIPE
```

```

org-name: Fundacao para a Ciencia e a Tecnologia, I.P.
...
address: PORTUGAL

% Information related to '2001:690::/32AS1930'

route6: 2001:690::/32
descr: The Portuguese National Research & Education Network
origin: AS1930
...
created: 2005-04-06T14:05:57Z
source: RIPE

```

Depois desta discussão sobre o endereçamento IP e as sua relações com o encaminhamento e a estrutura interna da Internet, vamos de seguida estudar o protocolo IP propriamente dito.

## 17.2 IP versão 4 e IP versão 6

O protocolo IP especifica o formato dos pacotes trocados na rede, compreendendo um cabeçalho (os primeiros bytes transferidos) com informação para o funcionamento do protocolo, seguido do *payload*, ou seja, os bytes com os dados que se pretende transferir, ver a Figura 17.7. O protocolo especifica igualmente a forma como os campos do pacote devem ser interpretados e tratados pelos comutadores de pacotes IP durante o encaminhamento do mesmo para o destino. Tradicionalmente, os comutadores de pacotes capazes de processarem pacotes IP e realizarem todas as operações envolvidas nesse processamento chamam-se comutadores IP, e em língua inglesa *IP routers*. Devido ao facto de que os especialistas em redes IP usam o termo em inglês mesmo quando a sua língua materna não é o inglês, no resto deste capítulo utilizaremos o termo *router* IP, ou simplesmente *router*, para designar os equipamentos capazes de realizarem processamento inteligente de pacotes IP e outros protocolos que lhe estão associados.

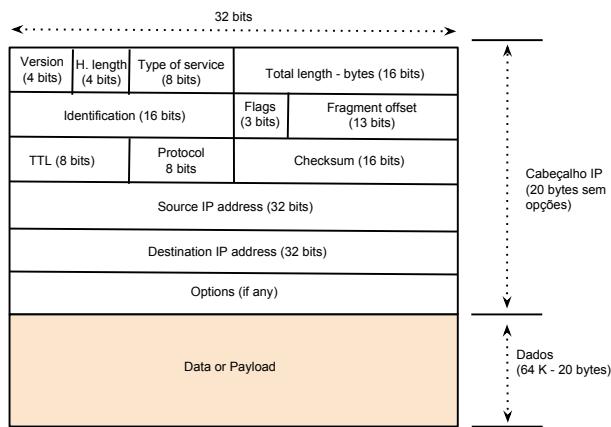


Figura 17.7: Cabeçalho de um pacote IP na versão 4

Começamos por discutir o papel dos diferentes campos do cabeçalho IPv4. O primeiro campo, com 4 bits, indica a versão do protocolo a que o pacote pertence.

Neste caso será necessariamente o valor 4 correspondente a esta versão. O campo seguinte é o campo **Header length** que especifica a dimensão do tamanho do cabeçalho em grupos de 4 bytes. Este campo é necessário pois o cabeçalho tem comprimento variável, dependente da presença de opções. Na ausência de opções, o cabeçalho ocupa  $5 \times 4 = 20$  bytes pelo que este campo tem o valor 5 nesse caso. O maior cabeçalho pode ter até  $15 \times 4 = 60$  bytes, pelo que as opções podem ocupar até 40 bytes.

O campo **Type of service** permite codificar uma indicação sobre o tipo de serviço que o emissor original do pacote pretende obter para o mesmo. Dependendo das funcionalidades implementadas pelos *routers* da rede ou redes atravessadas pelo pacote, essas indicações podem ou não ser respeitadas. No momento em que este capítulo foi escrito, na maioria das redes de trânsito este campo é ignorado. É neste campo que se podem registrar igualmente informações sobre a saturação dos *routers* atravessados pelo pacote usando a funcionalidade ECN, ver a Secção 8.4.

O campo **Total length** indica a dimensão total do pacote em bytes. Com 16 bits o campo permite que um pacote IP tenha no máximo 64 Kbytes. Como já foi referido, a grande maioria dos canais não permitem transmitir num único *frame* um pacote desta dimensão. Nesse caso, o protocolo IPv4 prevê que o *router* que se confronte com esta situação fragmente o pacote. Os campos **Identification**, **Flags** e **Fragment Offset** destinam-se a suportar esta funcionalidade que será discutida a seguir.

O campo **Time-To-Live** ou TTL é já nosso conhecido e destina-se a prevenir que um pacote envolvido num ciclo de encaminhamento permaneça eternamente dentro da rede. Pode também ser usado para outros efeitos, como por exemplo para limitar o âmbito do percurso de um pacote na rede, ver a Secção 3.4. Quando um pacote chega a um *router* este tem de decrementar o valor do TTL, caso o valor que daí resulte seja 0, o pacote deve ser destruído (ignorado).

Inicialmente pensava-se que o campo limitaria o tempo de vida dos pacotes em segundos, mas nunca foi realista implementar esta visão pois é muito difícil a um *router* calcular exactamente quanto tempo leva a processar o pacote, acrescido do tempo necessário para este chegar ao *router* seguinte. As implementações iniciais limitavam-se a decrementar de uma unidade o valor do TTL. Assim, a implementação que acabou por ser dominante é equivalente a um simples limitador do número de *routers* atravessados (*hop-count*). O valor do campo é inicializado pelo emissor original do pacote e os valores típicos de inicialização por omissão são 64 e 255, ver a Secção 3.4.

O campo **Protocol** contém um código que especifica qual o protocolo a que pertence o *payload* do pacote (e.g., ICMP, UDP, TCP, ...).

O campo **Checksum** contém um código de deteção de erros que também já é nosso conhecido, ver a Secção 2.4. Ele apenas detecta erros no cabeçalho do pacote e é calculado com o próprio valor a 0, quer pelo emissor, quer pelo receptor (ver a noção de *pseudo-header* nos RFCs 791 e 1812).

Este mecanismo foi introduzido numa época em que as técnicas de deteção de erros em alguns canais eram deficientes e destinava-se a evitar que um *router* analisasse pacotes IP com o cabeçalho corrompido. Actualmente todos os canais incluem mecanismos de deteção de erros com maior precisão que este, e os protocolos de nível superior também já incluem um campo do tipo *checksum*. O mecanismo foi abandonado em IPv6 e muitas implementações actuais do software IP deixaram de analisar este campo, mas fazem o seu cálculo antes da emissão de um pacote para respeitarem a norma e evitarem problemas com *routers* que o calculem. Com efeito, dado que o cabeçalho é sempre alterado (pelo menos o TTL é sempre decrementado), esta operação, que no essencial é inútil, tem de ser executada por todos os *routers*. Trata-se de mais um exemplo do papel benéfico, mas também limitador da evolução, que as normas introduzem.

Os dois campos obrigatórios finais são os endereços origem e destino cujos significado e utilização são óbvios.

### A problemática da fragmentação em IPv4

Quando o protocolo IP foi definido inicialmente, devido a elevadas taxas de erro e baixo débito, alguns canais tinham um MTU (*Maximum Transfer Unit*), ver a Secção 2.5, muito baixo, com apenas algumas centenas de bytes. Por esta razão o IPv4 tem um mecanismo que permite a um *router* confrontado com um pacote IP superior ao MTU do canal porque o deve transmitir, fragmentar esse pacote em vários fragmentos. No entanto, para prevenir o caso em que a operação tenha de ser repetida por diferentes *routers* para o mesmo pacote, foi adoptada a solução de que a recomposição (agrupamento dos fragmentos do pacote original) só é realizada no destino final do pacote, pois cada fragmento pode ser encaminhado de forma independente. Ou seja, adoptou-se a posição de que se houver necessidade de fragmentar um pacote, a probabilidade de este voltar a ter de ser fragmentado é elevada. Por outro lado, realizar a recomposição do pacote original apenas no destino também simplifica o trabalho e a gestão de memória dos *routers* intermediários visto que estes não têm de memorizar fragmentos à espera para posterior recomposição do pacote.

O campo **Identification** permite identificar os diferentes fragmentos como pertencentes ao mesmo pacote. O campo **Flags** permite distinguir o último fragmento dos restantes, e o campo **Fragment Offset** permite colocar cada fragmento na sua posição certa no pacote mesmo que os fragmentos cheguem fora de ordem. Para mais detalhes consultar os RFCs 791 e 815.

Adicionalmente, caso um fragmento se perca, como este só é enviado uma vez, o receptor será obrigado a destruir os restantes fragmentos, ao fim de um intervalo de tempo marcado por um alarme, pois não conseguirá recompor o pacote original. Dadas as características da Internet, este alarme tem de ser necessariamente de alguns segundos. Ver a noção de *Maximum Packet Life Time* na Secção 7.3.

Finalmente, como o campo **Identification** só admite 65535 identificações diferentes antes de obrigar à reutilização dos identificadores, com os débitos actuais, essa quantidade de pacotes pode ser emitida num período inferior à duração do alarme associado à recomposição dos fragmentos no receptor. Na eventualidade da perda de um fragmento, deixaria de estar garantida a fiabilidade do processo de recomposição de pacotes. Na Secção 17.6 é proposto um problema sobre esta possibilidade.

Actualmente quase todos os canais têm MTUs bastante mais elevados, com pelo menos 1,2 ou 1,5 Kbytes ou até bastante mais (*e.g., jumbo frames* de vários K bytes). Com efeito, mesmo os canais sem fios que usam às vezes *frames* menores, recompõem nas interfaces dos extremos do canal *frames* lógicos de maior dimensão. Assim, actualmente é preferível forçar o emissor original do pacote a emitir imediatamente pacotes IP de menor dimensão, do que envolver os *routers* e o destinatário do pacote na fragmentação.

Por todas estas razões realizar fragmentação dentro da rede é hoje em dia considerada uma má prática e o mecanismo foi abandonado em IPv6 ao nível dos *routers*. Mesmo em IPv4 recomenda-se igualmente a utilização de um protocolo designado *path MTU discovery*, ver o RFC 1981, e enviar apenas pacotes de dimensão compatível com esse MTU. É o que geralmente faz o protocolo TCP na abertura de uma sessão ao calcular o MSS (*Maximum Segment Size*), ver a Secção 7.3.

### Opções IPv4

Quando o protocolo IP foi definido, o seu cabeçalho recebeu o campo **Options** que permite introduzir mecanismos opcionais de forma flexível. A maioria das opções definidas destinam-se a permitir realizar *debug* da rede, como por exemplo a opção de registar o endereço dos *routers* que o pacote atravessou. Outras opções destinam-se a permitir a implementação de formas diferentes de encaminhamento, nomeadamente diversas formas de **encaminhamento controlado pelo emissor (source routing)**. Este tipo de opções permite ao emissor de um pacote indicar o endereço de *routers* por onde este deve passar no seu caminho para o destino.

As opções revelaram-se “um pau de dois bicos”. Por um lado, pelo menos à primeira vista, este tipo de mecanismos parecem ser úteis e desejáveis. Mas infelizmente alguns inconvenientes parecem suplantar os benefícios.

Toda a filosofia e o desempenho das redes IP deve basear-se na ideia de que o trabalho dos *routers* deve ser o mais simples que possível, relegando a complexidade para a periferia. As opções introduzem complexidade adicional nos *routers* e por essa razão a minoria de pacotes que as contém são sempre processados à parte e sem apoio de hardware específico. Com efeito, as opções não são amigas nem do desempenho, nem da simplicidade.

Este defeito poderia não ser demasiado grave visto que as opções são opcionais e só “paga o seu preço quem as usar”, mas para infelicidade do mecanismo a maioria das opções que envolvem *source routing* revelaram-se fatais do ponto de vista da segurança. Por essa razão a maioria dos *routers* dos operadores são parametrizados para ignorarem as opções IPv4.

## IP versão 6

A motivação principal para a introdução da versão 6 do protocolo IP veio da consciência de que um endereço IP com 32 bits iria tornar-se, mais tarde ou mais cedo, uma barreira intransponível à expansão da Internet. De facto, no momento em que este capítulo foi escrito todos os RIRs, com excepção do de África, já não podem afectar mais endereços IPv4 aos operadores e o *provider independent address space* também se encontra esgotado. Existem apenas pequenas reservas de endereços IPv4 que só são afectados em casos especiais.

Alterar a dimensão dos endereços passou necessariamente por uma modificação significativa do cabeçalho, por isso, aproveitou-se a experiência adquirida até então e introduziram-se outras alterações. O novo cabeçalho é maior mas mais simples que o anterior, ver a Figura 17.8.

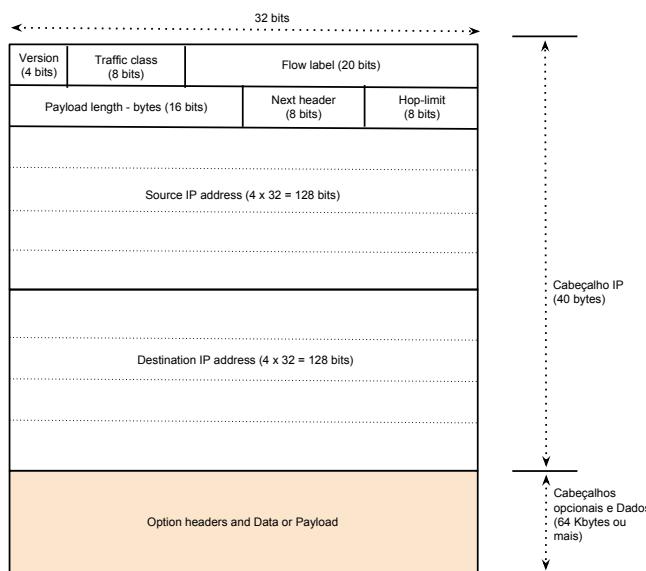


Figura 17.8: Cabeçalho de um pacote IP na versão 6

Na versão 6 do protocolo IP o cabeçalho é de dimensão fixa e não existem vários dos campos presentes em IPv4. Em particular foram suprimidos o campo *checksum*, o campo com a dimensão do cabeçalho pois este é de dimensão fixa, e não é suportada fragmentação dos pacotes pelos *routers*, nem opções. Apesar do cabeçalho mínimo IPv4 (com 20 bytes) ter sido expandido para 40 bytes em IPv6, o número de campos é inferior. A expansão da dimensão deve-se ao facto de que os endereços cresceram 4 vezes em dimensão (de 4 para 16 bytes cada um).

No que diz respeito à fragmentação a norma determina que todos os canais devem ter um MTU maior ou igual a 1280 bytes. Se necessário para garantir este limite inferior, as interfaces do canal devem implementar a fragmentação ao nível canal mas de forma transparente para os *routers*. Caso um *router* receba um pacote que necessite de fragmentação, o mesmo é suprimido e ocorre um erro.

O campo **Version** tem a mesma posição, função e dimensão que no cabeçalho IPv4. Naturalmente, neste caso o seu valor é sempre 6.

O campo **Traffic class** tem a mesma função e dimensão que o campo **Type of service** no cabeçalho IPv4. Tal como no caso do IPv4 este campo é na verdade constituído por dois sub-campos: 6 bits codificam o tipo de serviço e designam-se em IPv6 **Differentiated Services (DS)** e 2 bits dizem respeito à funcionalidade ECN. O campo **Flow label** não tem equivalente no cabeçalho IPv4. O seu papel é permitir identificar um conjunto de pacotes como pertencendo a um fluxo específico e que devem ser tratados de forma particular pelos *routers*. No entanto continua a ser considerado um mecanismo experimental que não é considerado em ambientes de produção.

O campo **Payload length** contém a dimensão do **payload** do pacote incluindo eventuais cabeçalhos opcionais. O campo **Next header** contém um código a indicar qual é cabeçalho seguinte. O mesmo pode ser um protocolo de nível superior como no IPv4 (*e.g.*, ICMP, UDP, TCP, ...) ou um cabeçalho opcional do IPv6. O campo **Hop limit** tem a mesma função e dimensão que o campo TTL no cabeçalho IPv4 e os campos **Source IP Address** e **Destination IP Address** têm os significados óbvios.

### Cabeçalhos opcionais ou de extensão IPv6

O cabeçalho IPv6 descrito acima apenas permite a implementação de mecanismos imprescindíveis ao encaminhamento, assim como funcionalidades opcionais de qualidade de serviço. Provavelmente, a grande maioria dos pacotes não precisam de quaisquer outros tratamentos. As funcionalidades opcionais do IPv6 são introduzidas através de cabeçalhos opcionais ou de extensão que são colocados imediatamente a seguir ao cabeçalho IPv6. A norma admite que sejam colocados vários dos cabeçalhos opcionais seguidos no mesmo *datagrama* IPv6 desde que os mesmos não sejam incompatíveis entre si.

Estes cabeçalhos permitem a introdução de funcionalidades opcionais como por exemplo: *source routing* e outros processamentos ditos *hop-by-hop*, e fragmentação mas só extremo a extremo. A *Jumbo Payload Option* permite aos extremos enviar pacotes com mais de 64 Kbytes (65535 bytes), também chamados de *Jumbograms*, e os cabeçalhos de segurança introduzem autenticação e cifra dos pacotes. Com excepção dos cabeçalhos com as opções de encaminhamento e outras ações do tipo *hop-by-hop*, os mecanismos introduzidos por estes cabeçalhos de extensão são extremo a extremo e podem, no essencial, ser ignoradas pelos *routers* intermédios.

O protocolo IP é a “cola” ou “camada unificadora” da Internet pois permite que todos os equipamentos que o suportam possam trocar pacotes de dados entre si. O cabeçalho IP na versão 4, a versão mais comum actualmente, para além dos endereços IP origem e destino do pacote, contém um conjunto de campos que apoiam o encaminhamento dos mesmos, nomeadamente nas facetas relacionadas com a prevenção de ciclos de encaminhamento (TTL) e qualidade de serviço.

Devido à falta de experiência e às características tecnológicas das redes quando o protocolo foi definido, o cabeçalho IPv4 inclui igualmente o suporte de mecanismos que se vieram a revelar pouco amadurecidos, como por exemplo a fragmentação, e o facto de o cabeçalho ser de comprimento variável devido às presenças no mesmo de várias opções (*e.g., source routing, record route, etc.*). Na prática, as opções IPv4 pouco ou nada são suportadas e a fragmentação não é recomendada.

A versão 6 do protocolo IP introduz uma alteração fundamental ao nível da dimensão e formato dos endereços IP, e supriu alguns mecanismos que se revelaram inúteis ou contraproducentes em IPv4. Adicionalmente, substituiu as opções por uma eventual pilha de cabeçalhos de extensão, sempre que são necessárias funcionalidades opcionais semelhantes às opções do IPv4, ou ainda novas opções de segurança, *jumbograms, etc.* Essas opções são fundamentalmente *end-to-end* e pouco utilizadas tal como em IPv4.

## 17.3 Encaminhamento de pacotes IP

Nesta secção vamos penetrar nos detalhes concretos de como é realizado o encaminhamento de pacotes IP. Começaremos por analisar como os sistemas nos extremos da rede recebem endereços e colaboram no encaminhamento. Será então posto em evidência como eles são poupadados às principais complexidades do encaminhamento e em particular como se evita que tenham de conhecer todos os destinos possíveis, mesmo quando estão ligados a mais do que um *router*. Veremos também como é realizado o processamento dos pacotes pelos *routers*.

### Planos de endereçamento IP

Todos os equipamentos que enviam e recebem pacotes IP dispõem de interfaces para canais. **Essas interfaces têm necessariamente um endereço IP específico.** Assim, para além do endereço *loopback*, qualquer um destes equipamentos tem tantos endereços IP diferentes quanto as suas interfaces activas. De facto, os endereços IP não estão associados a equipamentos mas sim a interfaces. O endereço IP associado à interface, é distinto e não substituí qualquer endereço de nível MAC (nível canal) que a interface também tenha.

Os endereços IP das interfaces pertencem aos prefixos IP do canal a que as mesmas estão ligados. Com efeito, **todos os canais usados no encaminhamento IP têm associado um prefixo IP específico.** Assim, se o canal é ponto-a-ponto, esse prefixo contém geralmente pelo menos 4 endereços: o primeiro endereço do prefixo, um endereço para cada extremidade, e um endereço dito endereço de *broadcast*. Quando o canal é um canal em difusão, *i.e.*, multi-ponto, o seu prefixo tem de acomodar tantos endereços quantos os sistemas acessíveis pelo canal mais os dois endereços dos extremos (o do prefixo e o de *broadcast*)<sup>7</sup>. O endereço de *broadcast* só existe em IPv4 e serve

---

<sup>7</sup> Na verdade o endereço do prefixo também pode ser usado por uma das interfaces ligadas

para endereçar todos os sistemas ligados ao mesmo canal e com o mesmo prefixo. Em IPv6 não existem endereços deste tipo e utilizam-se endereços IP Multicasting para o mesmo efeito.

Do ponto de vista do IP a noção de canal multi-ponto abrange qualquer infra-estrutura que permita endereçar directamente todos os sistemas finais ligados à mesma, como por exemplo uma rede de *switches* Ethernet. O nível IP considera uma tal rede, mesmo constituído por dezenas de *switches* Ethernet ligando centenas de computadores, equivalente a um canal multi-ponto pois os pacotes podem ser endereçados directamente a outro sistema ligado à mesma infra-estrutura sem necessidade de operações de encaminhamento IP via algum *router*. Para isso basta encapsular o pacote num *frame* do nível canal. Ou seja, o nível IP equipara essa infra-estrutura a um canal simples. No *calão IP* este tipo de canal chama-se uma ***subnet IP*** ou sub-rede IP.

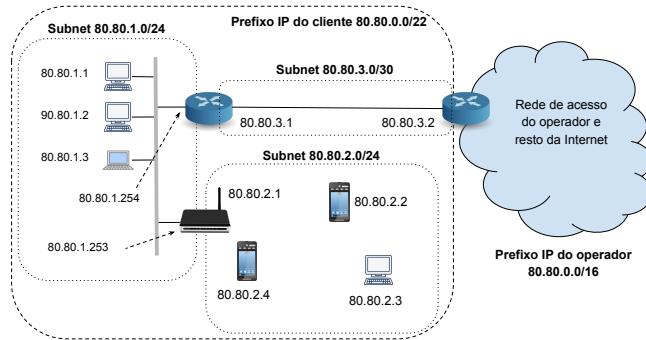


Figura 17.9: Plano de endereçamento de uma rede IP de um cliente ligado a um operador Internet

A Figura 17.9 mostra o plano de endereçamento de uma rede de uma instituição ligada à Internet via um operador (um ISP). O ISP recebeu da RIR da sua região o prefixo IP 80.80.0.0/16. Deste afetou ao cliente o sub-prefixo 80.80.0.0/22, que cobre os endereços de 80.80.0.0 a 80.80.3.255, para sua utilização interna. Como o cliente não está ligado a outro ISP, o seu espaço de endereçamento é *provider dependent address space* e só o prefixo do operador é anunciado na Internet global.

Internamente o cliente tem 3 *subnets*. Uma correspondente ao canal ponto-a-ponto que liga o seu *router* ao *router* do ISP, com o prefixo 80.80.3.0/30. O prefixo comporta 4 endereços mas só dois deles são usados para numerar as interfaces dos dois *routers*. A segunda *subnet* corresponde a um canal multi-ponto baseado numa infra-estrutura Ethernet que recebeu o prefixo 80.80.1.0/24. Todos os computadores e *routers* ligados a essa *subnet* têm endereços distintos nesse prefixo. Repare-se que os equipamentos com duas interfaces (o *router* e um *access point* de uma rede WiFi) têm mais do que um endereço IP distinto, cada um em *subnets* distintas e portanto em prefixes distintos. A terceira e última *subnet* corresponde à rede WiFi e tem o prefixo 80.80.2.0/24. O *access point* tem a interface rádio, que dá acesso ao canal WiFi, com um endereço neste prefixo. O cliente ficou com o prefixo 80.80.0.0/24 livre assim como a parte restante do prefixo 80.80.3.0/24, de 80.80.3.4 a 80.80.3.255.

Este plano de numeração vai repercutir-se nas tabelas de encaminhamento dos diferentes equipamentos ligados a esta rede. Por exemplo, a tabela de encaminhamento do computador com o endereço 80.80.1.3 na *subnet* da esquerda é a apresentada na Figura 17.10.

---

ao canal.

Prefixo IP	Endereço do gateway (next hop)	Interface	Número de hops
80.80.1.3/32	local	eth0	0
80.80.1.0/24	directo	eth0	0
80.80.2.0/24	80.80.1.253	eth0	1
80.80.3.0/30	80.80.1.254	eth0	1
0.0.0.0/0	80.80.1.254	eth0	
...	...	...	...

Figura 17.10: Tabela de encaminhamento do computador com o endereço 80.80.1.3

A tabela de encaminhamento dos computadores pode ser consultada dando o comando `netstat -r` na consola ou na linha de comando da grande maioria dos sistemas de operação.

Esta tabela contém entradas que permitem enviar pacotes para o próprio computador via a sua interface Ethernet (a primeira entrada com o prefixo 80.80.1.3/32). A segunda entrada, indexada pelo prefixo do canal (80.80.1.0/24), permite enviar pacotes directamente para equipamentos ligados ao mesmo canal de difusão que o próprio computador. Esta forma de encaminhamento chama-se **encaminhamento directo (direct routing)**.

As duas entradas seguintes, com os prefixos (80.80.2.0/24 e 80.80.3.0/30) permitem enviar pacotes para as outras *subnets* do cliente visto que são indexadas pelos respectivos prefixos. Estas entradas contém o endereço do *gateway / next-hop* que é o endereço de um *router*, acessível por um canal ligado a este computador, que “sabe” encaminhar para as *subnets* com aqueles prefixos. Esta forma de encaminhamento chama-se **encaminhamento indirecto (indirect routing)**.

Finalmente, a última entrada (indexada pelo prefixo 0.0.0.0.0/0 ou universo) corresponde uma forma especial de encaminhamento indirecto que se chama **encaminhamento por defeito ou omissão (default routing)**. Esta entrada permite encaminhar pacotes cujo endereço de destino é qualquer outro.

O processamento de um pacote por um computador ou um *router* IPv4 consiste, grosso modo, nas operações indicadas no algoritmo 17.1. Os métodos `packet.*_route()`, após selecionarem a interface pela qual o pacote deve ser transmitido, deverão implementar a fragmentação se a mesma for necessária, decrementar o TTL e recalcular o *checksum*. O encaminhamento realizado por cada um é descrito a seguir.

Em IPv6 o processamento é grosso modo idêntico excepto que não há teste do *checksum*, o processamento das opções é transformado no processamento dos cabeçalhos opcionais, e se for necessária fragmentação, o pacote é ignorado e desencadeia-se um processamento de erro.

### Encaminhamento local

O encaminhamento local é uma excepção pois o pacote é de novo entregue ao software IP local para processamento de um pacote a ele destinado. O mesmo acontece com os pacotes dirigidos ao endereço *loopback*.

### Encaminhamento directo

Este tipo de encaminhamento tem lugar quando o endereço de destino do pacote está contido no prefixo IP associado a um canal ligado ao equipamento. Na rede que já foi introduzida mais acima, ver a Figura 17.11, é o caso quando o computador com o

Listing 17.1: Pseudo código a alto nível do algoritmo de encaminhamento de um pacote IP (*packet*)

```

1 if ( ! packet.checksum_ok() ) { // ignore packet
2   packet.process_error()
3   return
4 }
5 if ( packet.getTTL() - 1 == 0 ) { // end of packet life
6   packet.process_error()
7   return
8 }
9 if ( packet.header_length() > 5 ) { // packet has options
10  packet.process_options()
11  return
12 }
13 d = packet.getDestAddress()
14 r = routing_table.get_longest_prefix_match( d )
15 if ( r == null ) { // cannot route packet
16   packet.process_error()
17   return
18 }
19 if ( r.type() == local ) packet.locally_route(r, d)
20 else if ( entry.type() == direct ) packet.directly_route(r, d)
21 else if ( entry.type() == indirect ) packet.indirectly_route(r, d)

```

endereço IP 80.80.1.3 pretende enviar um pacote para o computador com o endereço 80.80.1.1.

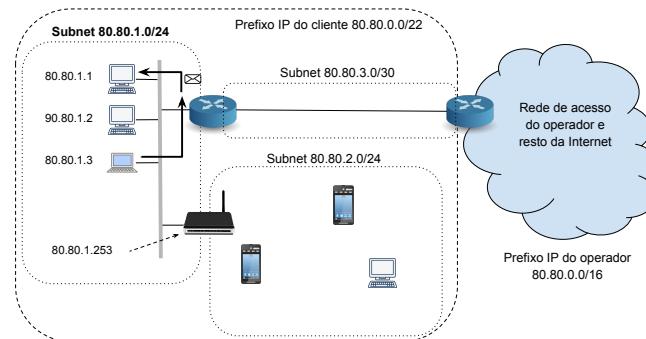


Figura 17.11: Envio de um pacote do computador com o endereço 80.80.1.3 para o com o endereço 80.80.1.1 através de encaminhamento directo.

Com o encaminhamento directo, se o canal é ponto-a-ponto basta transmitir o pacote para a outra extremidade do canal encapsulado num *frame* adequado. Se o canal é em difusão (*e.g.*, Ethernet com fios ou WiFi) ou uma rede baseada em inundação, *i.e.*, uma infra-estrutura de comunicação multi-ponto, (*e.g.*, switched Ethernet) o pacote deve ser transmitido directamente encapsulado num *frame* dirigido ao endereço MAC do dispositivo ao qual está associado o endereço IP de destino. A Figura 17.12 mostra, a título de exemplo, a entrada (a) que é a seleccionada na tabela de encaminhamento para envio directo de um pacote do computador de endereço IP 80.80.1.3 para o computador com o endereço 80.80.1.1 usando o algoritmo *longest prefix matching* pois o endereço está contido em vários dos prefixes da tabela (nomeadamente o (a) e o (c)).

Para realizar este envio directo via um canal em difusão ou uma infra-estrutura

	Prefixo IP	Endereço do gateway ( <i>next hop</i> )	Interface	Número de <i>hops</i>
	80.80.1.3/32	local	eth0	0
(a)	80.80.1.0/24	directo	eth0	0
(b)	80.80.2.0/24	80.80.1.253	eth0	1
	80.80.3.0/30	80.80.1.254	eth0	1
(c)	0.0.0.0/0	80.80.1.254	eth0	
	...	...	...	...

Figura 17.12: Entrada na tabela de encaminhamento do computador de endereço 80.80.1.3 que é selecionada por *longest prefix matching* para envio de pacotes: para (a) 80.80.1.1 por envio directo; (b) para 80.80.2.3 por envio indirecto; e (c) para 193.136.126.43 por envio por *default*.

de comunicação multi-ponto, o software IP e de controlo do canal têm de conhecer qual é o endereço MAC da interface associada ao endereço IP do destino. É possível resolver este problema de tradução de endereços de várias formas, mas a implementação corrente para este tipo de canais é usar o protocolo *Address Resolution Protocol* (ARP) que será explicado a seguir.

### Encaminhamento indirecto

Para envio de um pacote para o endereço 80.80.2.3 a entrada seleccionada pelo algoritmo *longest prefix matching* é a (b) da Figura 17.12, pois o endereço está contido em vários dos prefixos da tabela (nomeadamente o (b) e o (c)). Esta entrada indica como *next hop* o endereço da interface do equipamento onde começa o caminho para o destino. Este endereço tem de estar directamente acessível por encaminhamento directo, *i.e.*, acessível directamente por um canal ao qual o emissor do pacote esteja ligado. Tal é o caso pois o endereço 80.80.1.253 está no mesmo prefixo que um dos canais do emissor, logo é acessível por encaminhamento directo. Assim, o emissor encapsula o pacote num *frame Ethernet* e envia-o para o endereço MAC do *gateway*, *i.e.*, o endereço MAC associado ao endereço 80.80.1.253 do AP. Este endereço MAC pode-se obter via o protocolo ARP caso o mesmo não seja ainda conhecido pelo emissor.

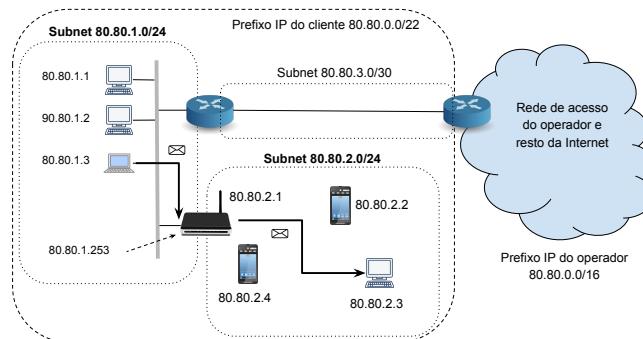


Figura 17.13: Envio de um pacote do endereço 80.80.1.3 para o endereço 80.80.2.3 através de encaminhamento indirecto.

Depois de o pacote chegar à interface com o endereço 80.80.1.253, o software IP do AP aplica o algoritmo 17.1 ao destino do pacote e concluirá que o destino é

acessível localmente por encaminhamento directo. Assim, retransmite-o encapsulado num *frame* Ethernet dirigido ao endereço MAC do computador com o endereço IP 80.80.2.3.

### Encaminhamento por omissão ou *default*

Para envio de um pacote para o endereço 193.136.126.43, a entrada seleccionada pelo algoritmo *longest prefix matching* é a (c) na Figura 17.12 pois o endereço só está contido nesse prefixo. Este encaminhamento indirecto especial diz-se por omissão ou *default routing* pois este prefixo é uma espécie de encaminhamento pela via do “último recurso”. Na verdade o mesmo pressupõe que o *router* intermediário tem possibilidades de chegar a todos os destinos válidos.

Apesar de o prefixo envolvido ser especial, trata-se de encaminhamento indirecto convencional e a entrada tem de conter um endereço de *next-hop* / *gateway* directamente acessível. Tal é efectivamente o caso do endereço 80.80.1.254 associado ao *router* de fronteira do cliente. O pacote é encapsulado num *frame* Ethernet dirigido ao endereço MAC associado ao endereço IP 80.80.1.254 e o pacote é transmitido para esse *router*, ver a Figura 17.14.

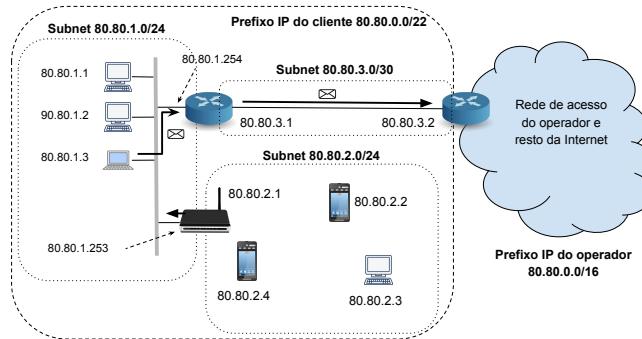


Figura 17.14: Envio de um pacote do endereço 80.80.1.3 para o endereço 193.136.126.43 através de encaminhamento por *default*.

Uma vez o pacote chegado ao *router* este consultará a sua tabela de encaminhamento, apresentada na Figura 17.15, e constatará que o seu endereço de destino apenas faz *matching* com a entrada correspondente à rota *default*, que indica que o *next-hop* está no endereço 80.80.3.2, acessível directamente pela interface do canal que liga o cliente ao ISP.

Prefixo IP	Endereço do gateway ( <i>next hop</i> )	Interface	Número de hops
80.80.1.254/32	local	eth0	0
80.80.1.0/24	directo	eth0	0
80.80.2.0/24	80.80.1.253	eth0	1
80.80.3.1/32	local	eth1	0
80.80.3.0/30	directo	eth1	0
0.0.0.0/0	80.80.3.2	eth1	

Figura 17.15: Tabela de encaminhamento (FIB) do *router* de fronteira do cliente

Este exemplo mostra toda a potência do mecanismo dos prefixos IP e do algoritmo *longest prefix matching*. Em particular, fica evidente que o encaminhamento é profundamente hierárquico e que os diferentes equipamentos apenas necessitam de tabelas de encaminhamento com um número de entradas proporcional ao número de *subnets* distintas que têm de conhecer de forma discriminada. O prefixo 0.0.0.0/0, associado ao encaminhamento por omissão, representa de forma comprimida “o resto do mundo”.

Naturalmente, no interior da rede do ISP o número de prefixos conhecidos será muito maior pois as redes de acesso têm de conhecer os prefixos de todos os clientes a ela ligados. Adicionalmente, se o ISP for *multi-homed*, o seu *backbone* terá de conhecer todos os prefixos visíveis no *core* da Internet, pois nesse caso não é comum existirem rotas por omissão.

A região da Internet onde já não são usadas rotas por omissão, *i.e.*, na região de interligação dos operadores, é designada por esse motivo *Default-Free Zone* (DFZ). As tabelas de encaminhamento dos *routers* desta zona têm de contar todos os prefixos usados na Internet, e se o endereço de destino de um pacote não está contido em nenhum, então o pacote não pode ser encaminhado.

## 17.4 Protocolos auxiliares do IP

Para que uma rede IP funcione, este protocolo é complementado com vários protocolos auxiliares de descoberta de vizinhos e envio de informação de controlo e meta-informação: ARP, ICMP e DHCP. No final da secção discute-se ainda o conceito de endereços privados e o seu papel.

### *Address Resolution Protocol (ARP)*

O software IP dos sistemas de operação dos computadores e dos *routers* dispõe de uma tabela, chamada **tabela ARP (ARP table)**, que mapeia endereços IP em endereços MAC. Quando se procura o endereço MAC correspondente a um endereço IP e o mesmo não está presente nessa tabela, é usado o protocolo ARP para o obter. Este protocolo comprehende apenas duas mensagens: *ARP query* e *ARP reply*. A mensagem *ARP query* é enviada em difusão pelo canal (por isso o ARP só funciona em canais ou infra-estruturas com suporte de *broadcasting*). Essa interrogação é realizada enviando um *frame ARP query* com a pergunta: `Who has IP address x, asks host with IP address y?` Esta questão é difundida a todos os sistemas ligados ao canal, e se existir um com aquele endereço IP, este responde através do envio de um *frame ARP reply* dirigido ao endereço MAC origem da interrogação, ver a Figura 17.16. Ou seja, a interrogação é enviada em *broadcast* e a resposta em *unicast*.

Naturalmente, sempre que obtém o endereço MAC associado a um endereço IP via o protocolo ARP, o software IP coloca-o na tabela ARP para não a ter de pedir sempre que esta é necessária. Para evitar que a tabela tenha informação desactualizada, a cada entrada está associado um alarme de alguns segundos e, quando esse tempo se esgota, a entrada é suprimida da tabela para evitar conter informação que provavelmente já não seria válida. Isto é, o protocolo não comporta nenhuma mensagem para invalidar associações entre endereços IP e endereços MAC e a gestão da tabela apenas garante que a informação nela contida era válida quando lá foi colocada. A informação contida na tabela ARP diz-se então ser *soft state*<sup>8</sup>.

Na maioria dos sistemas de operação a tabela de ARP é acessível através de comandos na consola, como por exemplo `arp -a` nos sistemas derivados de Unix (Linux,

---

<sup>8</sup> O que sugere que o termo *hard state* é reservado para situações em que a informação é permanentemente mantida de forma consistente, o que se revela geralmente mais complexo e com custos acrescidos.

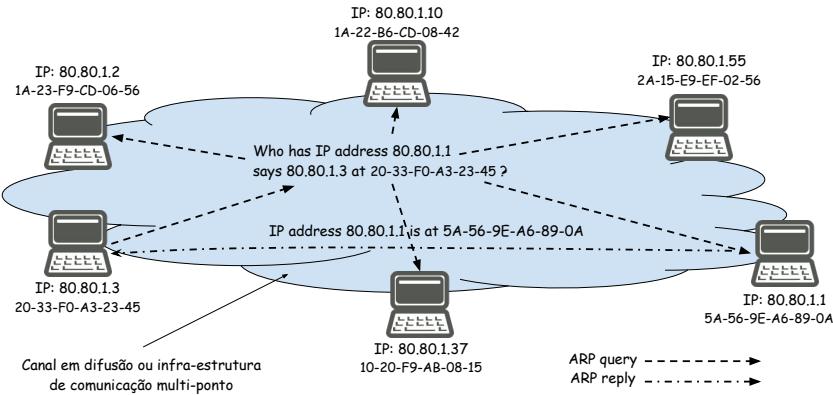


Figura 17.16: Funcionamento do protocolo ARP quando o computador com endereço IP 80.80.1.3 pretende descobrir o endereço MAC do computador com o endereço IP 80.80.1.1

Mac OS X, iOS, Android, ...). Em alguns destes sistemas as tabelas de encaminhamento e a tabela de ARP estão fundidas na mesma tabela. As implementações de ARP também utilizam as *ARP queries* para enriquecerem a tabela de ARP com os endereços IP e MAC do computador que emite a interrogação, visto que ambas as informações são acessíveis no *frame ARP query* que é recebido por todos os sistemas ligados ao canal multi-ponto.

O protocolo ARP não comporta nenhuma forma de autenticação, pode ser facilmente violado, e levar à propagação de informação errada. Basta, por exemplo, que um computador emita *frames ARP query* com endereços IP origem falsos. Por esta razão, este protocolo não é adequado a situações onde o aparecimento deste tipo de ataques seja frequente e possa resultar em consequências graves. Trata-se de mais um protocolo desenvolvido para *subnets* com baixa probabilidade de presença de intrusos.

#### **Dynamic Host Configuration Protocol (DHCP)**

Como vimos nos exemplo anteriores, um computador para poder comunicar em IP necessita de obter endereço(s) IP e preencher a sua tabela de encaminhamento. O mesmo problema se coloca quando se trata de parametrizar os *routers* da uma rede.

No início da Internet essa parametrização era realizada de forma manual. Quando se ligava mais um computador a uma rede IP era necessário saber qual o seu endereço IP, o prefixo IP do canal de ligação à rede, o endereço do *router* que representava a rota por omissão e os endereços de um ou mais servidores de DNS.

Mais tarde este processo de inicialização do software IP foi automatizado através do protocolo *Boot Protocol* que foi depois substituído pelo *Dynamic Host Configuration Protocol* ou DHCP, ver o RFC 1541. O mesmo é hoje em dia imprescindível pois, sempre que um computador móvel se liga a uma rede diferente, ou muda de rede, o software IP do mesmo tem de ser reinicializado.

Os *routers* que ligam as redes domésticas aos ISPs também utilizam o protocolo DHCP para obterem as informações necessárias ao seu correcto funcionamento. No entanto, muitos equipamentos de rede dos *backbones* continuam a ser parametrizados manualmente.

Quando um computador activa a interface de um canal que o liga à rede, esse canal é quase sempre um canal multi-ponto, ou um canal que o liga a uma infra-estrutura equiparada. Por essa via é pressuposto ser possível enviar pacotes para um

ou mais **servidores DHCP**. O protocolo comporta 4 mensagens: **Discovery**, **Offer**, **Request** e **Acknowledge**. De forma muito resumida, o cliente começa por tentar localizar um servidor DHCP enviando em difusão uma mensagem do tipo **Discovery**. Os servidores DHCP acessíveis deverão responder com mensagens **Offer**, propondo um endereço IP ao cliente e enviando outras informações relevantes. O cliente escolhe uma das ofertas de endereço recebidas e envia uma mensagem **Request** indicando o endereço escolhido. Por razões que ficarão mais claras a seguir, esta mensagem também é enviada em difusão. O servidor, caso confirme a oferta, deve enviar uma mensagem **Acknowledge** que confirma a afectação. A Figura 17.17 ilustra o processo.

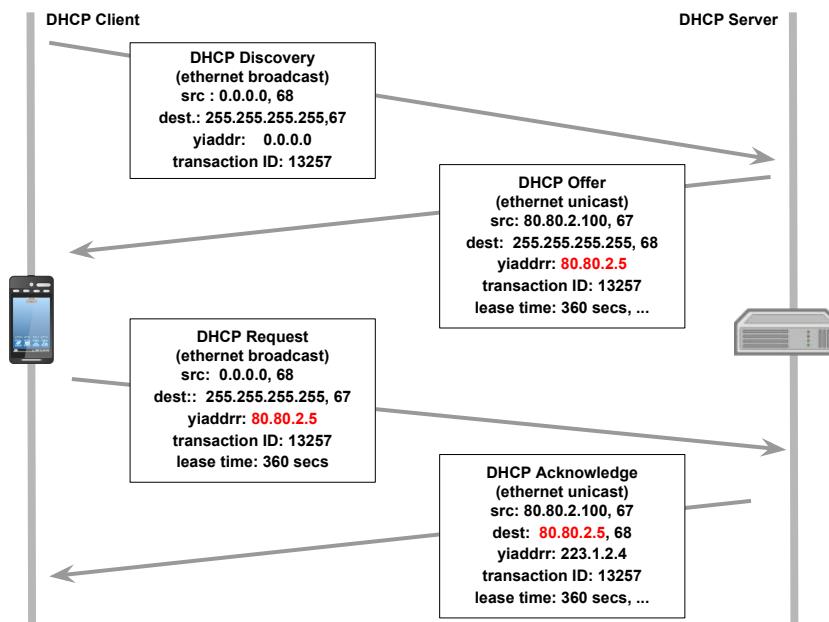


Figura 17.17: Funcionamento do protocolo DHCP

As mensagens DHCP são enviadas sobre UDP. A porta do cliente é a porta 68 e a porta do servidor é a 67. A mensagem **Discovery** tem por endereço origem o endereço indefinido ou 0 (0.0.0.0) pois o cliente ainda não conhece o seu endereço. O endereço de destino é o endereço de *broadcast* (255.255.255.255). A mensagem pode ser enviada mesmo sem o cliente conhecer o seu endereço IP porque a sua interface conhece o respectivo endereço MAC (afectado na fábrica) e o destino é o endereço de difusão, quer ao nível MAC, quer ao nível IP. O parâmetro **Transaction ID** é colocado pelo cliente para lhe permitir relacionar as respostas com os pedidos de forma mais segura. O parâmetro **yiaddr** (*Your IP Address*) é onde o servidor pode colocar o endereço oferecido ao cliente. No caso da mensagem inicial o seu valor é 0 (0.0.0.0).

O servidor responde com uma mensagem **Offer** propondo um endereço IP e enviando outros parâmetros úteis ao cliente (*e.g.*, prefixos IP, endereços de servidores DNS, rotas por omissão, *etc.*). Adicionalmente, o servidor envia um tempo de validade da oferta, este parâmetro é designado por *lease time*. A resposta é enviada em *unicast* Ethernet.

Como o cliente pode receber ofertas de vários servidores, como por exemplo um primário e outro de *backup*, o cliente deve escolher uma das ofertas e reafirmá-la envi-

ando uma mensagem **Request** em difusão, o que permitirá aos servidores não escolhidos cancelarem as suas ofertas. Tudo termina com uma mensagem **Acknowledgment** enviada em *unicast* pelo servidor escolhido ao cliente DHCP. Caso se percam mensagens, compete ao cliente reemitir os pedidos.

O protocolo DHCP comporta uma mensagem **Release** mas o seu envio pelo cliente é opcional pelas razões indicadas a seguir. O parâmetro *lease time* permite ao servidor reutilizar o endereço se este período expirar e o cliente não renovar o pedido, o que pode fazer enviando uma mensagem de **Request** a renovar o interesse na afectação do endereço.

Este simples mecanismo evita que o servidor se tenha de preocupar com o desaparecimento abrupto do cliente por este ter tido uma avaria ou pura e simplesmente ter sido desligado. Trata-se de outro exemplo de informação gerida por *soft state*, que é simples mas não dá garantias de validade. Basta pensar no caso em que o servidor DHCP falha e é reinicializado, ou substituído por outro, antes de os *lease time* das afectações anteriores se tenham esgotado. Uma solução para este problema é proposta em exercícios no fim do capítulo, ver a Secção 17.6.

O protocolo admite inúmeras opções introduzidas para permitir transmitir aos clientes as mais variadas informações. Em contrapartida, o DHCP não usa nenhuma mecanismo de segurança. Isso permite inúmeros ataques como por exemplo os baseados em servidores DHCP a enviarem informações falsas, clientes a fazerem ataques de negação de serviço para esgotarem os endereços disponíveis e clientes que usam endereços que não lhes foram afectados. Para combater estes problemas foram introduzidas variantes com mecanismos de segurança mas as mesmas não foram adoptadas de forma generalizada. Quando o DHCP é usado entre os ISPs e os seus clientes, são usados mecanismos complementares de autenticação dos clientes a outros níveis.

Para terminarmos esta discussão de protocolos de apoio ao funcionamento do IP, vamos a seguir analisar o protocolo ICMP.

### ***Internet Control Message Protocol (ICMP)***

Por opção conceptual, ver a Secção 4.1, o protocolo IP não obriga estritamente os *routers* IP a assinalarem aos seus parceiros e outros sistemas ligados à rede quando acontecem eventos que conduzem a erros ou outras situações equiparáveis (*e.g.*, TTLs esgotados, pacotes suprimidos por as filas de espera estarem cheias, destinos desconhecidos, pacotes mal formados, *etc.*).

No entanto, existe um protocolo chamado *Internet Control Message Protocol (ICMP)*, definido inicialmente pelo RFC 792 e referido no RFC 1122 e outros, que permite enviar notificações de erro. Adicionalmente, o ICMP permite igualmente troca de (meta-)informação entre equipamentos IP. As mensagens ICMP são transportadas directamente dentro de pacotes IP, são enviadas para o endereço IP do dispositivo que originou o erro, ou fez o pedido de informação, e não são endereçadas a portas pois devem ser tratadas directamente pelo software do nível IP. A Figura 17.18 mostra a parte comum do cabeçalho ICMP.

O campo **Type** organiza as mensagens ICMP em diversos tipos e o campo **Code** em variantes dentro dos tipos. Conforme o tipo das mensagens, o resto do cabeçalho pode variar e a parte de dados também. A tabela 17.3 lista alguns exemplos de tipos e códigos ICMP.

É fácil reconhecer o papel de alguns dos tipos de mensagens ICMP. Por exemplo, o tipo 8 (*Echo request*) e o tipo 1 (*Echo reply*) podem ser usados pelo programa **ping**, ver a Secção 3.3. O funcionamento do programa **traceroute**, ver a Secção 3.4, repousa nas mensagens ICMP de tipo 11 (*Time Exceeded - TTL expired in transit*). A mensagem de tipo 3, código 4 (*Fragmentation required, and DF flag set*) permite saber que era necessário realizar a fragmentação de um pacote mas que, dado o posicionamento de uma *flag* específica no cabeçalho do pacote, o *router* suprimiu-o e enviou uma

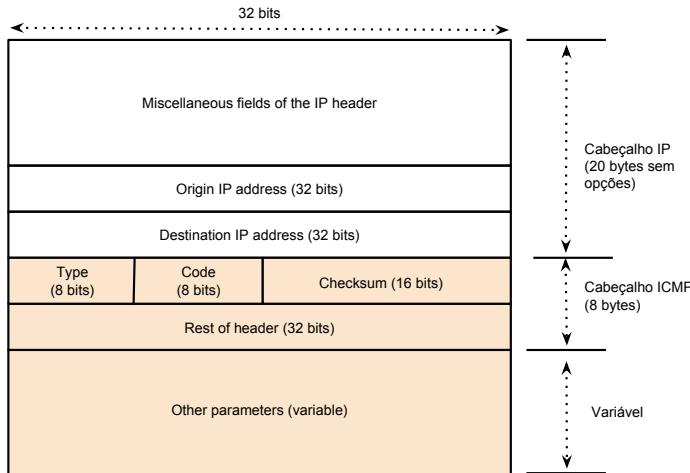


Figura 17.18: Formato das mensagens do protocolo ICMPv4

notificação ICMP. Este tipo e código estão na base do protocolo *Path MTU discovery*, ver o RFC 1191.

Algumas mensagens ICMP servem para controlo. Por exemplo, as mensagens com os tipos 13 e 14 (*Time request / Reply*) podem ser usadas para medir o tempo de trânsito com maior rigor entre dois equipamentos e os tipos 11 e 10 (*Router discovery / advertisement*) podem ser usados para um sistema terminal descobrir um *router* no canal a que está ligado e instalar uma rota por omissão.

A definição exacta do papel de cada mensagem e da forma como estas devem ser processadas pelo emissor e pelo receptor é objecto de descrição muito detalhada nos RFCs que lhes dizem respeito. O leitor deverá consultar esses documentos para conhecer os detalhes. Um aspecto comum a um grande conjunto de tipos de mensagens é o facto de que a sua emissão é opcional, *i.e.*, se um *router* suprime um pacote por o TTL ter expirado, o envio da mensagem ICMP 11 é opcional. Na realidade muitos *routers* estão preparados pelos seus administradores para ignorarem certas mensagens ICMP ou para não enviarem respostas às mesmas. Esta possibilidade é acomodada pelo carácter *best effort* do IP, ver a Secção 4.1. Muitas vezes a motivação para parametrizar os *routers* para não enviarem ou responderem a algumas mensagens ICMP tem a ver com razões de desempenho e segurança.

Muitas das mensagens ICMP têm cabeçalhos complementares e transportam informação que permite ao receptor relacionar uma resposta com um pedido. Por exemplo, as mensagens *Echo Request / Reply* têm campos para serem colocados números de sequência que suportam envios sucessivos e também é pressuposto que a parte de dados do *request* seja copiado para a parte de dados do *reply*. As mensagens ICMP com notificações de erros (*e.g., Host unreachable*) devem conter na parte de dados o cabeçalho do pacote IP que deu origem ao problema e os primeiros 8 bytes do respectivo *payload*.

### ARP, ICMP e DHCP em IPv6

Os protocolos ARP, ICMP e DHCP em IPv6 foram introduzidos ao longo dos anos à medida que a experiência e as necessidades de funcionamento e parametrização do IPv6 iam crescendo. Por esta razão existem algumas sobreposições entre esses protocolos e

Tabela 17.3: Alguns exemplos de tipos e códigos do ICMP

Type	Code	Descrição
1	0	Echo reply
3	0	Destination unreachable - Network unreachable
3	1	Destination unreachable - Host unreachable
3	3	Destination unreachable - Port unreachable
3	4	Destination unreachable - Fragmentation required, and DF flag set
3	5	Destination unreachable - Source route failed
8	0	Echo request
9	0	Router advertisement
10	0	Router discovery/selection/solicitation
11	0	Time Exceeded - TTL expired in transit
11	1	Time Exceeded - Fragment reassembly time exceeded
13	0	Time request
14	0	Time reply

vários deles dependem da disponibilidade directa de *broadcasting* no canal. Em IPv6, usando a experiência anterior, foi possível passar a limpo a definição destes protocolos.

O suporte da comunicação multi-ponto em IPv6 foi restringido à funcionalidade de IPv6 *multicasting* (não existe *broadcasting*). A definição de IPv6 *multicasting* comporta a definição de grupos pré-definidos (tal como o IPv4 *multicasting*) como por exemplo todos os nós acessíveis por um canal (ou infra-estrutura) multi-ponto, todos os *routers* ligados a um canal (ou infra-estrutura) multi-ponto, destes, todos os com suporte de OSPF, etc.

Baseado nas funcionalidades do IPv6 *multicasting*, o IPv6 acrescentou ao protocolo ICMPv6, como descrito no RFC 2461, um subconjunto de mensagens que servem para *Neighbor Discovery* (ND) e que integram as funcionalidades dos protocolos ARP e parte das do DHCP. Parte das funcionalidades de ND implementadas pelo ICMPv6 são as seguintes: *Router Discovery*, *Prefix Discovery*, *Parameter Discovery*, *Address Resolution*, *Duplicate Address Resolution*, *Redirect*, etc..

Com estas funcionalidades, o ICMPv6 substitui integralmente o ARP e permite que uma interface se auto-configure correctamente sem recurso a nenhum servidor DHCP usando *stateless auto-configuration*: conhecendo o prefixo da *subnet* do canal é possível gerar um endereço IPv6 usando o endereço MAC da interface como sufixo (os últimos 48 bits dos 128 bits do endereço). Apesar disso, o IPv6 também dispõe do DHCPv6, ver o RFC3315 e outros, que permite afectar endereços e outros parâmetros fornecidos por um servidor.

Não existe um fosso conceptual entre a operação destes protocolos em IPv4 e em IPv6, mas existem bastantes diferenças pois em IPv6 pretendeu-se favorecer a parametrização dinâmica de redes sem recurso a servidores e com pouca intervenção dos gestores da mesma. Por essa razão é preferível o leitor consultar documentação especializada como por exemplo as referências incluídas na Secção 17.5. Para terminar esta secção introduzem-se a seguir os endereços privados e a tradução de endereços.

### Endereços privados e tradução de endereços

Existem diversas razões que tornam desejável implementar uma rede cujo espaço de endereçamento seja privado, i.e., de alguma forma não visível do exterior da mesma. Historicamente esta solução teve como principal motivação poupar endereços IPv4. A ideia era permitir que vários computadores conseguissem partilhar entre si um único endereço IPv4 ligado à Internet. Definiu-se então o conceito de **espaço de endereçamento privado** já referido neste capítulo, (ver a Secção 17.1), ou seja, prefixos que não podem ser encaminhados na Internet global, e que podem ser reutilizados

entre diferentes redes privadas pois não são visíveis fora dessas redes. Uma tal solução requer alguma forma de **tradução entre endereços privados e públicos**, i.e., entre endereços privados e os endereços regulares, sob pena de os sistemas internos não poderem comunicar com o exterior da rede.

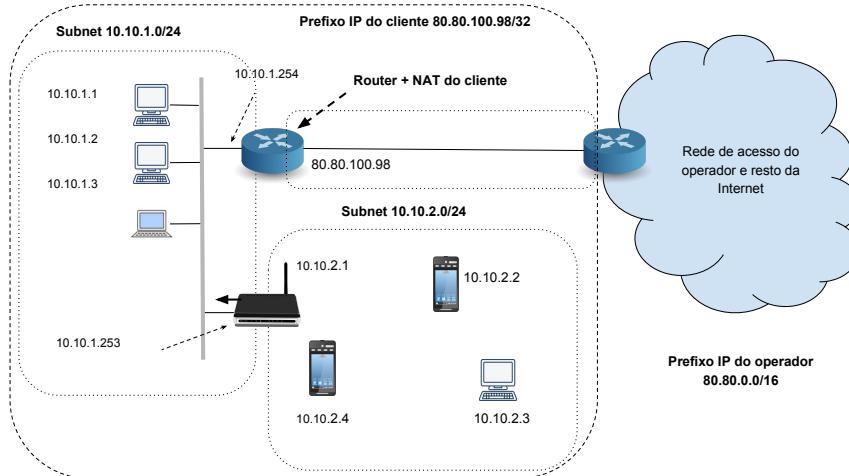


Figura 17.19: Rede de um cliente baseada em endereçamento privado

A Figura 17.19 apresenta a já nossa conhecida rede de um cliente mas que desta vez utiliza o prefixo privado 10.0.0.0/8 para numerar as suas *subnets* internas. O *router* que liga o cliente ao ISP tem uma interface ligada à rede privada e outra interface ligada ao ISP. O ISP desta vez apenas atribui um único endereço público ao cliente, exactamente o endereço da interface do *router* do cliente que termina a ligação deste à rede de acesso do ISP.

Uma outra motivação para usar esta solução tem a ver com o facto de que ela facilita a mudança de ISP. Com efeito, se o cliente não tem espaço de endereçamento dependente de um ISP, se trocar de fornecedor, as alterações estão confinadas ao seu *router* de fronteira assinalado na figura com a sigla NAT de *Network Address Translation*. Toda a sua infra-estrutura interna está isolada e não requer alterações. A terceira motivação só fica mais clara depois de vermos como é que a tradução de endereços pode funcionar.

Parece relativamente simples o *router* do cliente trocar cada endereço interno por um externo. No entanto, esta solução exige tantos endereços públicos quantos os privados. O que é necessário é conseguir utilizar um único endereço IP público para representar vários endereços privados em simultâneo. Isso consegue-se recorrendo a uma porta extra por cada interação (transação) entre um computador da rede privada e um computador na rede pública. A função de tradução de endereços é designada pela sigla NAT (*Network Address Translation*) e o mesmo mecanismo recorrendo a portas, é designado pela sigla NATP (*Network Address Translation with Ports*).

Para percebermos como o mecanismo pode funcionar vamos imaginar que o computador com o endereço IP privado 10.10.1.3 pretende abrir uma conexão TCP para o servidor HTTP no endereço IP público 193.136.126.43. Para tal envia um pacote, ver a Figura 17.20 (a), para o servidor. Seguindo a rota por omissão este chega ao *router* do cliente que então reescreve o cabeçalho do pacote alterando o endereço origem para o seu, e a porta origem para uma que afecta a esta conexão. Finalmente regista esta informação (antigo endereço e porta / novo endereço e porta) na sua tabela de

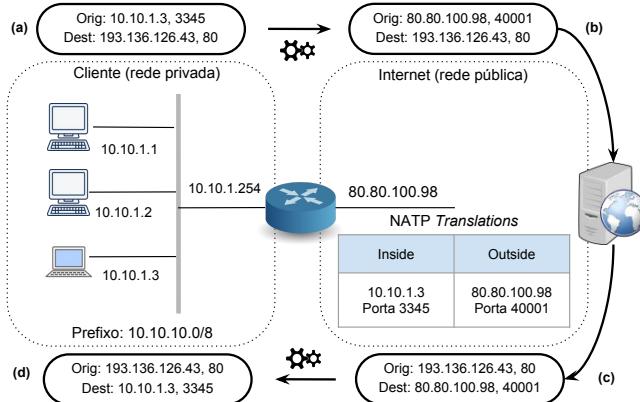


Figura 17.20: Tradução de endereços com base em portas (NATP)

NATP *translations* e transmite o pacote com endereçamento modificado, ver a Figura 17.20 (b).

O servidor processa o pacote e responde com o um novo pacote, ver a Figura 17.20 (c), dirigido ao endereço e porta do *router* NATP. Ao receber o pacote do servidor, através da porta o *router* consegue saber qual o endereço privado e porta para que o pacote tem de ser enviado, reescreve o cabeçalho e este chega ao computador com o endereço IP privado 10.10.1.3, ver a Figura 17.20 (d). Os outros pacotes enviados por ambas as partes sofrem exactamente a mesma transformação enquanto a conexão durar e a associação na tabela se mantiver.

O mecanismo funciona correctamente afectando uma porta diferente do *router* NATP para cada interacção (em UDP, em TCP, em ICMP, ...) iniciada por qualquer computador com endereço IP privado. Para que as entradas na tabela de *NATP translations* não fiquem lá eternamente, é preciso verificar quando as conexões TCP são fechadas, ou usar um alarme (*timeout*) para as interacções UDP. Como os pacotes ICMP não usam portas, o mecanismo é mais complexo para os mesmos. No final do capítulo são propostos exercícios sobre ambos estes problemas, ver a Secção 17.6.

Analizando o mecanismo é possível verificar que o mesmo só permite a existência de transações iniciadas na rede interna. Qualquer pacote que chegue ao *router* NATP para uma porta sem associação não pode entrar na rede privada. Adicionalmente, os *routers* NATP só utilizam para traduções as portas altas (por exemplo entre 40.000 e 60.000), e por outro, estes podem também memorizar para que endereço público é que o pacote associado à entrada foi enviado (o endereço de destino do pacote (b) no exemplo). Qualquer pacote dirigido à porta reservada pelo *router* à conexão, que não venha do endereço público associado (193.136.126.43 no exemplo) é deitado fora<sup>9</sup>. Desta forma a probabilidade de se conseguir enviar da rede pública para os computadores da rede privada, pacotes estranhos a interacções iniciadas na rede privada, fica bastante mais reduzida.

Esta é exactamente a terceira vantagem de se usarem mecanismos NATP pois estes introduzem uma barreira suplementar de segurança (ainda que ténue e às vezes enganadora). Com efeito, o mecanismo descrito implementa algo que está de acordo com o princípio da “*separation of concerns*”, ver a Secção 4.1, entre a rede privada e a rede pública e introduz o conceito de *gateway* de acesso ao espaço privado.

No entanto, pode acontecer que o cliente também quisesse ter um servidor HTTP

<sup>9</sup> Este controlo suplementar também pode criar alguns inconvenientes e restrições quando se pretende implementar mecanismos de NAT *traversal*, como veremos a seguir.

na sua rede interna acessível para clientes na Internet pública. Como exposto, isso não é possível a não ser que se complemente o mecanismo de tradução dinâmica e a pedido com um mecanismo inverso mas permanente. Essa tradução no sentido inverso designa-se **redirecção de portas (port forwarding)** e funciona como se explica a seguir.

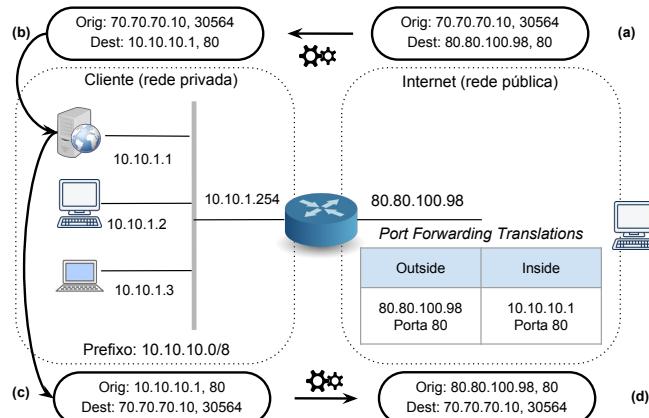


Figura 17.21: Acesso a um servidor interno com base em *port forwarding*

Numa tabela de traduções complementares pode-se associar um endereço privado interno de forma fixa a uma porta (e portanto um só endereço privado pode ser assim associado de forma fixa). Por exemplo, ver a Figura 17.21 (a), quando um cliente na Internet envia um pacote de abertura de uma conexão HTTP sobre TCP para a porta 80 do endereço público do cliente (endereço 80.80.100.98), o pacote chega ao *router* do cliente que reescreve o seu cabeçalho no endereço privado do servidor (endereço 10.10.1.1), ver a Figura 17.21 (b).

Quando o servidor responde ao pacote recebido, ver a Figura 17.21 (c), este é enviado através da rota por omissão e chega ao *router* NATP que reescreve de novo o cabeçalho, usando uma regra fixa, e o envia ao cliente HTTP, ver a Figura 17.21 (d).

Desta forma, as portas associadas a serviços bem conhecidos podem ser associadas a servidores na rede privada que passam a receber os pacotes vindos do exterior e dirigidos às mesmas. Este mecanismo tem as vantagens indicadas e serve para servidores de serviços bem conhecidos, mas não pode ser usado para que um qualquer computador na rede privada possa receber conexões do exterior pois exige afectar portas a computadores de forma fixa.

O outro inconveniente da tradução de endereços tem a ver com o facto de que estes mecanismos não são transparentes para protocolos aplicacionais que necessitam de passar endereços privados no interior do *payload* dos pacotes. Esses endereços, ao serem recebidos pelos destinatários, tornam-se inúteis pois não estão associados a computadores acessíveis. Alguns dos protocolos populares que ficam assim inutilizados passaram a ser reconhecidos pelo software de alguns equipamentos de NAT mas, como é evidente, esta solução é frágil, particular e pouco flexível.

O NAT tornou-se tão popular em ambiente IPv4 que foram desenvolvidas e normalizadas várias soluções que permitem a computadores em redes públicas tomarem a iniciativa de comunicarem com outros em redes privadas, ou até permitir a comunicação quando ambos os computadores estão em redes privadas. Estes mecanismos designam-se genericamente por mecanismos de NAT *traversal*.

Essas soluções passam por dois tipos de mecanismos. Por um lado, um computador

tem de ser capaz de reconhecer se está ou não por detrás de um dispositivo de NAT. À primeira vista isso parece simples pois os endereços privados estão recenseados. No entanto, é mais geral arranjar um protocolo que permite a um cliente saber qual o endereço IP origem com que um servidor público recebe os seus pacotes. Se for diferente do seu, é porque está por detrás de NAT. Isto torna-se imprescindível quando o NAT está, por qualquer motivo, escondido do computador.

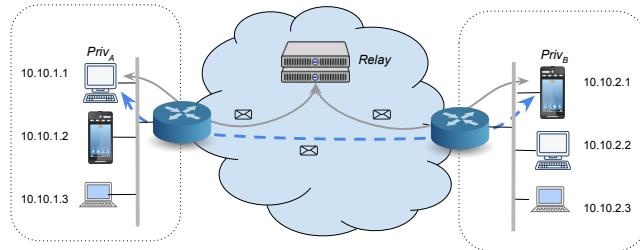


Figura 17.22: Comunicação entre dois dispositivos IP em redes privadas usando um servidor de *relaying*. A cheio o trajecto dos pacotes reais, a tracejado o trajecto lógico dos pacotes.

O outro mecanismo necessário é baseado num servidor de *relaying*, designado de agora em diante por *Relay*, ver a Figura 17.22. O computador na rede privada (*Priv<sub>A</sub>*) inicia uma transação com *Relay* de forma a que este tenha um caminho aberto no dispositivo NAT para lhe chegar, *i.e.*, uma entrada na tabela de NATP *translations* associada ao seu endereço e com uma porta dedicada. Esta via entre *Priv<sub>A</sub>* e *Relay* tem de ser periodicamente refrescada por iniciativa de *Priv<sub>A</sub>* para se manter activa. O outro computador (*Priv<sub>B</sub>*), fora da rede privada ou mesmo noutra rede privada, executa o mesmo protocolo com *Relay*. Finalmente, os dois computadores podem trocar pacotes via *Relay*, ver a Figura 17.22, que assegura uma via privada entre os dois e representa para cada um o outro interlocutor.

O mundo do NAT e do NAT *traversal* transformou-se num labirinto de soluções particulares pois as variantes dos controlos exercidos sobre a origem dos pacotes recebidos pelo dispositivo de NAT, e a necessidade de tradução de endereços nos pacotes e respectivo *payload* requerem inúmeras soluções particulares. O que é um claro indício de engenharia de má qualidade. Por outro lado, as soluções de NAT *traversal* são inúmeras e também dependem da solução NAT particular e do tipo das aplicações envolvidas. As soluções proprietárias para casos particulares são numerosas.

O IETF fez algum esforço na tentativa de normalizar soluções de NAT *traversal*, ver o RFC 5766.

Este estado de coisas teve origem no facto de que o NAT sempre foi visto como um subterfúgio para adiar a adopção de IPv6 por uma parte da comunidade Internet, o que fez com que o mesmo não fosse analisado, tratado e normalizado de forma sólida desde o início. Ao mesmo tempo a vida impôs a utilização maciça de NAT, incluindo pelos operadores com o chamado CGN (*Carrier Grade NAT*).

A generalização da utilização de IPv6 tornará a utilização de NAT inútil para vencer as restrições sobre a escassez de endereços. Quanto às outras vantagens indirectas do NAT, nomeadamente facilitar de forma simples a independência dos operadores sem aumentar o número de prefixos *provider independent* visíveis na Internet, e fornecer um primeiro mecanismo de isolamento e segurança, o mesmo já não é tão claro.

Todos os equipamentos com suporte de IP usam um conjunto de mecanismos obrigatórios e protocolos complementares (ARP, DHCP, ICMP, etc.) que permitem ao encaminho de pacotes IP funcionar na prática. Esses mecanismos têm no seu centro, como ideias construtoras, as noções de que a cada canal está associado um prefixo IP distinto, a ideia de que todas as interfaces IP para um canal têm um endereço IP no prefixo do canal, e a noção de tabela de encaminhamento como suporte do algoritmo *longest prefix match is the best*.

A tabela de encaminhamento de um computador, mesmo quando ligado a um só canal, discrimina as formas elementares do encaminhamento IP, nomeadamente: encaminhamento local, encaminhamento directo, encaminhamento indirecto e encaminhamento por omissão (por *default*). Estes mecanismos são suficientes para a ligação dos sistemas terminais e de redes simples ao mundo IP.

## 17.5 Resumo e referências

### Resumo

Devido à escala da Internet, onde existem biliões de interfaces globalmente endereçáveis, e a necessidade de tornar o encaminhamento mais simples, os endereços IP são de carácter topológico e hierárquico, são afectados por prefixes relacionados com a localização das diversas redes constituintes da Internet e são depois geridos e afectados pelos gestores das mesmas.

Os diferentes prefixes IP existentes têm entre si relações hierárquicas. Todos os prefixes são sub-prefixos do universo, denotado  $0.0.0.0/0$ , o prefixo de comprimento nulo. Diferentes sub-prefixos são depois afectados às redes que formam a Internet e sub-prefixos desses são depois afectados a redes subordinadas às mesmas.

Esta hierarquia não é estrita mas desempenha um papel fundamental no encaminhamento pois o encaminhamento no mundo IP consiste em encaminhar os pacotes para os prefixes a que pertencem. É esta característica que está por detrás da escalabilidade do encaminhamento na Internet pois as tabelas de encaminhamento (FIBs) contém prefixes IP e não endereços IP isolados.

O encaminhamento na Internet global, a chamada *Default-Free Zone*, é realizado com base nos prefixes IP anunciados pelas diferentes redes interligadas. Para evitar que uma hierarquização total dos prefixes redundasse num sistema demasiado rígido, e potencialmente com as características de um monopólio dos operadores Tier1, são anunciados prefixes correspondentes a todas as redes, mesmo de clientes finais, desde que estas tenham a necessidade de controlar de forma mais fina o encaminhamento dos pacotes que lhes são dirigidos.

Os prefixes afectados pelos operadores aos seus clientes fazem parte do espaço *provider dependent address space*. Esses clientes, se não forem *multi-homed*, estão num espaço de endereçamento que também se pode chamar *provider-based aggregation address space* porque poupa entradas nas tabelas de encaminhamento. Ao contrário, os prefixes das redes clientes mas independentes do fornecedor, têm prefixes no *provider independent address space* ou *without aggregation address space* e expandem necessariamente as tabelas de encaminhamento.

A regra *longest prefix matching is the best* na escolha das entradas das tabelas de encaminhamento dos comutadores funciona como garante de flexibilidade. O controlo fino do encaminhamento para uma rede, e as necessidades de *multi-homing* levam à subida incessante do número de prefixes IP visíveis no coração da Internet.

O protocolo IP é a “cola” ou “camada unificadora” da Internet pois permite que todos os equipamentos que o suportam possam trocar pacotes de dados entre si. O

cabeçalho IP na versão 4, a versão mais comum actualmente, para além dos endereços IP origem e destino do pacote, contém um conjunto de campos que apoiam o encaminhamento dos mesmos, nomeadamente nas facetas relacionadas com a prevenção de ciclos de encaminhamento (TTL) e qualidade de serviço.

Devido à falta de experiência e às características tecnológicas das redes quando o protocolo foi definido, o cabeçalho IPv4 inclui igualmente o suporte de mecanismos que se vieram a revelar pouco amadurecidos, como por exemplo a fragmentação, e o facto de o cabeçalho ser de comprimento variável devido à presença no mesmo de várias opções (*e.g., source routing, record route, etc.*). Na prática, as opções IPv4 pouco ou nada são suportadas e a fragmentação não é recomendada.

A versão 6 do protocolo IP introduz uma alteração fundamental ao nível da dimensão e formato dos endereços IP, e suprimiu alguns mecanismos que se revelaram inúteis ou contraproducentes em IPv4. Adicionalmente, substituiu as opções por uma eventual pilha de cabeçalhos de extensão, sempre que são necessárias funcionalidades opcionais semelhantes às opções do IPv4, ou ainda novas opções de segurança, *jumbograms, etc.* Essas opções são fundamentalmente *end-to-end* e pouco utilizadas, tal como em IPv4.

Todos os equipamentos com suporte de IP usam um conjunto de mecanismos obrigatórios e protocolos complementares (ARP, DHCP, ICMP, *etc.*) que permitem ao encaminho de pacotes IP funcionar na prática. Esses mecanismos têm no seu centro, como ideias construtoras, as noções de que a cada canal está associado um prefixo IP distinto, a ideia de que todas as interfaces IP para um canal têm um endereço IP no prefixo do canal, e a noção de tabela de encaminhamento como suporte do algoritmo *longest prefix match is the best*.

A tabela de encaminhamento de um computador, mesmo quando ligado a um só canal, discrimina as formas elementares do encaminhamento IP, nomeadamente: encaminhamento local, encaminhamento directo, encaminhamento indirecto e encaminhamento por omissão (por *default*). Estes mecanismos são suficientes para a ligação dos sistemas terminais e de redes simples ao mundo IP.

Alguns dos termos introduzidos no capítulo são a seguir passados em revista. Entre parêntesis figura a tradução mais comum em língua inglesa.

### Principais conceitos

**Endereço IP** (*IP address*) Endereços afectados às interfaces dos dispositivos com suporte do protocolo IP que se caracterizam por serem afectados num espaço com carácter topológico e hierárquico.

**Quad notation** Notação usada para representar os endereços IP versão 4 usando 4 grupos de dígitos decimais separados por um ponto (Exemplo: 193.136.126.43).

**Hexadecimal colon notation** Notação usada para representar os endereços IP versão 6 usando 8 grupos de 4 dígitos hexadecimais separados por dois pontos (Exemplo: 2001:0:0:0:0:45bf:ff23:8b4f).

**Prefixo IP** (*IP prefix*) Conjunto de endereços IP contíguos que partilham entre si um conjunto de bits mais significativos. São representados pelo endereço IP correspondente aos bits do prefixo, seguidos de bits a zero, seguido de uma barra e do número do comprimento do prefixo (Exemplos: 193.136.126.0/24 e 2001:690::/29).

**O prefixo mais longo é o melhor** (*longest prefix matching is the best*) As tabelas de encaminhamento IP são indexadas por prefixes IP. Um endereço IP pode estar contido em mais do que um dos prefixes existentes na tabela. Se for o caso, o prefixo mais longo, *i.e.*, com mais bits, toma precedência na escolha.

**Campos do cabeçalho IP** (*IP header fields*) Os campos do cabeçalho dos pacotes IP condicionam a forma como estes são processados e encaminhados entre a

origem e o destino. Entre os principais encontram-se os endereços origem e destino, assim como o campo que impede que um pacote possa permanecer eternamente num ciclo de encaminhamento.

**Encaminhamento directo** (*direct routing*) Forma de encaminhamento de um pacote por um dispositivo IP  $r$  em que o destino do pacote é outro dispositivo IP ligado directamente a  $r$  por um canal.

**Encaminhamento indirecto** (*indirect routing*) Forma de encaminhamento de um pacote por um dispositivo IP  $r$  em que o destino do pacote se encontra por detrás de um *router* IP directamente ligado a  $r$  por um canal.

**Encaminhamento por omissão** (*default routing*) Forma de encaminhamento indirecto associada ao prefixo IP  $0.0.0.0/0$ , também conhecido por rota por omissão (*default route*), que funciona como rota de último recurso quando mais nenhuma outra se aplica a um pacote.

**Default-Free Zone (DFZ)** A região da Internet em que os *routers* não usam rotas por omissão chama-se DFZ pois as suas tabelas de encaminhamento devem, em princípio, conter todos os prefixes IP válidos na Internet pública. Se o endereço de destino de um pacote não está contido em nenhum desses prefixes, então um *router* da DFZ não consegue encaminhá-lo. Tal só deve acontecer com endereços de prefixes não atribuídos ou inválidos.

**Address Resolution Protocol** (ARP) Protocolo baseado em difusão que permite a um dispositivo IP  $r$  obter o endereço MAC de outro dispositivo IP directamente ligado a  $r$  via um canal e de que apenas conhece o endereço IP. Este protocolo não existe em IPv6.

**Internet Control Message Protocol** (ICMP) Protocolo que permite enviar notificações de erro na sequência do tratamento de um pacote para o emissor inicial do mesmo. O protocolo também permite obter meta-informação sobre a rede. Na versão do ICMPv6 estão incluídas as funcionalidades do ARP para IPv6.

**Dynamic Host Configuration Protocol** (DHCP) Protocolo que permite a um sistema terminal obter a informação necessária para a sua parametrização (e.g., endereço, prefixo, endereços de servidores DNS, etc.).

**Network Address Translation with Ports** (NATP ou simplesmente NAT) Mecanismo que permite a um conjunto de dispositivos de uma rede, partilharem um único endereço quando necessitam de comunicar com o exterior dessa rede. Este mecanismo só permite que a iniciativa da comunicação tenha por origem sistemas pertencentes ao reduto “privado”.

**Port forwarding** Mecanismo complementar do NAT que permite que um sistema no reduto “privado” receber conexões do exterior usando uma porta fixa.

**NAT Traversal** Mecanismos complementares do NAT que permitem que um computador no exterior do reduto “privado” tome a iniciativa de comunicar com um dispositivo arbitrário no interior do reduto “privado”.

## Referências

O problema do crescimento da dimensão das tabelas de encaminhamento da DFZ tem sido uma preocupação constante pois se os *routers* não fossem capazes de lidar com a dimensão dessa tabela, ou de processarem os pacotes em tempo útil, seria necessário diminuir o número de prefixes visíveis. Este problema foi objecto de um *workshop* cujas conclusões, relatadas no RFC 4984, influenciaram uma investigação intensa quer na forma como conseguir melhorar os *routers*, quer na pesquisa de arquitecturas de encaminhamento que obvissem esses problemas, mantendo a liberdade dos clientes e dos ISPs.

O aumento do desempenho da pesquisa em tabelas de encaminhamento tem sido dominada pelo desenvolvimento de novos algoritmos [Ruiz-Sanchez et al., 2001] e implementações em hardware. No entanto, têm sido também conseguidos progressos em termos de implementações em software, como por exemplo as relatadas em [Dobrescu et al., 2009; Zec et al., 2012] que permitem manter algum optimismo no que respeita à possibilidade de se utilizarem tabelas de encaminhamento cada vez maiores.

Existem vários livros sobre IPv6. Os seguintes são dos mais reputados [Graziani, 2013; Hagen, 2014]. O artigo [Czyz et al., 2014] constitui uma análise detalhada sobre a adopção de IPv6 por volta do ano de 2014.

WebRTC[Jennings et al., 2013] é um *framework* JavaScript para permitir a comunicação directa entre *browsers* Web mesmo quando os interlocutores estão ambos em redes privadas. O *framework* é um exemplo importante de um contexto em que a problemática NAT *traversal* tem muita relevância e requer APIs e soluções normalizadas como: *Interactive Connectivity Establishment* (ICE), ver os RFCs 4091 e 4092, *Traversal Using Relays around NAT* (TURN) e *Relay Extensions to Session Traversal Utilities for NAT* (STUN), ver o RFC 5766.

#### **Apontadores para informação na Web**

- <http://ietf.org/rfc.html> – É o repositório oficial dos RFCs, nomeadamente dos citados neste capítulo.
- <http://www.icann.org> – É o *site* oficial da ICANN, responsável pela afectação de endereços IP.
- <http://www.internetsociety.org/deploy360/ipv6> – É o *site* oficial da Internet Society sobre o IPv6.
- <http://bgp.potaroo.net> – É um *site* com informação sobre os prefixos IP presentes nas tabelas de encaminhamento globais da Internet, com especial realce para um conjunto de relatórios sobre a sua evolução, incluindo apontadores para outros relatórios como o indicado a seguir.
- <http://www.cidr-report.org/> – É um *site* que contém um relatório elaborado semanalmente, assim como uma análise da evolução histórica, do estado da tabela de encaminhamento da *Default Free Zone* da Internet em IPv4 e IPv6.
- <https://www.vyncke.org/ipv6status/> – É um *site* mantido por Eric Vyncke que apresenta relatórios sobre a evolução da utilização de IPv6.

### **17.6 Questões para revisão e estudo**

1. Verdade ou mentira?
  - (a) O cabeçalho dos pacotes IP contém um campo designado TTL que permite a todo o momento saber quanto tempo passou desde que o pacote foi emitido.
  - (b) O cabeçalho dos pacotes IP contém um campo que assegura que se um pacote entrar num ciclo de encaminhamento dentro da rede, acabará por ser destruído.
  - (c) Os *routers* no interior da rede estão todos sob a mesma autoridade administrativa pois caso contrário seria impossível garantir que os pacotes chegam ao destino.
  - (d) O cabeçalho de um pacote IP contém um campo onde é registada a lista dos *routers* que este atravessou para que possa ser detectado se o pacote entrou em ciclo e deve ser destruído.

2. Verdade ou mentira?
  - (a) Os endereços do nível MAC são hierárquicos e reflectem informação sobre a topologia da rede, *i.e.*, informação sobre a região da rede em que a interface que tem o endereço se situa.
  - (b) Os endereços IP apesar de poderem ser utilizados numa rede com biliões de computadores têm uma estrutura interna que permite evitar que as tabelas de encaminhamento sejam da mesma dimensão.
  - (c) As interfaces Ethernet têm um endereço MAC fixo, potencialmente eterno. O ideal ao nível rede da Internet actual seria os utilizadores poderem ter endereços IP públicos fixos com a mesma propriedade e poderem-nos usar independentemente do ISP a que estivessem ligados fosse qual fosse o país ou região para que se deslocassem. Tal opção não é implementada apenas por conservadorismo da rede Internet.
  - (d) A razão pela qual os ISP não aceitam que os utilizadores Internet mantêm sempre o mesmo endereço IP fixo e independente do ISP (portabilidade de endereços) tem exclusivamente por origem estratégias comerciais de cativação dos clientes.
3. Quantos prefixos IP /24 existem dentro do prefixo 87.103.0.0/17?
4. Quais dos seguintes endereços IP: 192.168.0.129, 192.168.0.145, 192.168.1.129, 192.168.0.1 e 192.168.0.140 estão incluídos no prefixo 192.168.0.128/28?
5. Um *router* recebe por uma das suas interfaces um *frame* Ethernet contendo um pacote IP. Para o processar, o *router* executa várias operações, algumas das quais estão indicadas a seguir, mas as mesmas estão desordenadas. Coloque-as por ordem.
  - (a) se não há entrada na tabela de encaminhamento para o endereço de destino, descarta o pacote, e termina o processamento.
  - (b) processa o *frame* Ethernet, retirando o pacote e colocando-o numa variável.
  - (c) se descartou o pacote devido a um erro, envia um pacote ICMP para o emissor original e termina o processamento.
  - (d) encapsula pacote num *frame*, envia-o para *next-hop* encapsulado no *frame* adequado e termina o processamento.
  - (e) calcula o novo *checksum* do cabeçalho do pacote.
  - (f) decrementa o TTL de pacote e se este ficar com o valor 0 descarta-o.
6. Um pacote IPv4 com um cabeçalho IP com 20 bytes foi recebido e enviado de novo por um dado *router* que não suportava opções de qualidade de serviço, nem teve necessidade de o fragmentar. Que campos do cabeçalho do pacote IP foram modificados pelo *router*? Escolha uma e uma só das opções.
  - (a) Nenhum
  - (b) Os campos TTL, Type of Service, Options
  - (c) Os campos TTL, Offset
  - (d) Os campos TTL, Checksum
  - (e) Os campos TTL, Origin Address, Options
  - (f) Os campos TTL, Destination Address
  - (g) Os campos TTL, Cheksum, Origin Address

(h) Os campos TTL, Cheksum, Destination Address

7. O IPv4 suporta um mecanismo de fragmentação de pacotes pelos *routers*. Responda às seguintes questões:
  - (a) Se um pacote IP foi fragmentado, o receptor final do pacote só consegue recompor o pacote inicial se os diferentes segmentos chegarem pela ordem com que foram produzidos? Justifique a sua resposta.
  - (b) A recomposição de um pacote necessita de usar um alarme. Porquê?
  - (c) No IPv6 resolveu-se abandonar o mecanismo da fragmentação nos *routers*. Indique o porquê e qual ou quais as alternativas que passaram a ser usadas.
8. Qual a maior dimensão possível do *payload* de um pacote IPv4 que é aconselhável enviar por uma interface Ethernet? Mesma questão com um pacote IPv6?
9. O campo **Identification** do cabeçalho dos pacotes IPv4 tem 16 bits. Assim, após o envio por um computador de cerca de 64.000 pacotes para o mesmo destino, os números de identificação dos pacotes são necessariamente reutilizados. O computador está a enviar pacotes compatíveis com a dimensão da sua interface Ethernet mas não sabe se os pacotes vão ou não ser fragmentados pelos *routers*. Por hipótese o *Maximum Packet Life Time* (MPLT) é 120 segundos. Indique qual o débito máximo a que este computador pode enviar pacotes para o mesmo destino, para ter a certeza que em caso de fragmentação nenhum pacote pode ser corrompido pelo mecanismo de recomposição dos pacotes no receptor.
10. Um comutador IP (*router*) tem uma tabela de encaminhamento (`table`) com entradas do tipo `route` (obtidas por `table.get-route(address)` ). Uma rota tem os seguintes atributos: `IP prefix`, `type` (`local`, `direct`, `indirect`), `interface` (a `interface` ou `null` se o encaminhamento for indirecto), e `next-hop` (o endereço IP do `next-hop` se o encaminhamento é indirecto ou `null` nos outros casos). Os pacotes IP são do tipo `packet` com vários atributos entre os quais: `origin` (endereço IP), `destination` (endereço IP) e `TTL` (um inteiro). Os comutador pode processar os pacotes recebidos por um dos seguintes métodos:
  - `packet.ignore()` - que destrói o pacote retirando-o dos `buffers` do comutador
  - `packet.process()` - que processa um pacote dirigido a uma das interfaces do comutador e que necessariamente verifica a condição:

```
table.get-rota(packet.destination).tipo == local
packet.send(interface, next-hop) - para envio de um pacote e em que interface é necessariamente diferente de null e next-hop é um endereço IP diretamente alcançável pelo comutador.
```

Apresente o pseudo-código de processamento pelo router de um pacote p.

11. A tabela de encaminhamento de um computador tem as entradas a seguir indicadas. No entanto, uma delas é impossível e não serve para nada. Diga qual e justifique a sua resposta.

192.10.1.1/32	local	eth0
192.10.1.0/24	directo	eth0
192.10.2.0/24	indirecto	netx hop: 192.10.1.254
192.10.3.0/24	indirecto	netx hop: 192.10.6.254

12. A tabela de encaminhamento de um computador tem as entradas a seguir indicadas.

190.30.16.1/32	local	eth0
190.30.16.0/24	directo	eth0
190.40.0.0/16	directo	eth1
190.30.45.0/24	indirecto	next hop: 190.30.16.10
190.30.48.0/24	indirecto	next hop: 190.30.32.10
0.0.0.0/0	indirecto	next hop: 190.30.16.10

- (a) Dado o endereço 190.30.16.100 qual das entradas da tabela é escolhida para decidir qual é o *next hop*?
- (b) Dado o endereço 190.30.100.34 qual das entradas da tabela é escolhida para decidir qual é o *next hop*?
13. A tabela de encaminhamento de um computador tem as seguintes entradas.
- |               |           |                        |
|---------------|-----------|------------------------|
| 192.10.1.0/24 | directo   | eth0                   |
| 192.10.2.0/24 | directo   | eth1                   |
| 192.10.3.0/24 | indirecto | next hop: 192.10.1.254 |
| 192.10.4.0/24 | indirecto | next hop: 192.10.2.254 |
| 0.0.0.0       | indirecto | next hop: 192.10.1.254 |
- (a) Uma destas entradas é redundante e pode ser suprimida. Diga qual.
- (b) Dado o endereço 192.10.3.15 qual das entradas da mesma é escolhida para decidir qual é o *next hop*?
14. Considere a seguinte tabela de encaminhamento.
- |                |           |                       |
|----------------|-----------|-----------------------|
| 190.30.16.0/20 | directo   | eth0                  |
| 190.30.32.0/20 | directo   | eth1                  |
| 190.30.0.0/20  | indirecto | next hop: 190.30.16.1 |
| 190.30.48.0/20 | indirecto | next hop: 190.30.32.1 |
| 0.0.0.0        | indirecto | next hop: 190.30.16.1 |
- (a) Dado o endereço 190.30.64.34 qual das entradas da mesma é escolhida para decidir qual é o *next hop*?
- (b) Dado o endereço 190.30.49.34 qual das entradas da mesma é escolhida para decidir qual é o *next hop*?
15. Considere a rede de um cliente da Figura 17.23 e indique o conteúdo da tabela de encaminhamento do *router* do cliente.

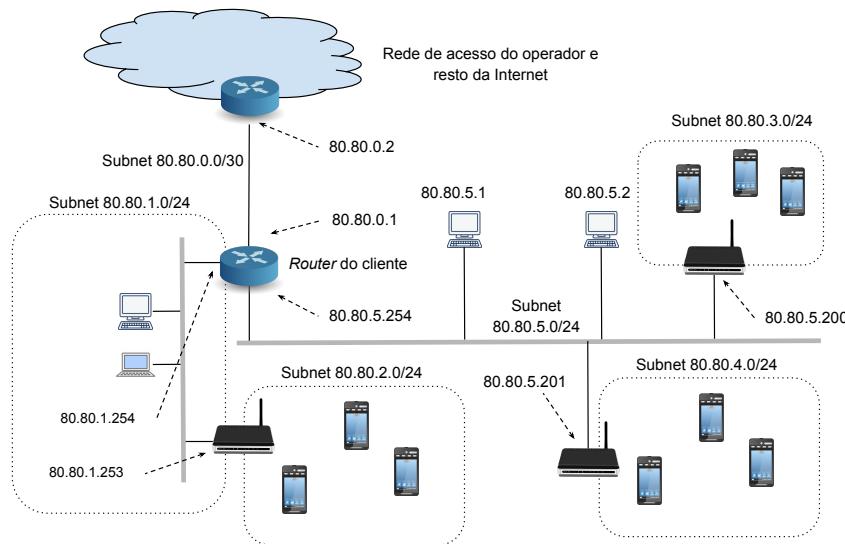


Figura 17.23: Rede de um cliente ligado à Internet através de um ISP

16. Considere a rede da Figura 17.24 e indique o conteúdo da tabela de encaminhamento do *router R1*. Indique também se o *router R2* deve ou não executar a função de NAT (*Network Address Translation*)?

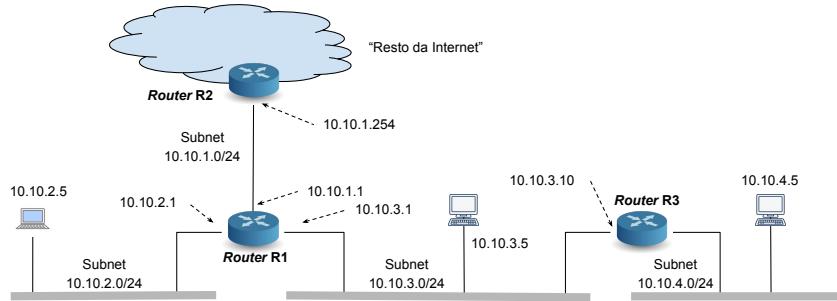


Figura 17.24: Rede ligada à Internet através de um ISP

17. Um *router r* tem duas interfaces: a interface *eth0* com IP 10.0.1.254 e *eth1* com IP 10.0.2.254. As tabelas de ARP e encaminhamento do *router r* têm o seguinte conteúdo:

10.0.1.1	23:45:A0:4F:67:CD	(arp)
10.0.1.254	23:45:AB:2F:83:AD	(arp)
10.0.2.254	23:45:AB:E3:56:D2	(arp)
10.0.1.0/24	directo eth0	
10.0.2.0/24	directo eth1	
0.0.0.0/0	indirecto via 10.0.2.254	

Indique que *frames* é que o *router* tem de enviar (tipo do *frame* e endereços MAC origem e destino) para encaminhar um pacote:

- (a) Para o endereço 10.0.2.254?
- (b) Para o endereço 10.0.2.100?
- (c) Para o endereço 10.0.3.30?
- (d) Caso a interface *eth1* receba um *frame* Ethernet do tipo ARP com a *query*: *Who has address 10.0.2.100?*, o *router* deve responder à mesma? Justifique a sua resposta.

18. Um *router r* tem duas interfaces: a interface *eth0* com IP 10.0.1.254 e *eth1* com IP 10.0.2.254. As tabelas de ARP e encaminhamento do *router r* têm o seguinte conteúdo:

10.0.1.1	23:45:A0:4F:67:CD	(arp)
10.0.1.254	23:45:AB:2F:83:AD	(arp)
10.0.2.254	23:45:AB:E3:56:D2	(arp)
10.0.1.0/24	directo eth0	
10.0.2.0/24	directo eth1	

Indique que *frames* é que o *router* tem de enviar (tipo do *frame* e endereços MAC origem e destino) para encaminhar um pacote para o endereço 10.0.3.30?

19. Um computador tem uma interface com o endereço 100.100.0.1/24 e recebe um pacote com endereço de destino 100.100.0.100. A tabela de ARP do computador tem o seguinte conteúdo:

100.100.0.1	00:00:AA:BB:00:10
100.100.0.4	00:00:AA:BB:00:02
100.100.0.10	00:00:AA:BB:00:01
100.100.0.67	00:00:AA:BB:00:55

Indique quais os *frames* Ethernet que o computador deve enviar para encaminhar o pacote até ao seu destino.

20. Descreva de que forma o programa **traceroute** utiliza o protocolo ICMP para funcionar.
21. Um servidor DHCP pretende assegurar-se, antes de afectar o endereço IP a um computador, que não existe nenhum outro computador a usar o mesmo endereço IP na sub-rede que controla. Como pode proceder?
22. Numa rede IP com suporte de difusão de *frames* Ethernet existe um servidor DHCP, o servidor DHCP1, que afeta endereços IP no prefixo 100.100.100.0/24 aos computadores que se ligam a essa rede, com um *lease time* de 1000 segundos. O servidor DHCP1 avariou-se e foi substituído imediatamente pelo servidor DHCP2 que não conhece, no momento do arranque, quais os endereços afetados pelo servidor DHCP1 que ainda estão a ser usados.  
DHCP2 afeta endereços no mesmo prefixo que DHCP1 e tem agora o problema de quando recebe um novo pedido de endereço, decidir que endereço afetar. DHCP2 poderia correr o risco de afetar endereços aleatoriamente na esperança de que não houvessem colisões com endereços afetados por DHCP1 que ainda estivessem a ser usados. Poderia também pedir a todos os computadores que devolvessem os endereços em uso e pedisse um novo, mas o protocolo DHCP não suporta essa hipótese.  
Que pode fazer DHCP2 para diminuir a probabilidade de afetar endereços que ainda estejam a ser usados por computadores ligados à sua rede? Justifique a sua resposta.
23. Um computador recebeu um endereço IP de um servidor DHCP (único nessa rede) mas pouco depois esse servidor foi aberto antes da *lease* do cliente expirar. O servidor DHCP não recuperou da avaria senão 24 horas depois. Quais das seguintes afirmações são de certeza verdadeiras?
  - (a) O computador perdeu a conectividade com a Internet porque o servidor DHCP deixou de encaminhar os seus pacotes para a Internet.
  - (b) O computador percebeu-se da ausência do servidor DHCP quando a *lease* expirou.
  - (c) Outros computadores deixaram de conseguir obter endereços IP na mesma rede.
  - (d) O computador deixou de conseguir aceder a *sites* WEB porque deixou de resolver *queries* DNS.
  - (e) O computador deixou de ter actualizações da sua tabela de encaminhamento.
24. Os RFCs relacionados com o protocolo DHCP recomendam que um servidor DHCP envie uma mensagem ARP sobre o endereço que vai afectar antes de fazer uma *offer* do mesmo. Quais das seguintes afirmações podem justificar esta recomendação?
  - (a) O servidor DHCP não sabe que endereços IP afetou antes de um seu anterior *crash*.
  - (b) O utilizador de algum computador pode utilizar um endereço livre mesmo sem o mesmo ter sido afectado pelo servidor DHCP.

- (c) O servidor DHCP foi reinicializado.
- (d) De forma a que os *switches* dessa *subnet* passem a conhecer o endereço MAC do servidor DHCP, e optimizem a sua localização pelos clientes.
25. Um *router* recebeu um pacote IPv4 com endereço origem 130.45.3.3 e endereço de destino 201.23.4.6. Não existe nenhum prefixo de rede na tabela de encaminhamento que contenha esse endereço e não há nenhuma rota por omissão (*default route*). Como deve o *router* tratar esta situação? (Só uma das opções abaixo é válida)
- (a) Devolver o pacote ao endereço 130.45.3.3
  - (b) Destruir o pacote e avisar por ICMP o *router* anterior
  - (c) Destruir o pacote e enviar uma mensagem ICMP do tipo “Host Unreachable” para o endereço 201.23.4.6
  - (d) Avisar o computador no endereço 130.45.3.3 de que não sabe encaminhar o pacote
  - (e) Destruir o pacote e enviar uma mensagem ICMP do tipo “Host Unreachable” para o endereço 201.23.4.6
  - (f) Destruir o pacote e enviar uma mensagem ICMP do tipo “Host Unreachable” para o endereço 130.45.3.3
26. Descreva como o protocolo ICMP pode ser usado para descobrir qual o valor do MTU que deve ser usado para o envio de pacotes sem fragmentação entre dois computadores.
27. NAT neste contexto designa *Network Address Translation with Ports*. Qual ou quais das seguintes afirmações são verdadeiras?
- (a) O NAT permite que um único endereço IP privado seja partilhado por vários computadores da rede interna.
  - (b) O NAT permite que um único endereço IP público seja partilhado por vários computadores da rede interna.
  - (c) A técnica de NAT permite que o TCP funcione mais rapidamente pois faz *caching* de conexões TCP e permite que o estabelecimento da conexão (o *three way hand shaking*) funcione mais depressa.
28. O computador *A* tem uma ligação TCP a um servidor *S* pelo que lhe envia pacotes contendo segmentos TCP. *A* e *S* estão ligados através de vários comutadores e diversos canais Ethernet. *A* tem o endereço 10.0.1.12 no prefixo 10.0.1.0/24 e *S* tem o endereço 193.136.126.43 no prefixo 193.136.126.0/24. Quer na emissão, quer na receção, as mensagens trocadas entre *A* e *S* têm três cabeçalhos (cabeçalho do *frame* Ethernet, cabeçalho IP e cabeçalho TCP). Durante o transporte de uma dessas mensagens de *A* para *S* alguns cabeçalhos podem eventualmente ser alterados. Qual ou quais das seguintes afirmações são verdadeiras ?
- (a) Os três cabeçalhos da mensagem recebida por *S* contendo um segmento TCP são os mesmos que os da mensagem transmitida originalmente por *A*.
  - (b) Os cabeçalhos do *frame* Ethernet e o cabeçalho IP da mensagem recebida por *S* são exatamente os mesmos que os da mensagem transmitida por *A*, mas o cabeçalho TCP é diferente
  - (c) Os cabeçalhos do *frame* Ethernet e o cabeçalho IP da mensagem recebida por *S* são diferentes dos da mensagem transmitida por *A*, mas o cabeçalho TCP é igual.

- (d) O endereço IP origem dos pacotes recebidos pelo servidor é o endereço do computador  $A$ .
- (e) O cabeçalho do *frame* Ethernet da mensagem recebida por  $S$  é exatamente o mesmo que o da mensagem transmitida por  $A$ , mas os cabeçalhos IP e TCP são diferentes.
- (f) Os três cabeçalhos da mensagem recebida por  $S$  contendo um segmento TCP são todos diferentes dos da mensagem transmitida originalmente por  $A$ .
- (g) O endereço MAC origem dos *frames* Ethernet recebidos pelo servidor  $S$  é o endereço MAC da interface externa do *router* que liga a sua rede do computador  $A$  ao seu fornecedor de serviços Internet.
29. O computador  $A$ , numa rede local de uma universidade, está a enviar segmentos TCP para um computador  $S$ , numa rede local de uma universidade de outra cidade.  $A$  tem um endereço privado e  $S$  um endereço público. Diga qual ou quais dos seguintes campos dos cabeçalhos IP e TCP dos segmentos são alterados entre a emissão por  $A$  e a recepção por  $S$ :
- (a) Números de sequência do segmento TCP
  - (b) *advertised window size* do segmento TCP
  - (c) Endereço IP origem do pacote IP
  - (d) Endereço IP destino do pacote IP
  - (e) Número de porta origem do segmento TCP
  - (f) Número de porta destino do segmento TCP
  - (g) Campo TTL do pacote IP
  - (h) Campo qualidade de serviço do pacote IP
  - (i) Campo *checksum* do pacote IP
  - (j) Campo *checksum* do pacote TCP
30. Um *router* está a desempenhar igualmente funções de NAT (*Network Address Translation with Ports*). Quando se abre uma nova conexão TCP, ou quando passa um primeiro segmento UDP (com endereço origem, destino, porta origem ou porta destino diferentes) é criada uma nova associação na tabela de associações NAT. O problema reside em decidir quando é que a mesma associação é declarada “morta” e pode ser removida. Em TCP é possível detectar o fecho da conexão, mas em UDP essa noção não existe. Por outro lado, em TCP se uma das partes “morre”, a conexão não pode ser fechada regularmente. Como arranjar uma forma adequada de lidar com estes dois problemas?
31. Um *router* está a desempenhar igualmente funções de NAT (*Network Address Translation with Ports*). Associado aos pacotes ICMP não existem portas. Será mesmo assim possível encontrar uma forma adequada de lidar com NAT *traversal* de pacotes ICMP?



## *Capítulo 18*

---

### *Encaminhamento na Internet global*

---

*"In theory, there is no difference between theory and practice. But in practice, there is."*

– Autor: *Yogi Berra*

A Internet é uma rede de redes IP ou um *internetwork* IP, i.e., uma interligação de redes com interfaces IP. Entre essas redes interligadas encontramos redes domésticas, redes institucionais, redes de centros de dados, redes de acesso e redes de trânsito. Com excepção das redes domésticas e institucionais, todas as redes são operadas por operadores especializados em fornecerem serviços de rede, cuja actividade é desempenhada normalmente no quadro de uma empresa com fins lucrativos.

Os mecanismos que foram introduzidos no Capítulo 17 são suficientes para o funcionamento das redes domésticas e das redes institucionais de pequena dimensão. Em muitas redes IP de média dimensão, os protocolos apresentados no Capítulo 16 são suficientes para a parametrização e gestão do encaminhamento interno às mesmas.

No entanto, a interligação de redes de grande dimensão, em particular de redes dos operadores, tem requisitos especiais. A este nível a escala é muito significativa e tem crescido sem cessar. Por outro lado, os critérios de escolha das rotas não dependem apenas de constrangimentos de desempenho, mas dependem igualmente de constrangimentos relacionados com relações comerciais e políticas entre operadores. Para essa interligação de redes IP existe um único protocolo normalizado e omnipresente que tenta responder a esses requisitos especiais, trata-se do protocolo BGP (*Border Gateway Protocol*). A discussão desses requisitos e deste protocolo é o objectivo deste capítulo.

#### **18.1 Os problemas da escala**

Quando uma rede tem uma dimensão da ordem de grandeza dos milhares ou mesmo milhões de *routers*, os protocolos de encaminhamento pelo melhor caminho apresentados no Capítulo 16 têm problemas em suportarem esta escala. Com efeito, numa rede com essa dimensão, o número de actualizações a propagar por unidade de tempo é certamente muito grande, pois em cada momento há sempre algum evento num *router* com repercuções na maioria dos outros *routers*, por exemplo quando uma sub-rede

deixa de ser acessível. Por isso os protocolos OSPF e IS-IS, ver o Capítulo 16, suportam a partição da rede em várias zonas: uma zona chamada *backbone*, e outras subordinadas, interligadas pela primeira, ver a Figura 18.1.

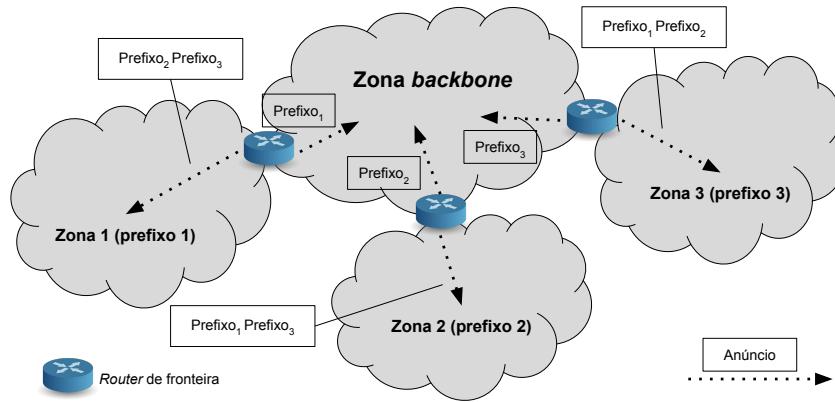


Figura 18.1: Organização de uma rede OSPF em várias zonas.

Com esta organização hierárquica as alterações de estado dos canais apenas são propagadas por inundação para os *routers* da zona onde a alteração teve lugar, e cada uma das zonas só conhece os detalhes da sua rede interna.

Compete aos *routers* que fazem a ligação de cada zona ao *backbone* fazerem a interface em termos de tráfego e de encaminhamento: o *backbone* deve conhecer os prefixos internos a cada zona e assegurar a ligação entre as mesmas, e cada zona deve conhecer todos os prefixos conhecidos pelo *backbone*, i.e., os prefixos das outras zonas. Estes *routers*, chamados de *fronteira*, isolam cada zona dos detalhes sobre o estado das outras zonas, mas propagam os prefixos conhecidos de cada lado da fronteira. Uma forma de aumentar drasticamente a escalabilidade desta estrutura consiste em organizar o endereçamento de tal forma que cada zona possa ser conhecida pelas outras apenas por um único prefixo IP, o qual tem como sub-prefixos todos os prefixos internos da zona.

Com uma gestão adequada dos prefixos e dos anúncios, também é possível fazer uma organização hierárquica do mesmo tipo com base em protocolos do tipo vector de distâncias.

A organização da rede em zonas permite aumentar a sua dimensão sem que os custos do seu *control plane* se tornem insuportáveis, quer do ponto de vista do tráfego de propagação das actualizações do estado da rede, quer do ponto de vista dos recursos usados nos *routers* para os processar. Em contrapartida, ela cria vários constrangimentos pois impõe uma organização estreita da rede que não se compadece com a evolução dos requisitos, da dimensão relativa das zonas etc. Ao fim de algum tempo torna-se necessário reorganizar as zonas e o endereçamento. Ora quanto maior for a rede, maior será o custo destas reorganizações.

No início da Internet, quando esta não passava de um projecto académico e científico, a estrutura adoptada era a de uma rede organizada como uma árvore em que a raiz era o *backbone* da NFSNET, i.e., o *backbone* subsidiado pela National Science Foundation dos EUA. Este *backbone* assegurava a interligação das redes regionais, as quais asseguravam a interligação das redes locais, ver a Figura 18.2.

Com a comercialização da Internet, ocorrida durante a primeira metade dos anos 90, apareceram vários grandes *backbones* comerciais, que se interligaram entre si e forneciam serviços às redes regionais e mesmo a algumas locais. O dinamismo da

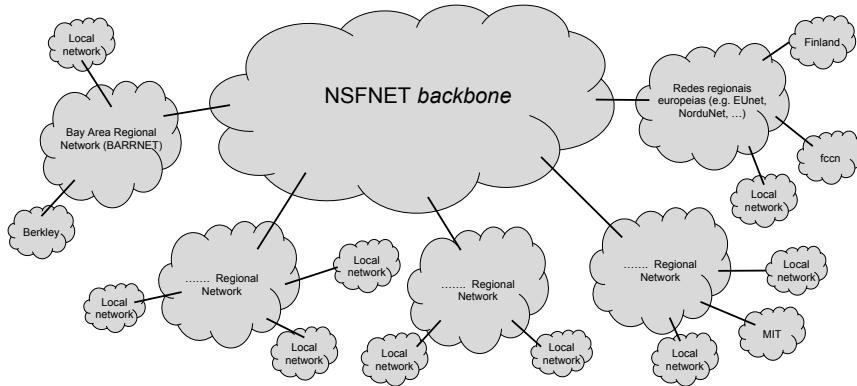


Figura 18.2: Estrutura da Internet pré-comercialização com exemplos de redes regionais e locais

Internet em termos de operadores a entrarem e a saírem do mercado, a evolução da dimensão relativa das diferentes redes nas diferentes regiões do mundo, a generalização da interligação directa de redes regionais e locais, *etc.* deixou de ser compatível com uma organização estritamente hierárquica e a hierarquização (mais ou menos estrita) dos prefixos IP.

Dada a dimensão gigantesca desta rede de redes, também não é realista usar os protocolos tradicionais para gerir o encaminhamento do conjunto dos milhões de *routers* existentes no seu conjunto. Os protocolos do tipo *link-state* não escalariam até à dimensão de um potencial *backbone* da Internet. Por outro lado, dada a heterogeneidade das diferentes redes interligadas, em dimensão, responsabilidade, forma de gestão, métricas do custo do encaminhamento, *etc.* é indesejável impor-lhes um funcionamento e práticas de gestão uniformizadas. Essa heterogeneidade requer descentralização e isolamento da gestão, das soluções técnicas adoptadas e também das métricas e outras técnicas usadas para gerir cada sub-rede da Internet.

Para responder a estes desafios foi necessário introduzir novos conceitos de organização do encaminhamento, nomeadamente a noção de sistema autónomo, e protocolos capazes de lidarem com a nova situação, nomeadamente o protocolo BGP.

## 18.2 Sistemas autónomos

Um **Sistema Autónomo (*Autonomous System*)**, que abreviaremos daqui para a frente por AS, é uma rede ou conjunto de redes sob a mesma administração e cujo encaminhamento interno é opaco para o exterior. A forma como o encaminhamento interno ao AS é realizado é um problema do AS que este deve resolver como achar melhor, usando os protocolos e métricas de custo que preferir. A fronteira dos ASs é constituída por *routers* especiais, os *routers de fronteira ou border routers*, que asseguram a ligação com os outros ASs.

Assim, o encaminhamento na Internet está estruturado a dois níveis: **Intra-AS**, o qual é da responsabilidade de cada AS, e **Inter-AS** que tem de ser uniformizado e normalizado. A Internet global é formada pela interligação entre si dos diferentes ASs, via o encaminhamento inter-AS que é suportado por um único protocolo de encaminhamento, o protocolo BGP ou *Border Gateway Protocol*, executado pelos *border*

*routers*<sup>1</sup>. Naturalmente, a interface de troca real de pacotes entre ASs tem de ser compatível com o protocolo IP.

Os protocolos de encaminhamento são assim classificados em protocolos IGP (*Internal Gateway Protocols*) e protocolos EGP (*External Gateway Protocols*), que se resumem actualmente ao protocolo BGP.

### Encaminhamento intra-AS

O encaminhamento dentro de um AS é assegurado usando encaminhamento estático e dinâmico. Neste último caso os protocolos mais populares são os protocolos RIP, EIGP, OSPF, IS-IS, *etc.* que são protocolos que se baseiam nos princípios apresentados no Capítulo 16. Nas redes dos operadores de grande dimensão, os chamados *carriers*, são usadas outras soluções mais complexas. Em alguns centros de dados também são usadas soluções especiais.

Os protocolos de encaminhamento usados constituem o chamado **control plane do AS**. Este tem por papel fazer chegar aos intra-*routers* a informação que eles usam para manterem as suas tabelas de encaminhamento (ou *Forward Information Bases* (FIBs)) actualizadas com informação da mesma natureza que as tabelas de encaminhamento IP apresentadas na Capítulo anterior, ver o Capítulo 17. Assim, a informação de acessibilidade que estes protocolos transmitem entre *routers* é sempre em termos de prefixos IP e custos. As métricas de custos desses protocolos são selecionadas de molde a atingirem-se os objectivos de gestão pretendidos pelos gestores do AS. A variedade de critérios e a diversidade das redes determina a liberdade de escolha das métricas e dos protocolos intra-AS. Seria impossível impor métricas uniformes e universais adequadas a todos os contextos pois os mesmos são muito diferentes uns dos outros.

O AS de um operador inclui também as redes dos seus clientes cujos prefixos estão escondidos dentro dos seus próprios prefixos, pois só a ele estão ligados e não têm AS próprio. Neste caso, os protocolos de encaminhamento usados entre os clientes e o operador também são escolhidos com bastante liberdade. Caso uma dessas redes tenha um único canal de ligação ao operador, uma simples rota por omissão (*default route*) é suficiente para que o caminho para todos os destinos da Internet seja conhecido em toda a rede do cliente.

Cada AS existente na Internet tem um número único a nível mundial que, tal como os prefixos IP, é afectado pela IANA via os RIRs. Os números de AS não têm significado geográfico e são globais. Inicialmente os números de AS foram afectados num espaço com 16 bits, mas actualmente são afectados num espaço com 32 bits pois o seu número tem crescido continuamente.

Um **Sistema Autónomo (Autonomous System)** é uma rede ou conjunto de redes sob a mesma administração e cujo encaminhamento interno é opaco para o exterior.

Os protocolos de encaminhamento são classificados em **protocolos IGP (Internal Gateway Protocols)** ou **Intra-AS**, ver o Capítulo ??, os protocolos usados dentro dos ASs, e **protocolos EGP (External Gateway Protocols)** ou **Inter-AS**. Neste capítulo introduziremos o protocolo deste tipo que é mais popular, o protocolo BGP.

### Encaminhamento inter-AS

Um AS conhece os outros ASs pelos seus números de AS e pelos prefixos IP que lhes pertencem. Como veremos a seguir, para efeitos de encaminhamento, os ASs

<sup>1</sup> Antigamente os *routers* que asseguravam a interligação das redes eram designados como *gateways*, daí a designação do protocolo.

podem ser assimilados e relacionam-se entre si como *routers* virtuais e os diferentes ASs encontram-se ligados por múltiplos canais, todos logicamente do tipo ponto-a-ponto. Do ponto de vista do encaminhamento inter-AS, não existem restrições à forma como os diferentes ASs se ligam entre si.

Os sistemas computacionais com endereços num dos prefixos pertencentes a um AS estão necessariamente no seu interior. O encaminhamento inter-AS tem por objectivo fazer chegar os pacotes ao AS a que pertence o prefixo que contém os seus endereços de destino. O encaminhamento intra-AS assume a responsabilidade de fazer a entrega ao sistema final.

Os ASs estabelecem assim uma partição das redes interligadas na Internet e uma partição do espaço dos prefixos IP. A este nível um prefixo IP visível está confinado a um único AS e é autónomo (do ponto de vista do encaminhamento pela regra do *longest prefix matching rule*) pois não pode estar subordinado do ponto de vista do encaminhamento a nenhum outro prefixo (mesmo os que o contenham). A Figura 18.2 ilustra diversos ASs, as sua relações com os prefixos e as suas interconexões.

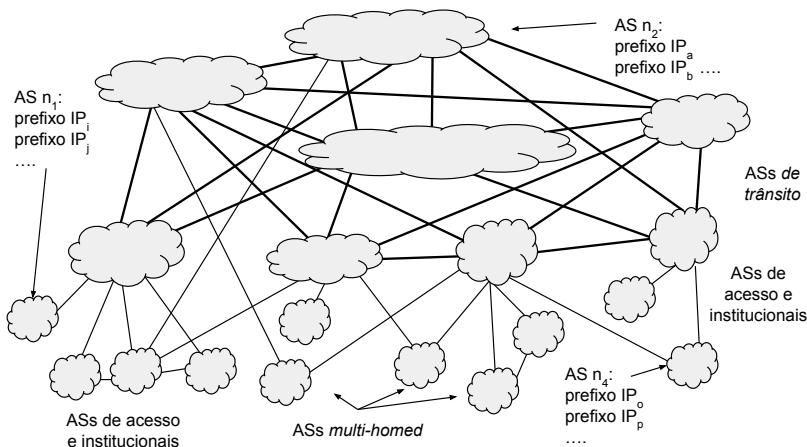


Figura 18.3: A Internet é formada por um conjunto de ASs interligados. Cada AS tem um número próprio e um conjunto autónomo de prefixos IP. Tipicamente, os ASs de maior dimensão são T-ier 1, os intermédio são Tier-2 e os mais pequenos Tier-3 ou *Stub-ASs*

O encaminhamento inter-ASs é assegurado pelo protocolo **BGP (Border Gateway Protocol)** que é geralmente executado pelos *routers* de fronteira ou *border routers*, os quais são responsáveis pela interligação dos ASs.

Todos os prefixos IP existentes na Internet pública pertencem necessariamente a um AS, ou estão incluídos num que pertence. Quando uma rede está ligada a diferentes ASs, constitui necessariamente um AS específico.

### Stub ASs e Transit ASs

A tipologia dos ASs e a caracterização das suas relações já foi brevemente introduzida na Secção 4.4. Uma primeira classificação dos ASs tem a ver com a sua dimensão. Os maiores ASs são designados por ASs *Tier-1* (nível 1), os intermédios por ASs *Tier-2*

(nível 2) e os mais pequenos por ASs *Tier-3* (nível 3). Neste capítulo a classificação que nos interessa é outra: os ASs também podem ser classificados em *Stub ASs* e *Transit ASs*, em função da origem e destino dos pacotes que neles circulam.

Os *Stub ASs* estão na periferia da Internet, *i.e.*, são os ASs dos “extremos”. Os pacotes que neles circulam têm um endereço de origem ou um endereço de destino contido nos seus prefixos próprios. Ou seja, o trajecto desses pacotes ou começa ou acaba nesses ASs.

Ao contrário, os *Transit ASs* asseguram o trânsito dos pacotes na Internet pois também transportam pacotes originados e com destino a outros ASs. Os ASs de trânsito transportam pacotes cujos endereços de origem e de destino não pertencem a sistemas finais existentes dentro da sua rede. Muitos *Transit ASs* também são *Stub ASs* para alguns dos seus prefixos.

Os *Stub ASs* que estão ligados a mais do que um AS mas que não fazem trânsito entre os mesmos, chamam-se *multi-homed ASs*. Tecnicamente não existem restrições a quem pode ou não ser um AS. Uma rede doméstica pode adquirir o estatuto (*e a complexidade de interligação*) de um AS.

Se considerarmos a analogia com *routers* virtuais, os *Stub ASs* são os sistemas finais e os *Transit ASs* são os sistemas intermediários (os *routers*). À primeira vista os *Stub ASs* e os *multi-homed ASs* deveriam ser todos de tipo *Tier-3*, *i.e.*, de pequena dimensão, mas essa correspondência não é directa. Para além de algumas multinacionais de grande dimensão, os ASs dos grandes fornecedores de conteúdos (*e.g.*, Google, Facebook, Amazon, *etc.*), que têm centros de dados directamente interligados entre si espalhados pelo mundo, mas que não fornecem serviços de trânsito de pacotes pois apenas recebem e enviam pacotes de outros ASs dirigidos aos seus servidores, são *multi-homed ASs*, mas do ponto de vista da dimensão seriam equiparáveis a ASs do tipo *Tier-1*.

### Caderno de encargos do encaminhamento inter-ASs

Os ASs estão interligados entre si por múltiplos canais. Por exemplo, os ASs *Tier-1* são transcontinentais e interligam-se com outros ASs *Tier-1* em diferentes cidades dos diferentes continentes. Os ASs *Tier-2* e *Tier-3* estão geralmente confinados a continentes ou a países, no entanto, para melhorarem a sua conectividade, estão também ligados a outros ASs em diferentes continentes, nomeadamente a diversos ASs *Tier-1*. Em geral, os *Stub ASs* apenas estão ligados a ASs da sua região geográfica, com exceção dos ASs dos grandes operadores de conteúdos que também são intercontinentais e estão ligados de forma generalizada aos outros operadores. Com efeito, para melhorarem a sua conectividade, é frequente estes ASs terem *reverse proxies*, ver a Secção 13.3, e *border routers* no interior das instalações de centenas de operadores.

Muitas cidades que concentram muita população e actividade económica e também muito tráfego Internet, dispõem de infra-estrutura de interligação de operadores, chamadas Internet *exchange points* ou Internet *peering points*. Tratam-se de edifícios onde se concentram muitos *border routers* dos operadores e onde é fácil estabelecer interconexões entre os mesmos.

O número de prefixos IP para os quais deve ser assegurado o encaminhamento na Internet era, no primeiro trimestre de 2016, cerca de 620.000, e o número de diferentes sistemas autónomos activos é de cerca de 55.000. Alguns dos maiores ASs têm milhares de ligações a outros ASs. A maioria dos ASs intermédios tem dezenas de ligações a outros ASs. Felizmente, a maioria dos *Stub ASs*, que são a larga maioria dos ASs, têm ligação a apenas um só AS. De qualquer forma, a escala do grafo dos ASs é muito grande e o grau dos seus nós centrais também é muito elevado.

O encaminhamento inter-ASs tem também requisitos especiais, relacionados com a estrutura da Internet e as relações, comerciais ou políticas, que são estabelecidas entre os operadores das redes de acesso, de centros de dados e de trânsito. Essas relações são geralmente do tipo cliente / fornecedor ou de cooperação. O protocolo não poderá

impõe constrangimentos a essa relações, nem influenciar o jogo da concorrência e das relações comerciais. Terá de ser neutro desses pontos de vista. Por exemplo, será essencial que admita a seleção de caminhos por critérios comerciais ou políticos.

Uma solução que implicasse uma hierarquização estrita da rede e dos prefixos resultaria muito difícil de gerir e pouco flexível, pois implicaria uma estrutura de relações entre ASs muito associada à dimensão e à geografia e a áreas de influência geo política. Adicionalmente, essa estrutura teria de ser pré-planeada e não seria fácil de alterar. Esta solução não é adequada pois essa estrutura dificilmente permitiria a flexibilidade e liberdade de escolha de caminhos acima enunciada.

#### **Que tipo de protocolos de encaminhamento escolher?**

Os protocolos baseados em anúncios de visibilidade permitem controlar os anúncios, fazendo-os de forma selectiva. Quando um anúncio é feito a um vizinho, quer dizer que se lhe está comunicar que se conhece um caminho para o destino, e também que se está disposto a encaminhar pacotes para esse destino. Uma maneira de aceitar encaminhar os pacotes enviados por um AS para um certo destino, e não aceitar para outro, passa por anunciar-lhe esse destino e esconder-lhe o outro. Assim, estes protocolos revelam maior flexibilidade para controlar a visibilidade e os caminhos possíveis.

No entanto, os protocolos do tipo “vector de distâncias” têm o problema de poderem introduzir ciclos de encaminhamento, por isso é necessário encontrar formas de o impedir. Para este efeito é possível incluir no anúncio o caminho usado para lá chegar. Desta forma o receptor do anúncio pode verificar se faz parte do mesmo e ignorar os anúncios que conduziriam a ciclos de encaminhamento. Por outro lado, conhecer o caminho usado também dá mais informação para o processo de escolha dos caminhos. Este tipo de protocolos dizem-se protocolos baseados em “vectores de caminhos” ou AS *paths*.

#### **AS *paths***

O protocolo BGP é um protocolo baseado em anúncios de caminhos, materializados numa lista de números de AS e designados por AS *paths*. Um anúncio BGP contém um ou mais prefixes IP, um AS *path*, i.e., o caminho em ASs para lá chegar, e ainda o endereço IP do *next hop*, i.e., o endereço IP do *border router* para que deve ser dirigido o tráfego destinado aos prefixes assim anunciados. Para cada prefixo, um *border router* pode receber dos seus vários vizinhos diversos AS *paths*, selecciona um deles e, se desejar, acrescenta o seu número de AS a esse AS *path* e propaga-o num novo anúncio com o AS *path* modificado.

O protocolo só propaga o AS *path* seleccionado pelo anunciante e não todos os que este conhece, senão o número de AS *paths* explodiria. A Figura 18.4 ilustra os anúncios BGP e como estes são propagados de AS em AS. Repare-se também que os anúncios fluem num sentido e o tráfego no sentido inverso. É como se os anúncios atraíssem o tráfego.

Um anúncio ou rota BGP é constituída por uma lista de prefixes e vários atributos comuns aos mesmos. Entre estes figura um caminho expresso como uma lista dos ASs, chamado AS *path*, que o anunciante usa para chegar ao AS onde residem os prefixes. Uma rota contém também um atributo com o endereço do *border router* de recepção do tráfego que é dirigido aos seus prefixes.

Os AS *paths* permitem saber exactamente por que ASs passarão os pacotes, o que facilita a tomada de decisão em função do caminho proposto e permite suprimir a introdução de ciclos de encaminhamento.

Dado os anúncios BGP conterem caminhos, o protocolo é caracterizado como sendo um protocolo com base em **prefixos e vectores de caminhos (prefix-based path-vector protocol)**.

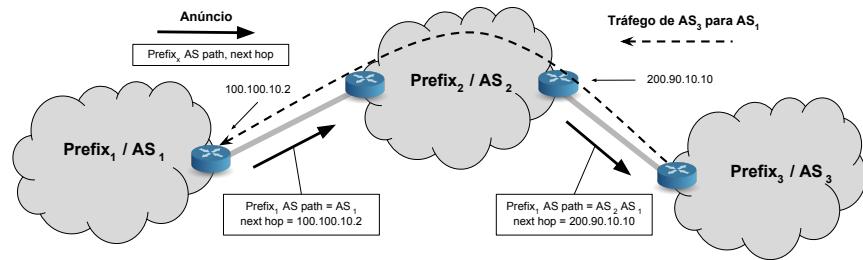


Figura 18.4: Forma e propagação dos anúncios BGP e como os mesmos atraem o tráfego dirigido ao prefixo anunciado.

### Anúncios BGP e relações comerciais entre ASs

As relações comerciais entre ASes são de diferentes tipos, nomeadamente: **cliente / fornecedor** (e vice-versa) e **peer-to-peer**, ver a Figura 18.5.

As relações cliente / fornecedor são simples de perceber: o  $\text{AS}_a$  é cliente de trânsito pago do  $\text{AS}_b$  se  $\text{AS}_a$  paga ao  $\text{AS}_b$  para este fazer chegar os seus pacotes e os dos seus clientes ao destino final, e também para que o  $\text{AS}_b$  lhe entregue os pacotes vindos de outros ASes que se destinam a ele ou aos seus clientes.

Para que o  $\text{AS}_b$  seja fornecedor do  $\text{AS}_a$ ,  $\text{AS}_b$  tem de anunciar aos outros ASes a que está ligado que tem um caminho para os prefixos do  $\text{AS}_a$  incluindo os dos seus clientes. Quando um AS fornece serviços de encaminhamento a outro AS diz-se que o primeiro fornece trânsito pago ao outro (*provides a paid transit service*).

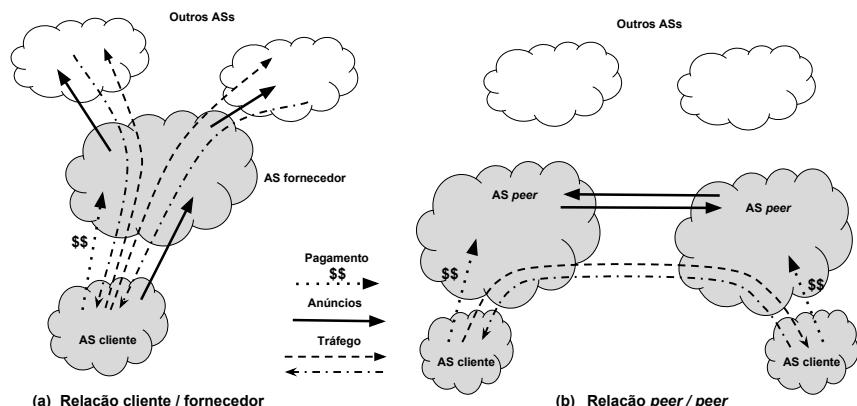


Figura 18.5: Relações entre ASs clientes e fornecedores (a) e relações entre ASs peers (b). Fluxos de anúncios dos prefixos, do tráfego e do dinheiro.

Quando os  $\text{AS}_a$  e  $\text{AS}_b$  têm uma relação do tipo *peer-to-peer*, quer dizer que  $\text{AS}_a$  está disposto a receber pela ligação a  $\text{AS}_b$  os pacotes originados em  $\text{AS}_b$ , ou nos seus clientes, e que se dirigem a  $\text{AS}_a$ , ou aos seus clientes, e vice-versa. Isso quer dizer que  $\text{AS}_a$  e  $\text{AS}_b$  anunciam mutuamente um ao outro os seus prefixos e os dos seus clientes. No entanto, quer  $\text{AS}_a$  quer  $\text{AS}_b$  não anunciam esse prefixos para terceiros ASes. Em geral, os serviços deste tipo não implicam pagamentos caso as quantidades de tráfego

enviadas em cada sentido sejam da mesma ordem de grandeza. No entanto, em caso de assimetria, é frequente os contratos de *peering* incluírem pagamentos.

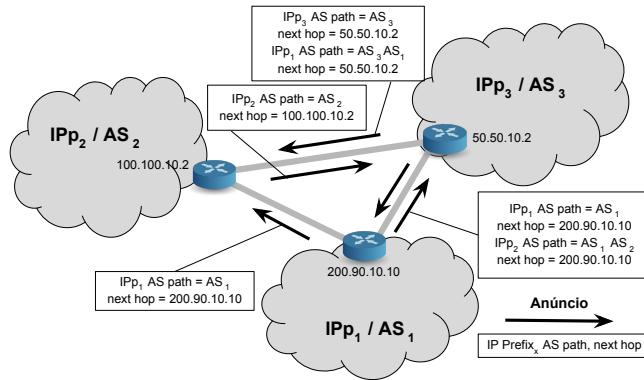


Figura 18.6: Utilização dos anúncios BGP para tomar decisões políticas sobre as escolhas de encaminhamento.

O controlo sobre os anúncios de prefixes através de *AS paths* BGP é muito flexível e permite implementar diversos tipos de políticas. A Figura 18.6 permite ilustrar esta faceta do protocolo. Por exemplo, o *AS<sub>2</sub>* recebe anúncios da acessibilidade do prefixo *IPp<sub>1</sub>* via o próprio *AS<sub>1</sub>* e via o *AS<sub>3</sub>*, no entanto, pode preferir alcançar esse prefixo passando pelo *AS<sub>3</sub>* visto que este caminho é mais barato (em termos monetários) que o directo. Mas se este caminho deixar de estar disponível, pode optar pelo caminho directo para o *AS<sub>1</sub>*. O protocolo BGP vai-lhe permitir fazer essas escolhas. Por outro lado, o *AS<sub>1</sub>* prefere esconder ao *AS<sub>2</sub>* que conhece um caminho para *IPp<sub>3</sub>* e assim *AS<sub>2</sub>* não pode usar essa via caso o seu canal para *AS<sub>3</sub>* não esteja disponível, visto que não sabe que a mesma existe.

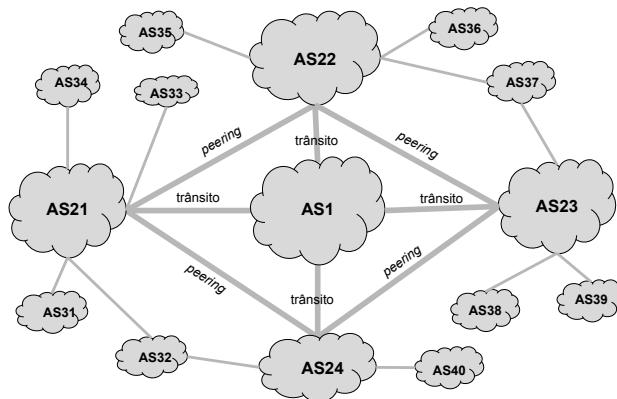


Figura 18.7: Um conjunto hipotético de ASes de diversos *tiers* com relações cliente / fornecedor e de *peering* entre eles.

### Exemplo mais completo

O exemplo que se segue, ver a Figura 18.7, é também hipotético mas mais completo e corresponde a opções comuns no mundo dos operadores. O AS1 fornece trânsito pago aos AS21, AS22, AS23 e AS24. Estes três ASs fornecem trânsito pago aos seus clientes directos e têm relações de *peering* mútuas dois a dois. Os AS232 e AS237 são *multi-homed*. Os prefixos estão omitidos para maior clareza da figura, mas por hipótese cada AS tem um único prefixo, correspondente à sua rede e com o mesmo nome que o seu AS.

### Discussão: anúncios realizados

A implementação destas políticas passa por os ASs AS31 até AS40 apenas anunciam o próprio prefixo para os seus fornecedores directos (lembre-se que cada AS apenas faz um anúncio sobre cada prefixo IP a cada um dos ASs a que está directamente ligado). Cada um dos *Tier 2* ASs (AS21 a AS23) anuncia os seus clientes directos para o AS1 (*Tier 1 AS*) e para os seus *peers*. O AS1 anuncia para os seus clientes de nível 2 todos os ASs que conhece, *i.e.*, todos os AS existentes na rede. Este AS representa a raiz de uma árvore que assegura a conectividade completa do conjunto. Os ASs do nível 2 têm caminhos para todos os ASs existentes, via os canais que os ligam aos seus clientes directos, ou via os canais que os ligam aos seus *peers*, e ainda para toda a restante rede via o AS1. Por isso anunciam todos os prefixos que conhecem aos seus clientes directos que assim passam a ter conectividade para toda a rede. Os clientes *multi-homed* necessitam destes anúncios para escolherem um AS *path* para cada destino. No entanto, os outros ASs clientes finais apenas precisam de uma rota por omissão (*default*) visto que não têm alternativas.

### Discussão: políticas implementadas pelo anúncios

As políticas descritas conduzem à utilização de AS *paths* específicos. Por exemplo, o AS31 envia pacotes para o AS34 através do AS21, *i.e.*, pelo AS *path* [AS21, AS34]; envia pacotes para o AS35 via o AS *path* [AS21, AS22, AS35] visto que a ligação de *peering* é preferida à ligação a pagar via o AS1; e envia pacotes para o AS38 via o AS *path* [AS21, AS1, AS23, AS38] visto que essa é a única via de que dispõe para lá chegar (o *peering* não é transitivo).

Supondo que por algum motivo os canais que ligam o AS22 ao AS23 e ao AS1 ficarem inoperacionais, existem diversos caminhos que permitiriam ao AS36 chegar ao AS38, mas nenhum deles é utilizável. O AS *path* [AS36, AS22, AS37, AS23, AS38] não pode ser usado porque o AS37 não fornece trânsito aos seus dois fornecedores e o AS *path* correspondente não existe na rede. O AS *path* [AS36, AS22, AS21, AS1, AS23, AS38] não é utilizável porque o AS21 não fornece trânsito a AS22, apenas *peering*, e o AS38 não é cliente de AS21.

### Discussão: alternativas de políticas

Em geral, os diferentes operadores são concorrentes e portanto, segundo a teoria de que entre empresas com fins lucrativos não “existem almoços grátis”, estas só fazem “favores” em troca de “pagamentos ou vantagens”. Por outro lado, se os clientes pagam pelos serviços de conectividade recebidos, só estão dispostos a fornecerem conectividade de trânsito, se os seus canais a suportasse, em troca também de “pagamentos ou vantagens”.

No entanto, os AS21 e o AS22 poderiam estabelecer um acordo mais completo em que, para além do *peering*, poderiam fornecer *backup* mútuo para o caso em que os canais que os ligam ao AS1 tivessem anomalias. Ou seja, caso os caminhos via AS1 não fossem acessíveis, passariam a usar o caminho via o parceiro. Para esse efeito, cada um deles deveria não só anunciar ao parceiro os seus clientes directos, mas também todos os ASs anunciados por AS1. Resta discutir como é que seria possível dar prioridades

aos diferentes caminhos, mas isso será mais claro quando discutirmos o protocolo BGP na Secção 18.3.

Com os protocolos convencionais, que só tomam decisões em termos de custos, só seria possível implementar uma política de conectividade total, usando sistematicamente todos os canais disponíveis em cada momento. Isso só seria adequado no caso em que o conjunto das redes envolvidas estivessem sob a mesma administração e autoridade. Ora no mundo dos operadores e dos clientes, esse tipo de relações não são adequadas dado o quadro de competição. O maior operador tem vantagens de escala que quer sempre aumentar para ter maior rentabilidade, ou até comprar os competidores.

Os ASs que só têm relações de *peering* com outros ASs e não são clientes de nenhum AS, dizem ASs *Transit-only*, visto que não dependem de anúncios “pagos”, isto é, recebidos ou propagados porque o AS paga a outro AS para esses efeitos. Geralmente, a maioria dos AS *Tier 1* são *Transit-only*.

### Características do grafo dos ASs

O número de ASs presentes nas tabelas de encaminhamento do conjunto das ASs *Tier 1* subiu de 5.000 em 1999 para cerca de 35.000 em 2010. Uma subida de cerca de 700% em 10 anos. Esta subida está relacionada com alguma subida do número de operadores, mas sobretudo com a generalização de *multi-homing* e outras práticas que implicam dispor de um AS próprio e fazer encaminhamento BGP. No fim do primeiro trimestre de 2016, os ASs da DFZ são cerca de 55.000, uma subida inferior a 100% em 6 anos.

Como os *Internet peering points* existem nas cidades onde o tráfego é mais intenso, e se torna mais conveniente terminar os cabos de fibras ópticas de longa distância, nos respectivos edifícios existem numerosos *border routers* concentrados e estabelecem-se inúmeras ligações negociadas entre os mesmos.

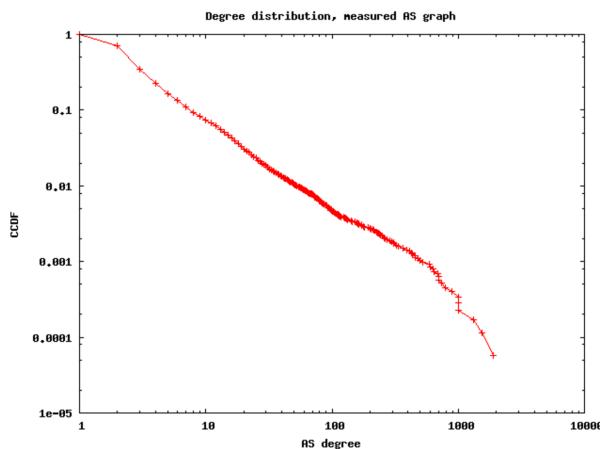


Figura 18.8: Distribuição cumulativa da probabilidade de um AS ter grau superior a um valor dado nas abscissas. Gráfico obtido no site <https://www.caida.org/research/topology/generator>.

Na prática verifica-se que o grau dos diferentes ASs, *i.e.*, o número de ligações de que cada um dispõe com os outros, é muito mais elevado nos ASs *Tier-1* do que nos outros. Por um lado estes ASs têm relações de *peering* entre eles e fornecem trânsito pago a muitos ASs de trânsito. Por outro, tal como na Web “*the winner takes it all*”, e

esses fornecedores concentram igualmente ligações de muitos clientes com *Stub ASs* de grande valor. A maioria dos ASs *Tier-2* têm várias centenas de ligações, quer a outros AS do mesmo ou de mais elevado *Tier*, quer também a muitos clientes. A grande maioria dos *Stub ASs* têm grau 1.

Existem diversos servidores de observação do funcionamento do BGP como por exemplo: <http://www.cidr-report.org> e <http://as-rank.caida.org>, que fornecem estatísticas publicamente acessíveis. Por exemplo, o gráfico da Figura 18.8, extraído do site da Caida no outono de 2016, mostra a distribuição cumulativa em percentagem do grau dos diferentes ASs. Assim, 100% dos ASs têm grau maior ou igual a 1 (um AS com grau 0 estaria isolado da Internet), menos de 10% têm grau superior a 10, enquanto que apenas uma percentagem inferior a 1% têm grau superior a 100.

A tabela 18.1 contém algumas informações sobre alguns ASs de grande dimensão, provavelmente todos do tipo *Tier-1*. Através da mesma é possível observar que os ASs mais importantes da Internet anunciam um elevado número de prefixos IP, e fornecem conectividade, directa ou indirectamente, a muitos outros ASs. A tabela inclui também vários ASs de menor *ranking* para comparação, e mostrar como a grande maioria dos 55.000 ASs são de facto *Stub ASs* que só dão acesso aos seus prefixos.

Tabela 18.1: Lista ordenada dos primeiros 10 ASs ordenados pelo número de ASs alcançados directa ou indirectamente, ou *AS customer cone* (o próprio, os clientes, os clientes dos clientes, ...) e prefixos IPv4 anunciados. A tabela foi obtida em 1 de Junho de 2016 a partir do site <http://as-rank.caida.org>. Incluem-se também alguns ASs de menor *rank* para comparação. O leitor não deve esquecer que o número total de ASs é superior a 50.000.

Rank	AS number	Nome da organização	AS customer cone	Número de prefixos IPv4
1	3356	Level 3 Communications, Inc.	29.494	224.970
2	174	Cogent Communications	23.299	172.963
3	1299	TeliaSonera AB	21.954	191.391
4	2914	NTT America, Inc.	18.991	174.304
5	3257	Tinet Spa	18.140	161.377
6	6762	TELECOM ITALIA SPARKLE	14.394	123.771
7	6453	TATA COMM.S (AMERICA) Inc	12.300	135.127
8	6939	Hurricane Electric, Inc.	8.088	79.800
9	2828	XO Communications	6.251	60.271
10	1273	Cable and Wireless Worldwide	5.878	42.258
....	....	....	....	....
500	35819	Mobily-AS	49	636
501	10835	Visionary Communications, Inc.	49	473
....	....	....	....	....
999	13767	DataBank Holdings, Ltd.	21	134
10000	201952	Atel-Rybinsk LTD	1	22
....	....	....	....	....

Têm sido feitos inúmeros estudos sobre a estrutura profunda da Internet no que diz respeito à topologia do grafo dos ASs. No final deste capítulo serão fornecidas referências para alguns desses estudos e as fontes em que se baseiam.

Na Internet global o grafo de ligação entre os diferentes ASs é conhecido através dos prefixos IP e dos AS *paths* escolhidos e anunciado pelos ASs uns aos outros. A partir de *routers* da DFZ (*default free zone*) podem observar-se muitos desses AS *paths*, o que permite analisar as relações existentes entre os diferentes ASs e verificar que esses caminhos estão subordinados a relações comerciais ou de colaboração.

O grafo dos ASs é de grande dimensão pois no momento em que este livro foi escrito tem cerca de 55.000 nós e várias centenas de milhares de arcos. O número de ligações de cada AS, *i.e.*, o seu grau, é muito heterogéneo e com uma distribuição muito desigual.

Esta distribuição permite concluir que o conjunto dos ASs *Tier 1* é responsável directa ou indirectamente pela conectividade entre praticamente todos os ASs existentes.

### 18.3 Protocolo BGP

O protocolo BGP<sup>2</sup> é o protocolo que permite aos ASs directamente interligados trocarem entre si informações de encaminhamento. Um *router* que estabelece relações BGP com outros *routers* chama-se um BGP *speaker*. Geralmente os BGP *speakers* coincidem com os *border routers*, mas isso não é obrigatório.

As mensagens BGP são trocadas através de conexões TCP que se chamam sessões BGP (BGP *sessions*). Quando dois BGP *speakers* estabelecem uma sessão, trocam os anúncios que têm de comunicar um ao outro através de mensagens BGP. Depois, mantêm a conexão activa para, sempre que se produzem alterações relevantes nas suas BGP RIBs (BGP *Routing Information Bases*), enviarem actualizações um ao outro. A Figura 18.9 contém um gráfico da evolução da dimensão da BGP RIB.

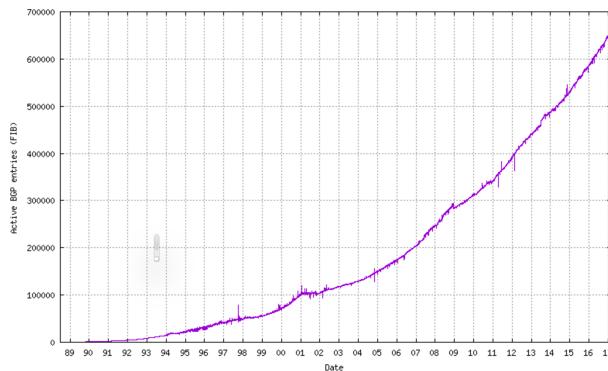


Figura 18.9: Evolução da dimensão da RIB BGP. Gráfico obtido no final de 2016 no site <http://bgp.potaroo.net/>.

Essas actualizações só contém alterações da informação transferida anteriormente. Ou seja, após a troca inicial de anúncios, a sessão só serve, enquanto durar, para enviar actualizações. Por isso os *routers* nas extremidades da conexão vigiam-se mutuamente através de mensagens de *keep alive*, que verificam se a conexão se mantém activa, o que garante que os anúncios enviados antes pelo vizinho são todos conhecidos.

Estas opções justificam-se porque a informação trocada pode ser muito volumosa e é pressuposto os *routers* memorizarem todas as rotas (identificadas por um prefixo e contendo um AS *path* e outros atributos) enviadas por cada vizinho, pelo que não é

<sup>2</sup> A descrição do protocolo que se segue baseia-se na sua versão 4, a versão em utilização no momento em que este livro foi escrito.

necessário refrescar toda a informação com origem num vizinho enquanto a sessão se mantiver operacional.

Se a sessão se quebrar, todos as rotas com origem no vizinho com que se perdeu o diálogo BGP são imediatamente suprimidas da BGP RIB. Nessa sequência, se o AS conhecer outras alternativas de rota para esses prefixos, a conectividade para os mesmos pode manter-se, senão é quebrada.

De cada vizinho, o BGP só recebe e só guarda uma única rota por ele anunciada. No entanto, os BGP *speakers* podem memorizar várias rotas diferentes para o mesmo prefixo, desde que recebidas de diferentes vizinhos. No entanto, como já foi referido, um AS só anuncia a um vizinho uma única rota com um AS *path* que deve ter origem no por si escolhido, modificado pelo concatenação do seu número de AS.

### Sessões BGP externas e internas

Um AS pode ter mais do que um BGP *speaker*. Nesse caso, as rotas BGP seleccionadas por cada *speaker* têm de ser difundidas para que os diferentes *border routers* se coordenem e, na posse de informação mais completa, tomem as opções mais adequadas. Assim, ver a Figura 18.10, o BGP mantém dois tipos de sessões: As sessões eBGP, que são sessões externas mantidas com os BGP *speakers* dos outros ASs, e as sessões iBGP, que são as sessões internas mantidas com os outros *routers* BGP do mesmo AS.

No iBGP os AS *paths* não introduzem ciclos internos pois cada *speaker* apenas propaga AS *paths* aprendidos do exterior. Assim, é necessário que cada *speaker* estabeleça uma sessão com cada um dos outros *speakers* do seu AS. Desta forma, as BGP RIBs de todos os *speakers* do AS contém as rotas seleccionadas por todos os outros *speakers* do AS.

Num AS com  $n$  destes *routers*, cada um deles tem de manter  $n - 1$  sessões iBGP. Isso pode tornar-se muito pesado, pelo que foram introduzidos outros esquemas de fazer a difusão. Por exemplo, usando um único *router* especial como reflector de rotas (*route reflector*), ver o RFC 4456. Este *router* terá de manter  $n - 1$  sessões iBGP, mas todos os outros apenas uma. Para além disso o reflector tem de evitar a introdução de ciclos internos ao AS.

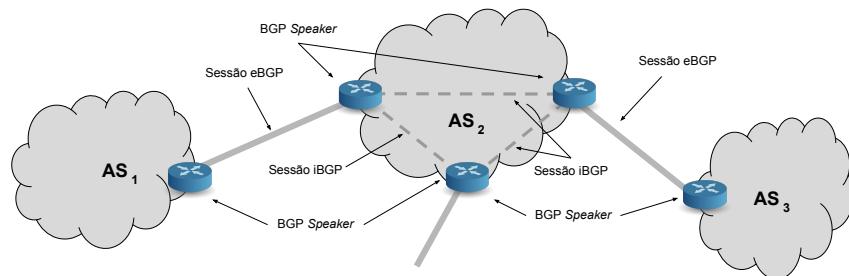


Figura 18.10: Sessões BGP estabelecidas pelos BGP *speakers* dentro do AS (sessões iBGP a tracejado) e os dos ASs vizinhos (sessões eBGP a traço grosso).

Através dos anúncios BGP os *routers* mantêm a informação das suas BGP RIBs actualizadas e tomam decisões de encaminhamento para os prefixos externos ao AS. Naturalmente, essa informação de encaminhamento tem também de ser injectada nos *routers* não BGP do AS através da colaboração entre os BGP *speakers* e os outros *routers* internos ao AS através da injeção de anúncios via os protocolos de encaminhamento do interior do AS (IGPs). Este aspecto será discutido no final da desta secção.

### Estabelecimento das sessões BGP e mensagens BGP

Os *routers* BGP são parametrizados manualmente sobre as sessões que mantêm com os seus vizinhos. Uma sessão BGP passa pelas fases indicadas na máquina de estados da parte esquerda da Figura 18.11. Primeiro os *routers* estabelecem a conexão TCP e trocam diversos parâmetros da sessão e depois iniciam a troca de anúncios de rotas. As mensagens BGP são enviadas sobre TCP, como sequências de bytes, e são as indicadas a seguir.

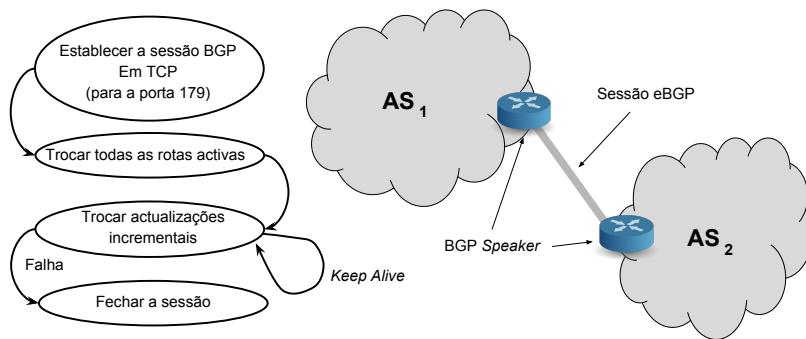


Figura 18.11: Estabelecimento das sessões BGP e fases da sessão.

**Open** Esta é a primeira mensagem enviada na sessão após o seu estabelecimento. Contém diversos parâmetros como a versão do protocolo (em 2016 a versão mais actual é a BGP 4, ver os RFCs 1771 e 4271), número do AS, identificador do *router*, duração do alarme para a deteção da interrupção da sessão e outros parâmetros opcionais.

**Keepalive** Esta mensagem serve para as partes detectarem se continuam activas e presentes na sessão.

**Notification** Serve para comunicar códigos de erro. Uma sessão que não é quebrada por um alarme é fechada por uma destas mensagens.

**Update** É através desta mensagem que os dois *speakers* trocam informações de encaminhamento. A mesma será discutida de seguida.

Um *router* BGP contém na sua RIB BGP uma tabela com todos os prefixos IP que conhece via os anúncios BGP recebidos e, para cada um deles, as rotas que pode usar para lá chegar. Destas, seleciona uma para utilização, e sempre que há uma alteração da rota seleccionada, comunica-a aos seus *peers* através de uma mensagem **Update**. A Figura 18.12 contém uma visão alto nível dos parâmetros desta mensagem. Esta permite o envio de várias actualizações de uma só vez pois tem vários campos de comprimento variável.

Na primeira parte da mensagem, correspondente às **withdrawn routes**, são indicados prefixos IP para os quais o emissor deixou de ter uma rota (um ou mais prefixos IP envolvendo uma ou mais rotas suprimidas). A seguir é indicada uma nova rota, para um ou mais prefixos IP, que o emissor instalou. A rota começa por um campo designado **path attributes field**, que indica um AS *path* e outros atributos sobre a rota e, finalmente, na parte **Network layer reachability info**, são enviados os prefixos que a mesma permite alcançar.

Alguns dos atributos de uma rota são os seguintes: **Origin** (opções: obtida via o IGP, via o EGP ou por outra forma, como por exemplo por encaminhamento estático), **AS path** (sequência de um ou mais ASNs terminado necessariamente pelo AS

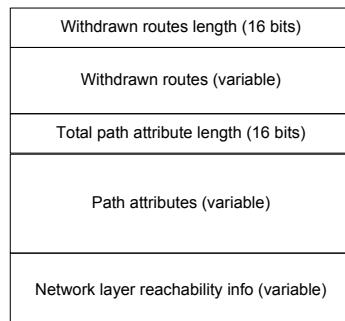


Figura 18.12: Mensagem BGP Update

*number* do *router* anunciante) e **next-hop** (o endereço do BGP *speaker* para o qual devem ser enviados os pacotes alvo da rota). Assim, um **Update** pode suprimir vários prefixos para os quais o emissor deixou de ter uma rota, e anunciar vários prefixos que partilham uma rota com um conjunto de atributos como o *AS path*, *next-hop*, etc.

O BGP é o protocolo que assegura a conectividade da Internet global e que tem como requisitos ser capaz de lidar com centenas de milhar de prefixos IP e várias dezenas de milhar de redes interligadas, cada uma das quais com políticas específicas de encaminhamento. Apesar desta escala, pretende-se que convirja rapidamente.

O protocolo é executado por *routers* especiais, ditos **BGP speakers**. Estes *routers* coincidem geralmente com *border routers*, os *routers* que asseguram a interligação dos AS. O diálogo entre os BGP *speakers* é assegurado através de **sessões BGP (BGP sessions)** que são ponto-a-ponto e utilizam conexões TCP. Os anúncios são incrementais e só comunicam actualizações (novas rotas ou supressão de rotas).

### Funcionamento do protocolo e seleção dos caminhos

A Figura 18.13 mostra como a rota para o prefixo 193.136.0.0/15, propagada inicialmente pelo AS1 (à esquerda), se transforma em várias rotas alternativas conforme vai sendo propagada pelos diferentes ASs que aceitam fornecer trânsito para esse prefixo. Assim, quando o AS6 (à direita) recebe as mesmas, vai dispor de duas rotas para o prefixo. A questão que se coloca então é saber como é que esse AS se decide por uma delas. Um protocolo de encaminhamento IGP usaria para esse efeito a noção de custo. No entanto o BGP não tem, por construção, essa noção numa forma convencional, e precisa de satisfazer outros requisitos.

O BGP não tem nenhuma noção convencional de custo pois cada AS pode usar diferentes IGPs e diferentes funções de custo. Por outro lado, um AS pode cobrir um continente e pode-se atravessá-lo de 1 a 50 ms ou mais, dependendo da entrada e saída usadas. A única função de custo que o BGP pode usar é comparar o número de ASs atravessado pela rota. No exemplo da Figura 18.13 este critério conduziria à seleção pelo AS6 da rota via o AS5 visto que esta tem 3 ASs de comprimento, inferior à rota via o AS4 que atravessa 4 ASs. No entanto, o AS não sabe se isso conduz ou não a um menor tempo de trânsito, ou uma maior capacidade da rota, ou a uma rota de melhor qualidade pois a probabilidade de um pacote se perder é menor. Para isso ter-se-ia de conhecer as características dos ASs atravessados para os comparar.

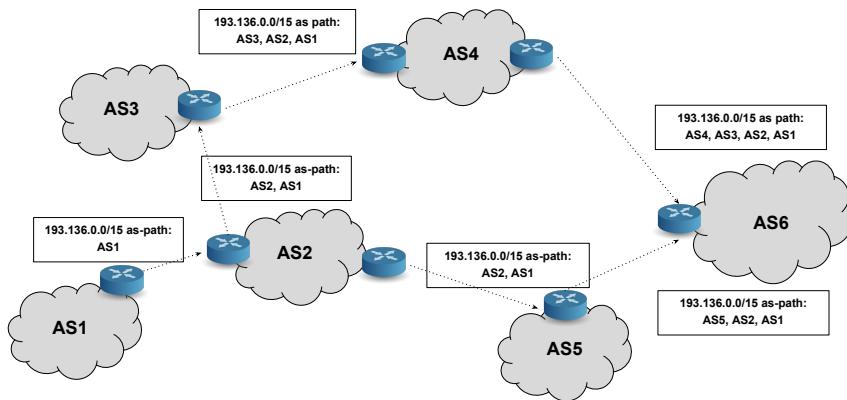


Figura 18.13: Propagação dos AS *paths* através dos ASs e recepção por um AS de mais do que uma rota para o mesmo prefixo.

No entanto, é aqui que entram em jogo outras preferências como por exemplo as ditadas por razões comerciais ou políticas. A norma do protocolo BGP fixa um conjunto de procedimentos a que qualquer *router* BGP deve obedecer para seleccionar rotas, mas deixa em aberto os fabricantes introduzirem mecanismos sofisticados em algumas delas, baseados por exemplo em linguagens de manipulação de *updates*.

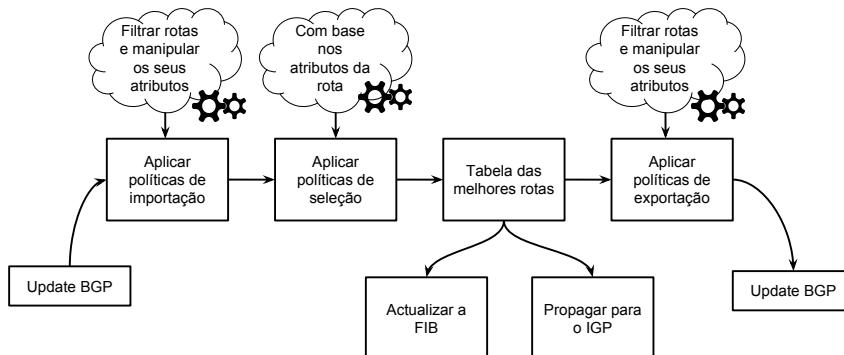


Figura 18.14: Passos seguidos desde que um BGP *update* é recebido até que o mesmo se repercuta na FIB do *router*, é propagado aos outros *routers* através do IGP e dá origem a outro(s) *update*(s) enviados para outros ASs.

A Figura 18.14 mostra os passos seguidos após a recepção de uma rota até que a mesma é tomada em consideração ou não, e eventualmente propagada aos vizinhos.

Inicialmente os atributos da rota são analisados e é verificado se a mesma é aceite ou não. Se for aceite, pode receber atributos suplementares. Existem várias razões que podem levar um *router* a ignorar certas rotas. Por exemplo, porque a mesma vem de um fornecedor mas contém o AS de um seu cliente, ou porque atravessa um AS que se pretende evitar. Uma política bastante comum consiste em filtrar todas as rotas enviadas por um cliente sobre cujos prefixos este não tem autoridade. O resultado

deste processo pode consistir em ignorar a rota ou, em alternativa, fixar-lhe o valor do atributo **LOCAL\_PREF** (preferência local) afectando-lhe uma preferência mais baixa.

Por exemplo, as rotas dos *peers* devem ser preferidas às dos fornecedores por serem mais baratas, mas as rotas recebidas via os fornecedores devem manter-se como alternativa menos prioritária para suprir a falha do canal para o *peer*. Estas regras de filtragem e modificação são expressas em linguagens que variam na sintaxe e sofisticação de fabricante para fabricante, visto que a norma não as define, apenas define os atributos das rotas.

Se a rota for aceite é memorizada e em seguida as diferentes alternativas recebidas dos diferentes vizinhos são comparadas para seleção da preferida pelo *router*. O processo de comparação das rotas usa diversos critérios aplicados sequencialmente, ver a Tabela 18.2. Passa-se ao critério seguinte quando a aplicação do corrente não permite discriminar as rotas em comparação e eleger a melhor delas.

Tabela 18.2: Critérios normalizados pelo BGP para comparar diferentes rotas para o mesmo grupo de prefixos IP

Prioridade	Atributo	Descrição, exemplo ou explicação
1	<b>LOCAL_PREF</b>	<i>Local preference value attribute: policy decision</i> <i>e.g., preferir as rotas dos peers às dos providers</i>
2	<b>AS_PATH</b>	<i>Shortest AS-PATH</i> Preferir as rotas com um número inferior de ASs
3	<b>MED</b>	<i>Lowest Multi-Exit-Discriminator</i> (MED) Se contratual, seguir as indicações do outro AS
4	<b>IGP Path</b>	<i>Closest Next-hop</i> Usar <i>Hot-potato routing</i>
5	<b>eBGP &gt; iBGP</b>	<i>External route better than internal</i> Preferir a rota recebida de outro AS
6	<b>Router ID</b>	<i>Smallest Next-hop IP Address</i> À falta de melhor

O primeiro critério é exactamente o da preferência local. É dada preferência a rotas que no passo anterior foram definidas como mais interessantes. Caso este critério não permita a seleção de uma rota, usa-se o critério seguinte que consiste em comparar o comprimento do *AS path*. É claro que, tal como a Figura 18.15 ilustra, comparar o número de ASs não é o mesmo que comparar rotas do ponto de vista das métricas convencionais dos IGP.

O critério seguinte tem a ver com o atributo *Lowest Multi-Exit-Discriminator* (MED) que é fixado pelo vizinho que comunicou a rota. Este critério permite a um AS indicar a um vizinho com o qual tem mais do que uma ligação, qual a que prefere que este use para lhe enviar tráfego por isso lhe ser mais conveniente. Caso o AS que recebe as rotas estiver obrigado contratualmente em seguir esta indicação, a mesma deverá ser respeitada.

A alternativa consiste em usar a rota tal que o *next-hop* indicado pelo vizinho está a menor distância do AS local do ponto de vista do IGP. Este tipo de critério conduz ao chamado *hot-potato routing*, um dos principais responsáveis por as rotas na Internet global não serem necessariamente simétricas.

#### ***Hot-potato routing***

Quando um AS recebe um pacote em trânsito, deve tentar entregá-lo o mais depressa que possível ao próximo AS, como se de uma batata quente se tratasse, pois menos

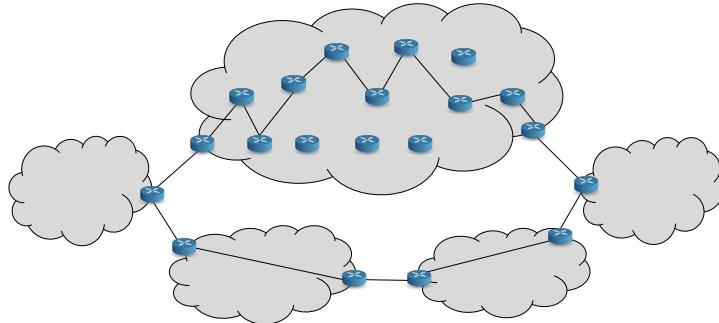


Figura 18.15: Comparar rotas pelo número de ASs pode revelar-se enganador.

recursos gasta com o mesmo. Quando a política seguida consiste em ignorar qualquer indicação enviada pelo vizinho e seguir a política do *closest next-hop*, quer dizer que a rota que é seleccionada é a que corresponde ao *router* mais próximo do vizinho. Se dois ASs *peers* de grande dimensão usarem esta política, as rotas seguidas pelos pacotes resultam assimétricas, ver a Figura 18.16.

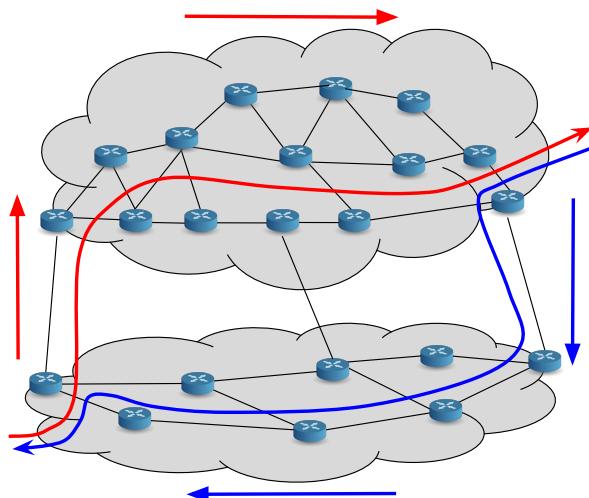


Figura 18.16: Política *hot-potato routing* e rotas assimétricas.

Finalmente, o *router* pode dar preferência a rotas recebidas dos vizinhos externos do que a rotas recebidas de outros BGP *speakers* do mesmo AS, e depois, à falta de melhor critério pode usar um factor arbitrário de desempate, que é grosso modo equivalente a tirar à sorte.

Uma vez a rota seleccionada, a mesma é repercutida nas FIBs e pode ser exportada para os outros ASs via eBGP. Em qualquer caso, deve ser propagada por iBGP aos outros BGP *speakers* do AS que eventualmente existam.

### Políticas de exportação de rotas

Antes de uma rota seleccionada ser propagada para o exterior do AS é sujeita a filtragem e afinação dos seus atributos. Existem vários razões para não exportar rotas, nomeadamente, como já vimos, o não querer receber tráfego de um vizinho dirigido às mesmas. Por exemplo, não exportar rotas recebidas de um *peer*.

Mesmo quando uma rota é exportada, o seu AS *path* é modificado acrescentando o número do AS local. Uma transformação suplementar consiste em acrescentar várias vezes o número do AS local (*repeated AS prepending*) que conduz a um aumento artificial da dimensão desse AS *path*. Esta transformação é frequentemente usada pelos ASs *multi-homed* pois permite-lhes tentar influenciar qual o fornecedor que é escolhido pelo resto da Internet para lhes enviar pacotes ou fazer distribuição de carga entre as diversas ligações externas que possui.

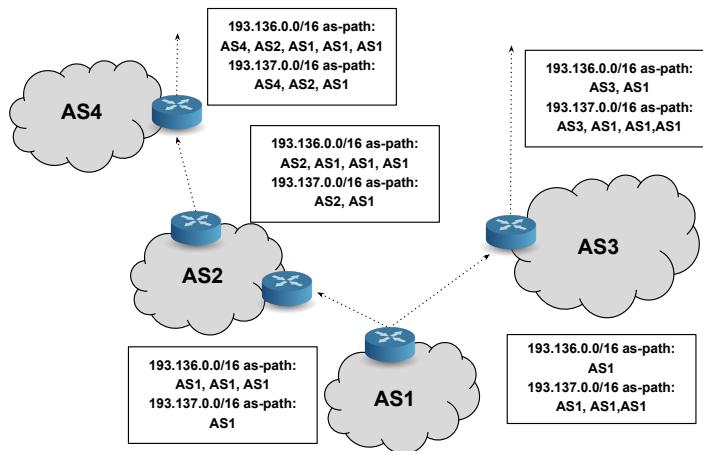


Figura 18.17: Utilização de *repeated AS prepending* para influenciar a distribuição do tráfego de entrada num AS *multi-homed*.

Na Figura 18.17 ilustra-se uma utilização desta técnica para fazer distribuição de carga. O prefixo do AS1 (193.136.0.0/15) foi dividido em dois (193.136.0.0/16 e 193.137.0.0/16) e cada um deles é anunciado de forma diferente a cada um dos fornecedores. Desta forma, à primeira metade dos endereços do prefixo é dada prioridade ao caminho via o AS3, enquanto que à segunda metade dos endereços do prefixo é dada prioridade via o AS2. Este tipo de procedimento leva ao aumento do número de prefixes distintos anunciados na Internet global mas o BGP não dispõe de outros mecanismos para controlar a distribuição de carga.

### Convergência do BGP

A convergência do protocolo BGP é um processo que em certas circunstâncias é bastante demorado dado o volume de informação envolvido. Com efeito, quando um canal de interligação de dois ASs vai abaixo, os *border routers* têm de desencadear diversas operações envolvendo bastante processamento.

Em primeiro lugar têm de suprimir todas as rotas da sua BGP RIB que foram anunciadas pelo ex vizinho. Em seguida, caso as rotas suprimidas tivessem sido seleccionadas como as melhores, é necessário enviar supressões de rotas por eBGP e iBGP e actualizar os protocolos IGP usados visto que os *routers* convencionais do AS devem também ser notificados da ausência dessas rotas e, finalmente, é necessário seleccionar uma rota alternativa e, caso a mesma exista, eventualmente propagá-la aos ASs

vizinhos e ao IGP. Como alguns ASs partilham centenas de rotas, algumas avarias de canais, ou simples reinicializações de sessões ou eventuais problemas de software de BGP *speakers* podem desencadear muitas actualizações.

Ao contrário de um IGP do tipo vector de distâncias, ver a Secção 16.4, mesmo que a conectividade para um prefixo se mantenha e o número de ASs para lá chegar não se altere, a simples alteração do caminho tem de ser comunicada e é propagada como uma alteração.

Nos *site* <http://potaroo.net> é possível ver que o número de alterações de rotas recebidas por um *border router* da DFZ é muito grande e possui picos que podem atingir milhares de actualizações por segundo. A caixa abaixo mostra um extracto dessas estatísticas extraídas do *site* <http://bgpupdates.potaroo.net>.

```
7 Day BGP Profile: 8-February-2017 00:00 - 14-February-2017 23:59 (UTC+1000)
```

```
Number of BGP Update Messages: 1952059
Number of Prefix Updates: 4607549
Number of Prefix Withdrawals: 262619
Average Prefixes per BGP Update: 2.49
Average BGP Update Messages per second: 2.82
Average Prefix Updates per second: 7.05
Peak BGP Update Message Rate per second: 2030 (07:51:56 Mon, 13-Feb-2017)
Peak Prefix Update Rate per second: 14538 (07:50:24 Mon, 13-Feb-2017)
Peak Prefix Withdraw Rate per second: 8498 (07:51:56 Mon, 13-Feb-2017)
Prefix Count: 671654
Updated Prefix Count: 244064
Stable Prefix Count: 427590
Origin AS Count: 56554
Updated Origin AS Count: 26035
Stable Origin AS Count: 30519
Unique Path Count: 291667
Updated Path Count: 170597
Stable Path Count: 121070
```

Por esta razão os ASs de grande dimensão fazem um investimento muito significativo nas suas redes de forma a darem-lhes estabilidade e também usam *border routers* de muito alta qualidade e robustez. Felizmente, os padrões reais mostram que a maioria dos ASs apresentam uma grande estabilidade e que a maioria das anomalias se concentram em relativamente poucos ASs, como se pode verificar no mesmo *site*, que contém o *ranking* dos ASs que enviam mais actualizações.

Caso a quebra de uma sessão BGP seja exclusivamente baseada na deteção da ausência de mensagens *keep-alive*, o BGP exige que haja 3 falhas sucessivas. Logo, a deteção da quebra da sessão pode levar bastantes segundos.

Para evitar sucessivas reconfigurações em cascata, o BGP, tal como normalizado em 2006 pelo RFC 4271 (BGP), impõe um limite mínimo de tempo entre o envio de duas actualizações seguidas sobre a mesma rota. Este *timer* designa-se por MRAI (*Minimum Route Advertisement Interval*), devia valer 30 segundos para o eBGP e 5 segundos para o iBGP, e devia também aplicar-se às simples supressões de uma rota. O racional por detrás deste limite era tentar que o BGP *speaker* receba outras actualizações expectáveis da rota e evite enviar actualizações que teria de anular logo a seguir. Mais à frente é ilustrada uma situação em que a convergência só se dá depois de várias trocas de mensagens, apesar de os prefixos da rota estarem inacessíveis logo desde o início.

Apesar de mais recentemente se considerar que esses valores são muito altos e que devem ser adaptados ao contexto de cada AS, o facto é que o valor deste período mínimo tem um impacto decisivo na convergência. Por outro lado, quando existe instabilidade numa rota, o RFC 4271 recomendava também a técnica *Route Flap Damping* (RFD), que resultava em deferir ou ignorar as actualizações da mesma rota durante um período de tempo ainda mais alargado, para compensar a instabilidade.

No entanto, algumas observações mais recentes vão também no sentido de considerar que este algoritmo é demasiado agressivo e por vezes tem como resultado atrasar inutilmente a convergência que chega a levar vários minutos, ver a Secção 18.4.

O problema da convergência do BGP continua a ter actualidade e estar sujeito a controvérsia, mas hoje em dia procura-se privilegiar a rapidez da convergência, em eventual detimento da instabilidade dos *routers* BGP. No final do capítulo são apresentadas algumas referências sobre este problema cuja discussão ultrapassa o âmbito deste livro.

No que se refere à convergência, vamos apenas chamar a atenção para o facto de que esta é diferente da dos protocolos de encaminhamento pelo melhor caminho que foram apresentados no Capítulo 16. Como o BGP memoriza obrigatoriamente rotas alternativas, isso pode ser usado para encontrar imediatamente uma alternativa e manter a conectividade.

No entanto, em certas circunstâncias isso apenas serve para alargar a convergência por um processo designado *path exploration*, como é ilustrado na Figura 18.18.

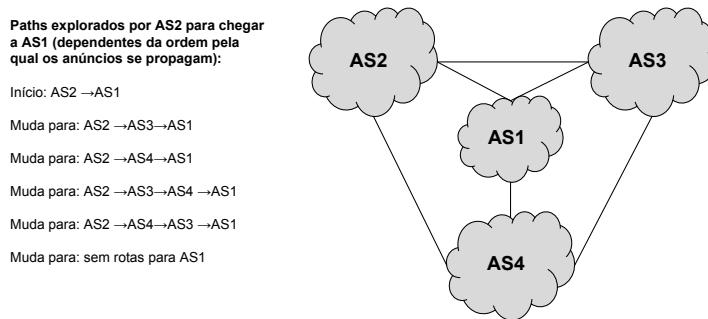


Figura 18.18: Instabilidade introduzida pela *path exploration* quando o AS1 deixou de estar acessível. Evolução dos caminhos usados por AS2 para chegar a AS1 até concluir que este deixou de estar acessível. Uma evolução semelhante passa-se também com o AS3 e o AS4.

O AS1 é cliente dos AS2, AS3 e AS4. Estes são *peers* de forma transitiva e anunciam AS *paths* para AS1 sem restrições. Inicialmente todos os ASs da figura têm conectividade para o AS1 e usam os canais directos que os ligam a AS1. No entanto, mais tarde o AS1 deixa de estar totalmente acessível e esses canais deixam de estar operacionais. Então todos os ASs mudam para caminhos alternativos. Como todos os *peers* fazem o mesmo, enviam actualizações das novas rotas usadas. No entanto, dependendo da ordem de recepção e tratamento das actualizações, a sequência de actualizações vistas pelo AS2 pode ser a que a figura ilustra visto que as relações de *peering* são transitivas. Este processo pode levar várias dezenas de segundos ou mesmo minutos.

Em conclusão, o encaminhamento com base em vectores de caminhos evita a anomalia designada *count to infinity*, ver o Capítulo 16, no entanto não deixa de prolongar a convergência de forma indesejável devido à exploração de diversas alternativas de caminhos até concluir que afinal nenhum está disponível.

### Interacção entre o BGP e o encaminhamento IGP

Quando os pacotes dirigidos aos prefixos de um AS chegam aos seus *border routers*, estes usam o IGP para os fazer chegar ao destino final. Com efeito estes *routers* executam simultaneamente os protocolos BGP e IGP. A FIB é comum mas as RIBs são distintas, apesar de interagirem.

No entanto, como se passa o processo inverso? Como é que os *routers* internos ao AS fazem chegar os pacotes dirigidos ao exterior do AS aos *border routers*?

Num *Stub* AS com uma única ligação ao exterior, o *border router* pode injectar no IGP uma rota por omissão (*default*), que é suficiente para o objectivo pretendido. Num AS *multi-homed*, com vários *border routers*, que o ligam a diversos provedores de acesso à Internet global, cada um deles pode injectar no IGP uma rota por defeito. No entanto, este esquema não é suficiente para que os pacotes dirigidos ao exterior sejam encaminhados para o *border router* mais próximo do destino.

Para que fosse possível introduzir alguma racionalidade no caminho escolhido internamente para os pacotes dirigidos ao exterior, seria necessário introduzir no IGP a totalidade das rotas adquiridas pelo eBGP num processo de encaminhamento que se designa por *default free*. É claro que isso exige *routers* internos de grande capacidade visto que a tabela global de prefixos contém centenas de milhar de prefixos. Tal pode revelar-se contraproducente e até inutilmente caro.

Existem diversas alternativas possíveis. Uma primeira passa por injectar no IGP apenas algumas rotas mais importantes, responsáveis pelo maioritário do tráfego para o exterior. Por exemplo todos os prefixos dos *peers* nacionais e dos fornecedores de conteúdos mais usados. Outra hipótese é usar um esquema hierárquico em que o conjunto dos *routers* do *backbone*, que inclui todos os *border routers*, são *default free*. Em contrapartida as diferentes zonas ligadas ao *backbone* usam rotas por omissão para chegarem ao mesmo. Naturalmente, os *backbones* dos AS de trânsito são todos necessariamente *default free*.

Esta discussão põe mais uma vez em evidência como a hierarquização do encaminhamento a todos os níveis é fundamental para a escalabilidade do mesmo. Por outro lado, ela também mostra que qualquer rede de dimensão razoável, e ligada à Internet por várias vias, usa simultaneamente vários métodos de encaminhamento interno e externo em cooperação.

O protocolo BGP possui diversos mecanismos que permitem a implementação de encaminhamento com base em políticas, nomeadamente: controlo da importação de rotas, manipulação de atributos das rotas, regras sofisticadas para comparação de rotas, controlo da exportação de rotas, etc.

Adicionalmente, o protocolo tem mecanismos para permitir aos diferentes *speakers* do mesmo AS conhecerem as rotas usadas pelos outros (*internal BGP* ou *iBGP*) e interage com os protocolos internos ao AS (os *Internal Gateways Protocols* ou *IGPs*).

### Encaminhamento BGP e segurança

Todos os protocolos de encaminhamento têm problemas de segurança dado que um “*router*” intruso tem hipóteses de sabotar o encaminhamento e atrair para si tráfego alheio, ou fazer ataques de negação de serviço. Na maioria das redes sob controlo de uma só organização, estes ataques são possíveis mas mais difíceis, pois os gestores da rede controlam todos os equipamentos que podem originar anúncios, assim como as respectivas vizinhanças. Por outro lado, é relativamente fácil parametrizar os equipamentos para só aceitarem anúncios de vizinhos legítimos.

O BGP executa na Internet global e quase sempre um *border router* tem sessões eBGP com *border routers* sob controlo de outras organizações. Sem desobedecer ao RFC 4271, um *router* pode ser configurado para anunciar um número de AS qualquer, e originar quaisquer prefixos, seja qual for a sua dimensão (e.g., /30). Se o vizinho não estiver parametrizado para filtrar este anúncios, vai aceitá-los como bons. Por isso muitos ASs de trânsito tomam especiais cuidados em só aceitarem de *Stub* ASs clientes

anúncios com os seus prefixos e número de AS, o que é facilmente implementado usando filtros de aceitação de rotas externas. Geralmente as informações necessárias fazem parte dos anexos técnicos dos contratos.

No entanto, quer porque esses controlos são caros do ponto de vista da gestão, quer porque conforme se sobe na hierarquia dos ASs de trânsito as regras de filtragem são cada vez mais complexas, e não existe nenhuma forma segura de as estabelecer de forma automática, na prática é fácil de introduzir falhas no controlo.

A maioria das RIR (*Regional Internet Registries*) mantém bases de dados de acesso público sobre os AS e as suas políticas. Estas bases de dados são acessíveis via o serviço *whois*. Infelizmente, muitas dessas bases de dados estão incompletas ou desactualizadas e não podem ser usadas para automatizar os filtros.

Esta situação proporciona a introdução de anúncios errados por acidente ou malícia. Por exemplo, em 1977 um AS de uma universidade introduziu milhares de anúncios de prefixos /24 errados por acidente, e atraiu para si o tráfego que lhes era dirigido provocando um “buraco negro” na sua conectividade que durou várias horas. Também é célebre o caso em que o AS da Telecom do Afeganistão introduziu um anúncio em que se apresentava como o originador do prefixo de servidores do YouTube, atraindo para si o tráfego que lhes era dirigido e impedindo o seu acesso normal.

A Figura 18.19 ilustra como tem lugar um ataque deste tipo. O atacante anuncia uma rota para um prefixo mais específico e, dada a regra *longest prefix matching*, atrai para si o tráfego correspondente a esse prefixo mais específico. No exemplo, um atacante no AS1 consegue atrair para o seu AS o tráfego do servidor de endereço IP 193.136.126.43 ligado ao AS2. O AS1 para esse efeito anuncia o prefixo 193.136.126.32/27, que por ser mais específico que o prefixo 193.136.0.0/15, atrai para si o tráfego dirigido ao servidor legítimo. Mesmo que não envie pacotes de resposta utilizando um endereço origem falso, conseguiria impedir desta forma o acesso ao servidor legítimo.

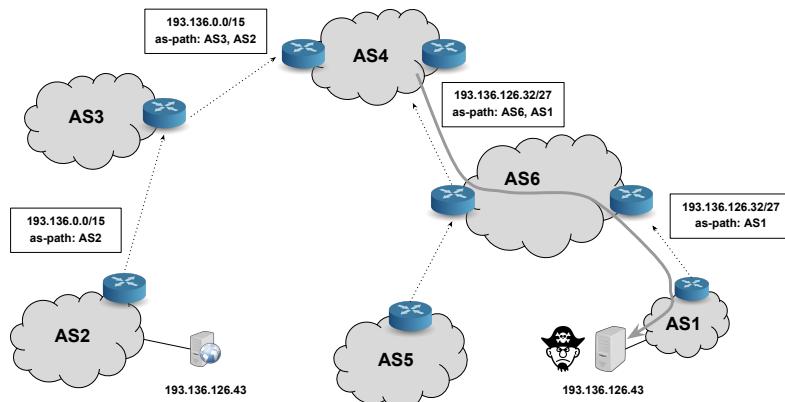


Figura 18.19: Ataque ao encaminhamento BGP através do anúncio pelo AS1 de uma rota para um prefixo mais estreito que não lhe pertence. A regra *longest prefix matching* desvia para o AS1 o tráfego dessa parte do prefixo legítimo anunciado pelo AS2.

Uma vez identificado que o ataque está a utilizar o protocolo BGP, é relativamente fácil perceber a origem do mesmo seguindo o AS path da rota. Esta faceta talvez explique porque razão estes ataques não são generalizados. Por outro lado, os operadores de trânsito que os permitissem de forma continuada, acabariam por ser

colocados em listas negras pelos outros operadores. Com efeito, se a generalidade dos operadores passassem a filtrar as rotas que contivessem o seu número de AS, isso levá-lo-ia à falência. Apesar da fragilidade óbvia do BGP, quando comparada com a sua responsabilidade em assegurar o funcionamento da Internet global, a verdade é que até ao momento em que este livro foi escrito não existem ataques ao BGP de forma continuada e generalizada.

Com efeito, a maioria dos *Stub ASs* são clientes de ASs que fornecem trânsito. Ora os prefixos IP anunciados por cada cliente são geralmente em pequeno número e a sua lista faz parte de contratos. Por isso a grande maioria dos ASs coloca filtros aos anúncios enviados pelos seus clientes, quer por razões de segurança, quer sobretudo por motivações comerciais. Por esta razão, a maioria dos ASs até não conseguem fazer anúncios falsos e existem motivações por parte dos seus fornecedores para o impedir. Adicionalmente, a percepção pelos ASs de *Tier-2* de que o risco de propagarem anúncios falsos, é superior aos potenciais lucros, é motivação suficiente para tentarem exercer controlo sobre os anúncios que recebem e propagam.

Por estas razões, os atacantes mais comuns preferem outros tipos de ataques mais subtils, focados em objectivos mais precisos e sem impacto generalizado sobre os operadores que os “apoiassem”. De qualquer forma, o problema persiste, e num quadro de fragmentação da Internet, ou de ciber guerra generalizada, os ataques via BGP serão mais realistas, e com efeitos potencialmente devastadores, pela instabilidade no encaminhamento global que introduziriam.

O BGP é um protocolo que propaga prefixos IP e rotas para os alcançar. Por defeito dois BGP *speakers* autenticam-se um ao outro. No entanto, se não for exercido controlo sobre os anúncios recebidos e propagados, é relativamente fácil introduzir instabilidade no encaminhamento global e impedir o acesso a regiões do espaço de endereçamento. O protocolo apenas define os mecanismos que permitem a introdução desses controlos. No entanto, não existem mecanismos que os tornem obrigatórios.

Apesar destas fragilidades, o quadro em que o protocolo opera minora a probabilidade de os controlos não serem exercidos e, por isso, não existem de forma frequente, na realidade actual, ataques conduzidos por esta via. Só num quadro de agressividade entre operadores ou blocos políticos esses ataques seriam mais prováveis.

Existem várias propostas de tentar tornar o BGP em geral mais seguro (*e.g.*, [Kent et al., 2000; Charles, 2000]). Infelizmente, as mais eficazes exigem requisitos de gestão e de capacidade computacional que, na ausência de incentivos claros, retarda a generalização da sua adopção.

### **Implementação de IP Anycasting**

Nos capítulos anteriores foi referida a utilização de IP Anycasting como uma forma de dirigir um pacote para um grupo de servidores, selecionando automaticamente o servidor mais próximo do emissor do pacote. Por exemplo, foi discutida esta utilização para endereçamento dos *root servers* do DNS, em que um único endereço IP representava um conjunto de diferentes servidores, localizados em diferentes ASs, ver a Secção 11.3. Foi também referida a utilização de IP Anycasting como forma de endereçar os *edge servers* de uma CDN mais próximos do emissor, ver a Secção 13.3.

A forma mais simples de implementar IP Anycasting consiste em associar um prefixo IP específico, e um AS do proprietário do serviço, a um conjunto dos endereços IP Anycasting, e anunciar esse prefixo e AS simultaneamente a partir de várias redes distintas. Cada um dos outros ASs vai seleccionar apenas um AS *path* para o prefixo

anunciado, naturalmente o que corresponder à sua política de encaminhamento. Desta forma, mediante a seleção realizada, a instância do serviço alcançada será a que estiver mais próxima (em termos das métricas de encaminhamento usadas) do emissor, em função das políticas seguidas pelos ASs atravessados. O mesmo esquema pode ser implementado dentro de um AS, usando um esquema equivalente com os protocolos de encaminhamento usados no interior do AS. No fundo a técnica é aplicável com qualquer protocolo de encaminhamento, ficando o âmbito do grupo anycasting subordinado ao âmbito dos anúncios realizados via o protocolo de encaminhamento usado.

Esta implementação do IP Anycast tem por repercussão o aumento da dimensão das tabelas de encaminhamento BGP mas não existem alternativas à mesma quando se pretende obter o seu efeito a nível global da Internet.

O anycasting funciona sem problemas em aplicações baseadas em UDP e com interações rápidas entre os clientes e os servidores, como é o caso do protocolo do DNS. No caso do TCP, se durante a conexão TCP variar o servidor em que a mesma termina, existirão problemas e a conexão não sobreviverá a essa alteração.

O anycasting é implementado realizando um conjunto de anúncios equivalentes, a partir de diferentes pontos da rede.

Esta técnica tem inúmeras vantagens, nomeadamente: 1) aumento da eficiência do encaminhamento, pois os pacotes dirigem-se para a instância do serviço mais próxima em termos das métricas de encaminhamento; 2) tolerância a falhas, pois se uma instância do serviço falha, desde que a instância do anúncio do prefixo IP deixe de existir, os novos clientes não se apercebem da falha; e 3) a técnica é muito útil para distribuir a carga.

Por estas razões, mesmo para aplicações baseadas no protocolo HTTP, o anycasting é usado em numerosas CDNs pois as suas vantagens são superiores aos seus potenciais defeitos. Com efeito, sendo o HTTP um protocolo predominantemente sem estado, tendo em consideração que muitas aplicações Web são implementadas usando inúmeras transações HTTP, geralmente de pequena duração, e finalmente usando técnicas de GETs HTTP parciais para a obtenção de objectos volumosos mas estáticos, é possível usar IP Anycasting com TCP de forma bastante eficaz. Esta prática está a generalizar-se entre os fornecedores de serviços de CDN para efeitos da aceleração a conteúdos estáticos e para fornecimento de serviços de segurança. A mesma é também usada pelos servidores *caching only* públicos (*e.g.*, Open DNS, Google DNS, ...).

## 18.4 Resumo e referências

### Resumo

Para aumentar a escala de um sistema, é frequente o mesmo ser partitionado em sub-componentes isoladas e introduzir alguma forma de hierarquia entre as mesmas. Este princípio, aplicado à Internet global, levou à sua subdivisão em sub-redes independentes, cada uma das quais é chamada um sistema autónomo.

Um **Sistema Autónomo (Autonomous System)** é uma rede ou conjunto de redes sob a mesma administração e cujo encaminhamento interno é opaco para o exterior. Todos os prefixes IP existentes na Internet pública pertencem necessariamente a um AS, ou estão incluídos num que pertence. Quando uma rede está ligada a diferentes ASs, constitui um AS específico e usa prefixes IP próprios.

De acordo com esta divisão da Internet em sub-redes, classificam-se os protocolos de encaminhamento em **protocolos IGP (Internal Gateway Protocols)**, os protocolos usados dentro dos ASs para encaminhamento intra-AS, e **protocolos EGP**

(***External Gateway Protocols***) para encaminhamento inter-AS. O encaminhamento inter-AS é assegurado pelo protocolo **BGP** (*Border Gateway Protocol*) que é geralmente executado pelos *routers* de fronteira ou *border routers*, os quais são responsáveis pela interligação dos ASs.

Um anúncio ou rota BGP é constituída por uma lista de prefixos e vários atributos comuns aos mesmos. Entre estes figura um caminho expresso como uma lista dos ASs, chamado *AS path*, que o anunciante usa para chegar ao AS onde residem os prefixos. Uma rota contém também um atributo com o endereço do *border router* de recepção do tráfego que é dirigido aos seus prefixos. Os *AS paths* permitem saber exactamente por que ASs passarão os pacotes, o que facilita a tomada de decisão em função do caminho proposto e permite suprimir a introdução de ciclos de encaminhamento. Dado os anúncios conterem caminhos, o BGP é caracterizado como um protocolo com base em **prefixos e vectores de caminhos** (*prefix-based path-vector protocol*).

Na Internet global o grafo de ligação entre os diferentes ASs é conhecido através dos prefixos IP e dos *AS paths* escolhidos e anunciado pelos ASs uns aos outros. A partir de *routers* da DFZ (*default free zone*) podem observar-se muitos desses *AS paths*, o que permite analisar as relações existentes entre os diferentes ASs e verificar que esses caminhos estão subordinados a relações comerciais ou de colaboração.

O grafo dos ASs é de grande dimensão pois tem cerca de 55.000 nós e muitos mais milhares de arcos. O número de ligações dos ASs é muito heterogénea e com uma distribuição muito desigual. Esta distribuição permite concluir que o conjunto dos ASs *Tier 1* é responsável directa ou indirectamente pela conectividade entre praticamente todos os ASs existentes.

O BGP é um protocolo complexo que assegura a conectividade da Internet global e que tem como requisitos ser capaz de lidar com centenas de milhar de prefixos IP e várias dezenas de milhar de redes interligadas, cada uma das quais com políticas específicas de encaminhamento. Apesar desta escala, pretende-se que convirja rapidamente.

O protocolo é executado por *routers* especiais, ditos **BGP speakers**. Estes *routers* coincidem geralmente com *border routers*, os *routers* que asseguram a interligação dos AS. O diálogo entre os BGP speakers é assegurado através de **sessões BGP** (*BGP sessions*) que são ponto-a-ponto e utilizam conexões TCP. Os anúncios são incrementais e só comunicam actualizações (novas rotas ou supressão de rotas).

Como os caminhos em BGP são definidos como listas de ASs, é possível estabelecer regras para suporte de opções de encaminhamento com base em políticas e relações contratuais. O protocolo possui diversos mecanismos que permitem a implementação dessas políticas, nomeadamente: controlo da importação de rotas, manipulação de atributos das rotas e regras sofisticadas para comparação de rotas, controlo da exportação de rotas, etc. Adicionalmente o protocolo tem mecanismos para permitir aos diferentes speakers do mesmo AS conhecerem as rotas usadas pelos outros (*internal BGP* ou *iBGP*) e interage com outros protocolos IGP do AS.

Alguns dos termos introduzidos no capítulo são a seguir passados em revista. Entre parêntesis figura a tradução mais comum em língua inglesa.

## Principais conceitos

**Sistema Autónomo - AS** (*Autonomous System*) Um AS é uma rede ou conjunto de redes sob a mesma administração e cujo encaminhamento interno é opaco para o exterior. A Internet está partitionada em alguns milhares de ASs. Cada AS corresponde grosso modo a um operador ou a uma organização.

**AS border router** Os *routers* responsáveis pela ligação dos ASs uns aos outros designam-se por *border routers*.

**Protocolos IGP** (*Internal Gateway Protocols*) Protocolos de encaminhamento usados internamente a um AS, ou protocolos intra-AS.

**Protocolos EGP** (*External Gateway Protocols*) Protocolos de encaminhamento usados externamente aos ASs, ou protocolos entre ASs ou inter-AS.

**Protocolo BGP** (*Border Gateway Protocol*) É o único protocolo especialmente concebido para encaminhamento inter-AS.

**BGP speaker** Os *routers* que executam o protocolo BGP designam-se BGP *speakers* e geralmente, mas não obrigatoriamente, coincidem com os *border routers*.

**Caminho de ASs** (*AS path*) Uma sequência de números de ASs que denota um caminho sem ciclos entre ASs na Internet global.

**Rota BGP** (*BGP route*) Um anúncio ou rota BGP é constituída por uma lista de prefixos IP e vários atributos entre os quais um *AS path*, que o anuncianta usa para chegar ao AS onde residem os prefixos. Uma rota contém também um atributo com o endereço do *border router* de recepção do tráfego que é dirigido aos seus prefixos.

**Default Free Router** Um BGP *speaker* ou outro *router* que não possui rotas por omissão (*default*) e conhece todos os prefixos IP reconhecidos na Internet global, diz-se um *Default Free Router*.

**Sessão BGP** (*BGP session*). Um canal virtual estabelecido entre dois BGP *speakers* chama-se uma BGP *session*. Estas sessões são ponto-a-ponto, e são mantidas sobre uma conexão TCP pelo que não exigem a adjacência por um canal físico entre os dois *speakers*. Os anúncios BGP que circulam nas sessões são incrementais e só comunicam actualizações (novas rotas ou supressão de rotas).

**Protocolo iBGP** (*Internal BGP*) Quando um AS tem mais do que um BGP *speaker* estes têm de conhecer as rotas externas seleccionadas uns pelos outros. A versão do protocolo executada entre os BGP *speakers* do mesmo AS chama-se iBGP.

**Protocolo eBGP** (*External BGP*) É a versão do BGP executada entre *speakers* de diferentes ASs.

**Filtros de rotas BGP** Para implementar as políticas de encaminhamento de um AS este usa filtros sobre as rotas importadas de outros ASs (*BGP import filters*) e filtros sobre as rotas exportadas para os outros ASs (*BGP export filters*). Os filtros mais comuns são só aceitar rotas de clientes para os prefixos próprios de cada cliente, e só exportar rotas para outros ASs sobre prefixos para os quais se aceita encaminhar tráfego.

**Seleção de rotas** Os *routers* BGP têm de selecionar a rota que vão usar para atingir cada prefixo. Para esse efeito compararam diversos atributos de todas as rotas para o prefixo que importam dos seus vizinhos. Uma das regras mais simples consiste em selecionar a rota com o *AS path* mais curto.

## Referências

O livro [Huitema, 1995] é uma referência clássica que apresenta uma introdução ao BGP e discute as suas bases algorítmicas. Como o BGP é bastante complexo, normalmente raramente é tratado de forma muito completa em livros genéricos sobre redes de computadores. É necessário recorrer a livros especializados como por exemplo [Doyle and Carroll, 2017] que é considerado um dos melhores, e que inclui também indicações concretas sobre como o parametrizar em comutadores de pacotes populares.

No site da Caida, ver abaixo, encontram-se muitas referências bibliográficas e dados sobre a estrutura interna da Internet global recolhida de várias fontes entre as quais de tabelas de rotas BGP importadas de vários *routers* da DFZ. O *survey* [Motamedi et al., 2015] apresenta uma panorâmica das técnicas usadas para fazer levantamentos da topologia da Internet em termos de ASs.

O problema da convergência do BGP é um tema actual de investigação. O artigo [Gill et al., 2013] permite uma primeira incursão no mesmo. Tem sido conduzida uma

investigação intensiva sobre os problemas de segurança do BGP. [Butler et al., 2010; Huston et al., 2011] apresentam dois excelentes *surveys* sobre o tema, incluindo boas introduções ao BGP. No *site* da Internet Society, ver abaixo, encontram-se também muitas indicações sobre este tema.

#### Apontadores para informação na Web

- <http://www.icann.org> – É o *site* oficial da ICANN com informação como são atribuídos os números de ASs.
- <http://bgp.potaroo.net> – É o *site* que mantém um relatório semanal actualizado do funcionamento do BGP na Internet global. Contém imensas estatísticas sobre a dinâmica do protocolo.
- <http://www.cidr-report.org> – É um *site* que mantém informação actualizada sobre os prefixos IP visíveis na Internet global e sobre a sua dinâmica de agregação e desagregação.
- <http://as-rank.caida.org> – É o *site* da Caida que mantém informação actualizada sobre todos os ASs existentes na Internet.
- <http://as-rank.caida.org/?mode0=as-ranking&n=10&ranksort=2> – É o *site* da Caida que mantém informação actualizada sobre o *ranking* dos ASs.
- <http://www.internetsociety.org/deploy360/securing-bgp/> – Este *site*, da responsabilidade da Internet Society, contém muita informação, artigos e recomendações sobre o problema da segurança do BGP.
- <http://ietf.org/rfc.html> – É o repositório oficial dos RFCs, nomeadamente dos citados neste capítulo.
- As diversas RIRs (*Regional Internet Registries*) mantêm um serviço *whois* sobre os diferentes ASs da sua região, assim como sobre os prefixos por estes exportados. Por exemplo, os comandos abaixo ilustram como obter informações do RIPE:

```
whois -h whois.ripe.net 1930
whois -h whois.ripe.net 193.136.0.0/15
```

## 18.5 Questões para revisão e estudo

1. Verdade ou mentira? Justifique a sua resposta.
  - (a) Quando no protocolo OSPF se introduzem zonas, estas são sub-redes que para todos os efeitos são equivalentes a um sistema autónomo (AS).
  - (b) O *backbone* NSFNET foi e é hoje em dia o AS número 1.
  - (c) O protocolo BGP é um protocolo de encaminhamento interno.
  - (d) O protocolo BGP é um protocolo de encaminhamento externo.
  - (e) O protocolo BGP é do tipo *link-state*.
  - (f) O protocolo BGP é do tipo *distance vector*.
  - (g) Um AS *multi-homed* assegura trânsito pago aos seus fornecedores de conectividade para a Internet global.
  - (h) Um *border router* de um AS escolhe sempre o caminho mais curto para alcançar um destino externo.

- (i) Um AS exporta via BGP uma rota para o prefixo IP  $p$ . Sempre que se produz uma alteração de estado na rede interna desse AS, que altera o custo com que  $p$  é alcançado internamente, essa alteração tem de desencadear uma actualização da rota para  $p$  via o BGP.
- (j) Quando um AS exporta uma rota, nem sempre acrescenta o seu número de AS ao AS *path* da mesma.
- (k) Um BGP *speaker* quando exporta um prefixo, exporta todas as rotas que conhece para o mesmo.
- (l) Um BGP *speaker* só exporta uma rota para um prefixo IP se assegurar encaminhamento para o mesmo.
- (m) Se um AS fizer encaminhamento interno para um dado prefixo, então necessariamente exporta uma rota para o mesmo.
- (n) O protocolo OSPF é usado para fazer chegar automaticamente aos BGP *speakers* os prefixos IP que o AS deve exportar através de rotas.
- (o) Como o encaminhamento interno de um AS deve ser opaco para o exterior, os prefixos IP públicos do AS não devem ser divulgados em rotas exportadas pelo BGP.
2. Verdadeiro ou falso? Justifique a sua resposta.
- (a) Dentro de um *Stub* AS apenas circulam pacotes com origem ou destino aos prefixos IP que lhe pertencem.
- (b) Dentro de um *Stub* AS circulam pacotes com destino aos prefixos IP que lhe pertencem.
- (c) De dentro para fora de um *Stub* AS circulam pacotes com origem nos prefixos do AS.
- (d) De fora para dentro de um *Stub* AS circulam pacotes com origem nos prefixos do AS.
- (e) Dentro de um AS de trânsito circulam pacotes com origem ou destino nos prefixos IP que lhe pertencem.
- (f) Dentro de um AS de trânsito apenas circulam pacotes com origem ou destino nos prefixos IP para os quais o AS anuncia rotas.
- (g) Dentro de um AS de trânsito circulam pacotes com destino aos prefixos IP para os quais o AS anuncia rotas.
- (h) Dentro de um AS de trânsito circulam pacotes com origem nos prefixos IP para os quais o AS anuncia rotas.
- (i) Dentro de um AS de trânsito circulam pacotes com origem nos prefixos IP dos seus clientes.
3. A maioria dos protocolos de encaminhamento IGP usam UDP ou equivalente para passarem informação de encaminhamento entre comutadores vizinhos. No entanto, o BGP usa TCP. Apresente algumas das razões que justificam esta opção.
4. Descreva o processo de decisão usado pelo BGP para comparar rotas alternativas para o mesmo destino.
5. Descreva os mecanismos usados pelo BGP para implementar decisões de encaminhamento de carácter político, *i.e.*, independentes de quaisquer métricas de custo.
6. Este exercício e o seguinte estão propostos em [Peterson and Davies, 2012]. Seja  $a$  o número de ASs na Internet e  $d$  (de diâmetro) o tamanho máximo de um AS *path*.

- (a) Descreva um modelo de conectividade entre ASs tal que  $d = O(\log a)$ .  
 (b) Descreva um modelo de conectividade entre ASs tal que  $d = O(\sqrt{a})$ .
7. Admitindo que na Internet existem  $n$  prefixos, dê o valor aproximado da dimensão da informação recebida de um vizinho por um BGP *speaker* para que o mesmo preencha completamente a sua FIB nos seguintes modelos de conectividade.
- (a) Modelo de conectividade entre ASs tal que  $d = O(\log a)$ .  
 (b) Modelo de conectividade entre ASs tal que  $d = O(\sqrt{a})$ .
8. A Figura 18.20 apresenta um conjunto de ASs com as ligações indicadas. Cada  $AS_i$  é simultaneamente o identificador de um AS e o identificador do prefixo por este anunciado. Os anúncios das rotas para os prefixos devem seguir a seguinte notação: [prefixo, [AS path]]. O  $AS_2$  é um operador de trânsito, enquanto que todos os outros são apenas clientes. As ligações oblíquas são de *peering* directo entre cada um dos ASs clientes que lhes estão ligados e apenas para os respetivos prefixos IP.

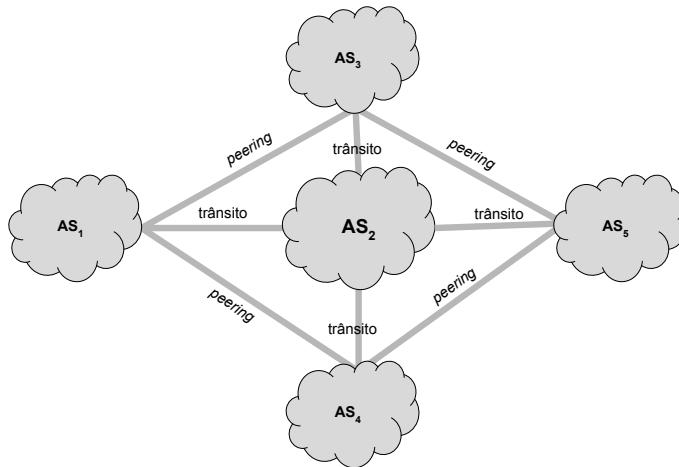


Figura 18.20: Rede do exercício 8

- (a) Quantas rotas pode o  $AS_1$  explorar para chegar ao prefixo do  $AS_5$  e qual a que prefere?  
 (b) Quantas rotas pode o  $AS_1$  explorar para chegar ao prefixo do  $AS_3$  e qual a que prefere?  
 (c) Caso o canal que liga o  $AS_2$  ao  $AS_5$  tivesse uma avaria e ficasse indisponível, como anunciaría o  $AS_2$  tal evento ao  $AS_1$ ? Quantas rotas poderia então o  $AS_1$  explorar para chegar ao prefixos do  $AS_5$ ?  
 (d) Admita agora que o  $AS_3$  pretende dar trânsito ao  $AS_1$  mas só para o  $AS_5$ . Como é que isso poderia ser implementado e que critérios deveria usar o  $AS_1$  para escolher entre as duas possíveis rotas que teria para chegar ao  $AS_5$  quando a rede estiver com todos os canais disponíveis?

9. A Figura 18.21 apresenta um conjunto de ASs interligados com as ligações indicadas. Cada  $AS_i$  é simultaneamente o identificador de um AS e o identificador do prefixo por este anunciado. Os anúncios das rotas para os prefixos devem seguir a seguinte notação: [prefixo, [AS path] ]. Os sistemas autónomos  $AS_A$  a  $AS_D$  representam redes de trânsito que fazem *peering* transitivo entre si, i.e., , conhecem todas as redes clientes uns dos outros. Os  $AS_x$  a  $AS_k$  são clientes. Cada um dos clientes anuncia o seu prefixos para os seus fornecedores de trânsito e recebem destes o anúncio de todos os prefixos alcançáveis na rede. Os ASs  $AS_x$  e  $AS_y$  têm um canal directo entre si e estabeleceram um contrato entre os dois para que apenas o tráfego de e para cada um dos dois passe diretamente nesse canal (acordo de *peering* directo). Adicionalmente, o  $AS_x$  paga ao  $AS_y$  para que este lhe dê conectividade para o resto da rede apenas no caso em que o canal de  $AS_x$  para  $AS_B$  for abaixo.

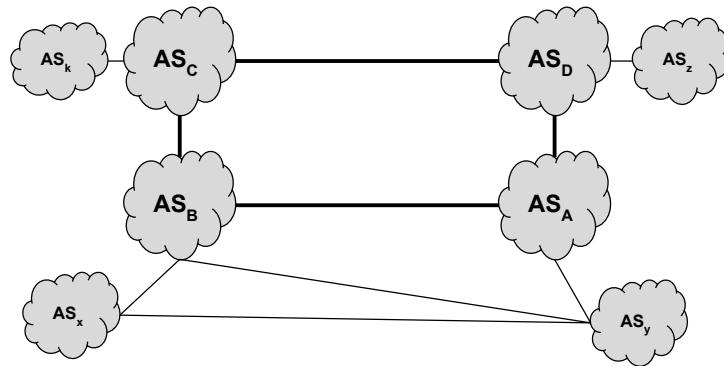


Figura 18.21: Rede do exercício 9

Indique os anúncios BGP enviados:

- de  $AS_x$  para  $AS_B$
  - de  $AS_x$  para  $AS_y$
  - de  $AS_y$  para  $AS_A$
  - de  $AS_y$  para  $AS_B$
  - de  $AS_A$  para  $AS_y$
  - de  $AS_B$  para  $AS_y$
  - de  $AS_B$  para  $AS_x$
  - de  $AS_y$  para  $AS_x$
10. A Figura 18.22 apresenta um conjunto de ASs interligados com as ligações indicadas. Cada  $AS_i$  é simultaneamente o identificador de um AS e o identificador do prefixo por este anunciado. Os anúncios das rotas para os prefixos devem seguir a seguinte notação: [prefixo, [AS path] ]. Por exemplo,  $AS_3$  anuncia ao  $AS_5$  o caminho que usa para chegar a  $AS_8$  através do seguinte anúncio: [ $AS_8$ , [ $AS_3, AS_2, AS_8$ ]]. Não existem restrições de uso de caminhos devido a políticas comerciais e todos os ASs escolhem o caminho que atravessa um menor número de ASs. Caso um AS congeça duas rotas distintas para o mesmo prefixo e com o mesmo tamanho, prefere o caminho cujo primeiro AS tenha o identificador mais baixo (preferiria  $AS_8$  a  $AS_9$  por exemplo). Os anúncios BGP possíveis são: `update` e `withdraw`.

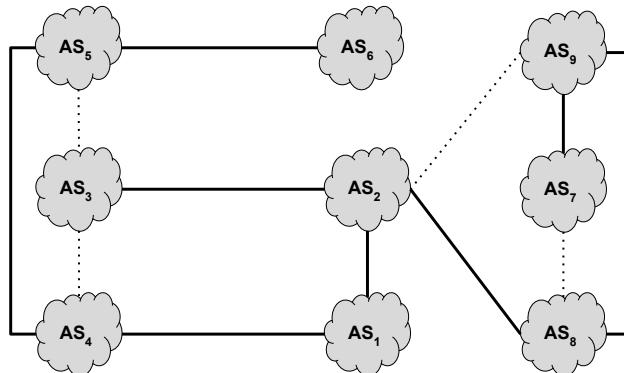


Figura 18.22: Rede do exercício 10

- (a) Indique para cada um dos ASs todas as rotas para o prefixo  $AS_9$  que cada um deles conhece.
- (b) O canal que liga  $AS_2$  a  $AS_9$  foi abaixo. Que mensagens BGP enviou  $AS_2$  aos seus vizinhos.
- (c) O canal que liga  $AS_2$  a  $AS_9$  foi abaixo no momento  $t$  e  $AS_2$  detectou esse evento imediatamente. Durante quanto tempo  $AS_2$  não conseguiu enviar pacotes destinados a  $AS_9$  sabendo que o valor do *timer MRAI* é de 30 segundos? Justifique a sua resposta.
- (d) Considere agora que os diferentes ASs têm uma relação comercial entre eles de tal forma que os canais a negrito entre dois ASs constituem uma relação cliente / fornecedor em que o fornecedor é o AS com identificador inferior. Por exemplo, o canal entre o  $AS_1$  e o  $AS_2$  é um canal pelo qual o  $AS_2$  paga a  $AS_1$  para ter conectividade para  $AS_4$ ,  $AS_5$  e  $AS_6$  (note que  $AS_3$  e  $AS_8$  são clientes de  $AS_2$ ). Os canais a tracejado são canais de *peering* que, por hipótese, só podem ser usados para comunicar directamente entre os dois ASs a eles ligados. Por exemplo, o canal entre  $AS_2$  e  $AS_9$  só pode ser usado para  $AS_2$  comunicar com  $AS_9$  e vice-versa e o canal entre  $AS_7$  e  $AS_8$  também só pode ser usado para  $AS_7$  e  $AS_8$  comunicarem directamente. Neste quadro indique todos os caminhos anunciados por  $AS_2$  aos seus vizinhos.
11. Este exercício está proposto em [Marsic, 2013]. A Figura 18.21 apresenta um conjunto de ASs interligados com as ligações indicadas. Cada  $AS_i$  é simultaneamente o identificador de um AS e o identificador do prefixo por este anunciado. Os ASs  $AS_A$  e  $AS_B$  são *peers* e ambos compram trânsito a  $AS_C$ , o qual assegura a ligação ao resto da Internet. Todos os ASs usam *hot potato routing*. Os anúncios realizados são os compatíveis com as relações comerciais existentes entre os ASs. Os ASs que são *peers* não fornecem *backup* um ao outro. Todos os AS paths entre ASs vizinhos diretos não contém ocorrências repetidas de ASs. Os diferentes ASs usam todos o mesmo IGP e todos os canais internos a um AS têm custo idêntico nesse IGP.
- (a) De quantos e quais AS paths dispõem o computador  $y$  para chegar ao computador  $x$ ?
- (b) Porque routers passam os pacotes de  $y$  para  $x$  e vice-versa?

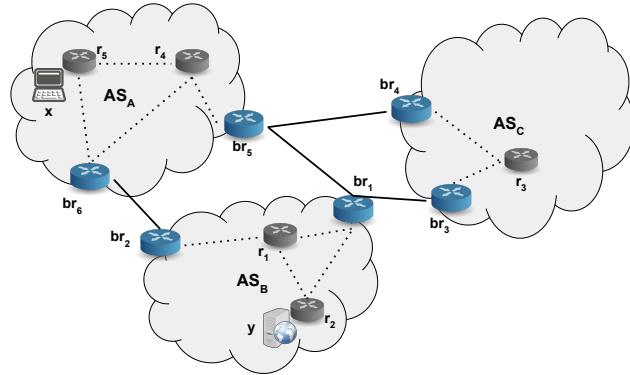


Figura 18.23: Rede do exercício 11

- (c) Em que situação esses dois caminhos poderiam coincidir?
- (d) Em que circunstâncias os pacotes de  $y$  para  $x$  atravessam o AS  $AS_C$ ?
- (e) Como se assegura que o tráfego de  $AS_A$  para o resto da Internet não vá via o  $AS_B$  caso a ligação de  $AS_A$  a  $AS_C$  ficar inoperacional?
- (f) Admita que se pretende que de facto  $AS_A$  e  $AS_B$  dêem *backup* mútuo na ligação à Internet um do outro. Como se pode atingir esse objectivo?
12. Na maioria dos *routers* é possível introduzir rotas estáticas manualmente e fazer com que os anúncios dos protocolos IGP os incluem. Este tipo de mecanismos é frequentemente usado para anunciar uma rota por omissão (*default route*). As perguntas a seguir devem ser respondidas no contexto da rede de uma instituição com um *backbone IP* próprio, com vários *routers*, em várias localidades, e uma ou mais ligações à Internet. O IGP usado é o OSPF. O EGP é o BGP mas a comunicação entre os dois só se faz usando rotas estáticas, *i.e.*, colocadas manualmente pelos gestores.
- Suponha que a rede só tem uma ligação à Internet. O *router* que a assegura tem uma rota estática, associada à rota por omissão (*default*), que aponta para a interface de ligação ao exterior do *border router*. O OSPF permite dar manualmente um custo a esta rota. Essa rota é normalmente anunciada a todos os outros *routers* da rede interna. Quando é que os outros *routers* usam a rota por omissão assim anunciada?
  - Suponha que a interface de ligação à Internet referida na alínea anterior vai abaixar. Que alterações têm lugar nas FIBs da rede interna, como e com que velocidade?
  - Suponha agora que há várias ligações à Internet. Estando todas activas, como se passa a distribuição de carga entre elas?
  - Se no contexto da alínea anterior uma das interfaces de ligação externa for abaixar, que alterações têm lugar nas FIBs dos *routers* da rede interna e como?
13. A rede  $u$  está ligada à rede  $t$  (operador de trânsito de  $u$ ) para ter conectividade Internet. A rede  $u$  tem o prefixo  $100.100.100.0/24$  e a rede  $t$  tem o prefixo  $100.100.0.0/16$ . Como a rede  $u$  só tem um fornecedor externo, os seus gestores parametrizaram-na de tal forma que o *router*  $r_1$  que a liga à rede  $t$  injecta uma

rota estática por omissão ( $0.0.0.0/0$ ) no seu IGP. A rede  $t$  apenas anuncia por BGP na Internet global o seu prefixo ( $100.100.0.0/16$ ). Com efeito, a rede  $u$  não tem outras ligações ao mundo e é vista como uma sub-rede da rede  $t$  sem AS próprio.

Mais tarde, devido a um projecto de colaboração, a rede  $u$  ligou-se também à rede  $isp$ , com o prefixo  $200.200.200.0/24$ , através do router  $r_2$ . O objectivo é permitir que os utilizadores da rede  $u$  acedam via esse router à rede  $200.200.200.0/24$ , e vice versa, sem necessidade de passar pela Internet global. Os outros ASs não sabem desta “ligação directa”.

(a) Para fazer esta ligação a rede  $u$  e a rede  $isp$  têm de usar o mesmo protocolo de encaminhamento IGP nas suas redes internas (RIP, OSPF, ...) e têm de estabelecer uma ligação BGP entre si? Justifique.

(b) Que rota deve anunciar estaticamente o router  $r_2$  para dentro da rede  $u$  e que rota deve anunciar estaticamente o router dessa ligação na rede  $isp$ ? Justifique.

(c) Mais tarde a rede  $u$  resolveu que a sua ligação à rede  $isp$  passaria a dar-lhe também trânsito para a Internet, mas ao mesmo tempo pretendia continuar com a ligação à Internet via a rede  $t$ . Para esse efeito arranjou um número de AS, o AS- $u$ , e parametrizou os routers  $r_1$  e  $r_2$  para passarem a usar BGP.

Os routers  $r_1$  e  $r_2$  passaram a anunciar o AS Path [ $100.100.100.0/24$ , [AS- $u$ ]] por BGP para os outros operadores. A rede  $u$  arranjou também forma de os routers internos dividirem a carga do tráfego de saída entre si (dentro da rede  $u$  ambos os routers anunciam uma *default route*). A rede  $t$  continuou a anunciar apenas o seu prefixo via a rota [ $100.100.0.0/16$ , [AS- $t$ ]] e a rede  $isp$  passou a anunciar para a Internet as rotas [ $100.100.100.0/24$ , [AS- $isp$ , AS- $u$ ]] e [ $200.200.0.0/16$ , [AS- $isp$ ]]. Por que rede ou redes passou a entrar o tráfego da Internet dirigido à rede  $u$ ? Justifique.

(d) Como poderia a rede  $u$  conseguir que o tráfego vindo da Internet para o seu prefixo passasse a usar os dois fornecedores? Justifique.

(e) Qual a forma mais simples que tem a rede  $u$  de impedir que o tráfego vindo da rede  $t$  e dirigido a clientes da rede  $isp$  atravesssem a sua rede e vice-versa? Justifique.

(f) Que meios poderiam usar os gestores da rede  $u$  para afinar a distribuição de carga por esses dois fornecedores quer em entrada, quer em saída?

14. Inspire-se do protocolo BGP para recomendar uma nova versão do protocolo RIP que permita ao mesmo ser usado em redes de muito maior escala e anule o seu principal defeito ligado a um tempo de convergência demasiado elevado.

15. Execute os comandos:

```
whois -h whois.ripe.net AS1930
whois -h whois.ripe.net 193.136.0.0/15
```

e interprete os seus resultados, nomeadamente sob o ponto de vista das relações e políticas de importação e exportação de rotas dos ASs referidos.

16. Através do estudo da bibliografia sugerida, apresente um resumo das principais fragilidades do protocolo BGP do ponto de vista da segurança.

17. Através do estudo da bibliografia sugerida, apresente um resumo das principais propostas para colmatar as falhas de segurança do protocolo BGP.



---

## Bibliografia

---

- Abboud, O., Pussep, K., Kovacevic, A., Mohr, K., Kaune, S., and Steinmetz, R. (2011). Enabling resilient p2p video streaming: survey and analysis. *Multimedia Systems*, 17(3):177–197.
- Abed, G. A., Ismail, M., and Jumari, K. (2011). A survey on performance of congestion control mechanisms for standard tcp versions. *Australian Journal of Basic and Applied Sciences*, 5(12).
- Afanasyev, A., Tilley, N., Reiher, P., and Kleinrock, L. (2010). Host-to-host congestion control for tcp. *Communications Surveys Tutorials, IEEE*, 12(3):304–342.
- Akamai Technologies, I. (2016). The akamai state of the internet report - first quarter of 2016. Available on-line from the company site.
- Allen, K. S. (2012). *What Every Web Developer Should Know About HTTP*. OdeToCode LLC, Kindle Edition, 1st edition.
- Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. (2001). Resilient overlay networks. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, SOSP '01, pages 131–145, New York, NY, USA. ACM.
- Balakrishnan, H., Seshan, S., Amir, E., and Katz, R. H. (1995). Improving tcp/ip performance over wireless networks. In *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking*, MobiCom '95, pages 2–11, New York, NY, USA. ACM.
- Beard, C. and Stallings, W. (2015). *Wireless Communications and Systems*. Pearson Education.
- Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition.
- Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. (1994). The world-wide web. *Commun. ACM*, 37(8):76–82.
- Berners-Lee, T., Hall, W., Hendler, J., Shadbolt, N., and Weitzner, D. J. (2006). Creating a science of the web. *Science*, 313(5788):769–771.
- Birrell, A. D., Levin, R., Schroeder, M. D., and Needham, R. M. (1982). Grapevine: An exercise in distributed computing. *Commun. ACM*, 25(4):260–274.

- Birrell, A. D. and Nelson, B. J. (1984). Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)*, 2(1):39–59.
- Board, I. A. (2000). IAB Technical Comment on the Unique DNS Root. RFC 2826 (Informational).
- Boudec, J.-Y. (2014). Rate adaptation, congestion control and fairness: A tutorial. Available on-line from Ecole Polytechnique Fédérale de Lausanne (EPFL).
- Brakmo, L. S., O’Malley, S. W., and Peterson, L. L. (1994). Tcp vegas: New techniques for congestion detection and avoidance. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications, SIGCOMM ’94*, pages 24–35, New York, NY, USA. ACM.
- Budzisz, L., Garcia, J., Brunstrom, A., and Ferrús, R. (2012). A taxonomy and survey of sctp research. *ACM Comput. Surv.*, 44(4):18:1–18:36.
- Butler, K., Farley, T. R., McDaniel, P., and Rexford, J. (2010). A survey of bgp security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122.
- Calvert, K. and Donahoo, M. (2011). *TCP/IP Sockets in Java: Practical Guide for Programmers*. The Practical Guides. Elsevier Science.
- Cerf, V. and Kahn, R. (1974). A protocol for packet network intercommunication. *Communications, IEEE Transactions on*, 22(5):637–648.
- Chadwick, D. (1994). *Understanding X.500: The Directory*. Chapman & Hall, Ltd., London, UK, UK.
- Charles, S. K. (2000). Secure border gateway protocol (s-bgp)—real world performance and deployment issues. *Proceedings of the NDSS2000*.
- Chen, F., Sitaraman, R. K., and Torres, M. (2015). End-user mapping: Next generation request routing for content delivery. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM ’15*, pages 167–181, New York, NY, USA. ACM.
- Chen, X., Zhai, H., Wang, J., and Fang, Y. (2005). A survey on improving tcp performance over wireless networks. In *In Resource Management in Wireless Networking, Cardei M, Cardei I, Du D-Z (eds*, pages 657–695. Kluwer Academic Publishers.
- Chiu, D. M. and Jain, R. (1989). Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14.
- Chu, Y.-h., Rao, S. G., and Zhang, H. (2000). A case for end system multicast (keynote address). In *Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS ’00*, pages 1–12, New York, NY, USA. ACM.
- Clark, D. (1988). The design philosophy of the darpa internet protocols. In *SIGCOMM ’88: Symposium proceedings on Communications architectures and protocols*, pages 106–114, New York, NY, USA. ACM.
- Clark, D. D. and Tennenhouse, D. L. (1990). Architectural considerations for a new generation of protocols. In *Proceedings of the ACM Symposium on Communications Architectures & Protocols, SIGCOMM ’90*, pages 200–208, New York, NY, USA. ACM.

- Cohen, B. (2003). Incentives build robustness in BitTorrent. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*.
- Comer, D. E. and Stevens, D. L. (1997). *Internet With TCP/IP – Volume III – Client-Server Programming and Applications*. Prentice-Hall.
- Couch, L. (2000). *Digital and Analog Communication Systems*. Prentice-Hall.
- Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., and Bailey, M. (2014). Measuring ipv6 adoption. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 87–98, New York, NY, USA. ACM.
- Dai, J., Liu, F., and Li, B. (2010). The disparity between p2p overlays and isp underlays: issues, existing solutions, and challenges. *Netwkrk. Mag. of Global Internetwkg.*, 24:36–41.
- David, E. and Jon, K. (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA.
- Day, J. (2008). *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall.
- Dijkstra, E. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, 1(269-270):6.
- Dobrescu, M., Egi, N., Argyraki, K., Chun, B.-G., Fall, K., Iannaccone, G., Knies, A., Manesh, M., and Ratnasamy, S. (2009). Routebricks: Exploiting parallelism to scale software routers. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 15–28, New York, NY, USA. ACM.
- Donnet, B., Gueye, B., and Kaafar, M. A. (2010). A survey on network coordinates systems, design, and security. *IEEE Communications Surveys Tutorials*, 12(4):488–503.
- Doyle, J. and Carroll, J. (2006). *Routing TCP/IP, Volume 1, 2nd edition*. Pearson Education and Cisco Press.
- Doyle, J. and Carroll, J. (2017). *Routing TCP/IP, Volume 2, 2nd edition*. Pearson Education and Cisco Press.
- Dreibholz, T., Rathgeb, E., Rungeler, I., Seggelmann, R., Tuxen, M., and Stewart, R. (2011). Stream control transmission protocol: Past, current, and future standardization activities. *Communications Magazine, IEEE*, 49(4):82–88.
- Feldmann, A. (2007). Internet clean-slate design: What and why? *SIGCOMM Comput. Commun. Rev.*, 37(3):59–64.
- Feldmeier, D. (1995). Fast software implementation of error detection codes. *Networking, IEEE/ACM Transactions on*, 3(6):640–651.
- Floyd, S. and Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413.
- Floyd, S. and Paxson, V. (2001). Difficulties in simulating the internet. *IEEE/ACM Trans. Netw.*, 9(4):392–403.
- Ford, D. R. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press, Princeton, NJ, USA.

- Forouzan, B. A. (2007). *Data Communications and Networking – Fourth Edition*. McGraw-Hill.
- Francois, P., Filsfils, C., Evans, J., and Bonaventure, O. (2005). Achieving sub-second igp convergence in large ip networks. *SIGCOMM Comput. Commun. Rev.*, 35(3):35–44.
- Gill, P., Schapira, M., and Goldberg, S. (2013). A survey of interdomain routing policies. *SIGCOMM Comput. Commun. Rev.*, 44(1):28–34.
- Gourley, D., Totty, B., Sayer, M., Aggarwal, A., and Reddy, S. (2002). *HTTP: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition.
- Graziani, R. (2013). *IPv6 Fundamentals: A Straightforward Approach to Understanding IPv6*. Cisco Press, Indianapolis, IN, USA.
- Grigorik, I. (2013). Making the web faster with http 2.0. *Queue*, 11(10):40:40–40:53.
- Gross, J. and Yellen, J. (2005). *Graph Theory and Its Applications, Second Edition*. Textbooks in Mathematics. Taylor & Francis.
- Grosvenor, M. P., Schwarzkopf, M., Gog, I., Watson, R. N. M., Moore, A. W., Hand, S., and Crowcroft, J. (2015). Queues don't matter when you can jump them! In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 1–14, Oakland, CA. USENIX Association.
- Ha, S., Rhee, I., and Xu, L. (2008). Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74.
- Hagen, S. (2014). *Praise for IPv6 Essentials, Third Edition - Integrating IPv6 into Your IPv4 Network*. "O'Reilly Media, Inc.".
- Hall, W. and Tiropanis, T. (2012). Web evolution and web science. *Comput. Netw.*, 56(18):3859–3865.
- Halperin, D., Hu, W., Sheth, A., and Wetherall, D. (2010). 802.11 with multiple antennas for dummies. *ACM SIGCOMM Computer Communication Review*, 40(1):19–25.
- Hamming, R. (1980). *Coding and Information Theory*. Prentice-Hall.
- Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., and McKeown, N. (2012). Reproducible network experiments using container-based emulation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 253–264, New York, NY, USA. ACM.
- Harold, E. (2013). *Java Network Programming, 4th Edition*. O'Reilly Media.
- Held, G. (2003). *Ethernet Networks: Design, Implementation, Operation, and Management*. John Wiley & Sons.
- Hofmann, M. and Beaumont, L. R. (2005). *Content Networking: Architecture, Protocols, and Practice (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Howes, T. A., Smith, M. C., and Good, G. S. (2003). *Understanding and Deploying LDAP Directory Services, Second Edition*. Addison-Wesley Professional.
- Hsu, W. (2003). *Analog and Digital Communications*. McGraw-Hill.

- Huitema, C. (1995). *Routing in the Internet*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
- Huston, G., Rossi, M., and Armitage, G. (2011). Securing bgp - a literature survey. *IEEE Communications Surveys Tutorials*, 13(2):199–222.
- Jacobson, V. (1988). Congestion avoidance and control. In *Symposium Proceedings on Communications Architectures and Protocols*, SIGCOMM '88, pages 314–329, New York, NY, USA. ACM.
- Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley professional computing. Wiley.
- Jennings, C., Hardie, T., and Westerlund, M. (2013). Real-time communications for the web. *Communications Magazine, IEEE*, 51(4):20–26.
- Kamp, P.-H. (2015). Http/2.0: The ietf is phoning it in. *Commun. ACM*, 58(3):40–42.
- Kent, S., Lynn, C., and Seo, K. (2000). Secure border gateway protocol (s-bgp). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592.
- Kleinrock, L. (1972). *Communication Nets; Stochastic Message Flow and Delay*. Dover Publications, Incorporated.
- Kleinrock, L. (1979). Power and deterministic rules of thumb for probabilistic problems in computer communications. In *Proceedings of the International Conference on Communications*, pages 1–10.
- Knight, S., Nguyen, H., Falkner, N., Bowden, R., and Roughan, M. (2011). The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29(9):1765 –1775.
- Kohler, E., Handley, M., and Floyd, S. (2006). Designing dccp: Congestion control without reliability. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 27–38, New York, NY, USA. ACM.
- Kurose, J. and Ross, K. (2013). *Computer Networks – A Top-Down Approach – Sixth Edition*. Always learning. Pearson Education.
- Lampson, B. W. (1983). Hints for computer system design. *SIGOPS Oper. Syst. Rev.*, 17(5):33–48.
- Lampson, B. W. and Sproull, R. F. (1979). An open operating system for a single-user machine. In *Proceedings of the Seventh ACM Symposium on Operating Systems Principles*, SOSP '79, pages 98–105, New York, NY, USA. ACM.
- Liu, C. (2002). *DNS and BIND Cookbook*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition.
- Luo, Z. and Suh, C. (2011). An improved shortest path bridging protocol for ethernet backbone network. In *The International Conference on Information Networking 2011 (ICOIN2011)*, pages 148–153.
- MacKay, D. (2005). Fountain codes. *Communications, IEEE Proceedings-*, 152(6):1062–1068.
- Marsic, I. (2013). *Computer Networks - Performance and Quality of Service*. Rutgers University, New Jersey, USA.

- Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M. Y., and Wang, R. (2001). Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 287–297. ACM.
- Metcalfe, R. M. and Boggs, D. R. (1976). Ethernet: distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404.
- Motamedi, R., Rejaie, R., and Willinger, W. (2015). A survey of techniques for internet topology discovery. *IEEE Communications Surveys Tutorials*, 17(2):1044–1065.
- Moussavi, M. (2012). *Data Communication and Networking: A Practical Approach*. Delmar.
- Müller, C. and Timmerer, C. (2011). A vlc media player plugin enabling dynamic adaptive streaming over http. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM ’11, pages 723–726, New York, NY, USA. ACM.
- Murray, David; Terry Koziniec, K. L. and Dixon, M. (2012). Large mtus and internet performance. In *Proceedings of the 13th IEEE Conference on High Performance Switching and Routing*, HPSR 2012. IEEE.
- Nygren, E., Sitaraman, R. K., and Sun, J. (2010). The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19.
- Paasch, C. and Bonaventure, O. (2014). Multipath tcp. *Commun. ACM*, 57(4):51–57.
- Paasch, C., Detal, G., Duchene, F., Raiciu, C., and Bonaventure, O. (2012). Exploring mobile/wifi handover with multipath tcp. In *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, CellNet ’12, pages 31–36, New York, NY, USA. ACM.
- Pathan, A.-M. K. and Buyya, R. (2007). A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, page 4.
- Peng, T., Leckie, C., and Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv.*, 39(1).
- Perkins, C., Hodson, O., and Hardman, V. (1998). A survey of packet loss recovery techniques for streaming audio. *Network, IEEE*, 12(5):40–48.
- Perlman, R. (1985). An algorithm for distributed computation of a spanning tree in an extended lan. *ACM SIGCOMM Computer Communication Review*, 15(4):44–53.
- Perlman, R. and Eastlake, D. (2011). Introduction to trill. *The Internet Protocol Journal*, 14(3).
- Peterson, L. and Davies, B. (2012). *Computer Networks – a Systems Approach – Fifth Edition*. Elsevier.
- Pierce, J. (1984). Telephony—a personal view. *Comm. Mag.*, 22(5):116–120.
- Pouzin, L. (1975). The cyclades network – present state and development trends. In *Proceedings of the Symposium on Computer Networks*, pages 8–13. IEEE Computer Society.

- Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and Handley, M. (2011). Improving datacenter performance and robustness with multipath tcp. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 266–277, New York, NY, USA. ACM.
- Raiciu, C., Iyengar, J., and Bonaventure, O. (2013). Recent advances in reliable transport protocols. In *H. Haddadi, O. Bonaventure (Eds.), Recent Advances in Networking*, pages 59–106.
- Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and Handley, M. (2012). How hard can it be? designing and implementing a deployable multipath tcp. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 29–29, Berkeley, CA, USA. USENIX Association.
- Rizzo, L. (1997). Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM Computer Communication Review*, 27(2):24–36.
- Rodrigues, R. and Druschel, P. (2010). Peer-to-peer systems. *Commun. ACM*, 53(10):72–82.
- Rorabaugh, C. (1996). *Error Coding Cookbook*. McGraw-Hill.
- Ruiz-Sanchez, M. A., Biersack, E. W., and Dabbous, W. (2001). Survey and taxonomy of ip address lookup algorithms. *IEEE Network*, 15(2):8–23.
- Saltzer, J. H., Reed, D. P., and Clark, D. D. (1984). End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2:277–288.
- Sermersheim, J. (2006). Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511 (Proposed Standard).
- Spurgeon, C. (2000). *Ethernet: the definitive guide*. "O'Reilly Media, Inc.".
- Stallings, W. (2011). *Wireless Communications & Networks*. Pearson Education.
- Stallings, W. (2013). *Data and Computer Communications – 10th Edition*. Pearson Education.
- Stevens, R., Fenner, B., and Rudoff, A. M. (2004). *Unix Network Programming – Volume 1*. Addison-Wesley.
- Stevens, W. and Wright, G. (1994). *TCP/IP Illustrated: The protocols*. Number v. 1 in Addison-Wesley professional computing series. Addison-Wesley Publishing Company.
- Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 149–160, New York, NY, USA. ACM.
- TAN, K., SONG, J., ZHANG, Q., and SRIDHARN, M. (2006). A compound tcp approach for high-speed and long distance networks. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, page 1–12.
- Tanenbaum, A. and Wetherall, D. (2011). *Computer Networks – Fifth Edition*. Pearson Education.

- Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371.
- Vogels, W. (2009). Eventually consistent. *Commun. ACM*, 52(1):40–44.
- Wang, J. (1999). A survey of web caching schemes for the internet. *SIGCOMM Comput. Commun. Rev.*, 29(5):36–46.
- Wang, Z. and Crowcroft, J. (1992). Eliminating periodic packet losses in 4.3-tahoe bsd tcp congestion control algorithm. *ACM Computer Communication Review*, 22(2):9–16.
- Webber, J., Parastatidis, S., and Robinson, I. (2010). *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly Media, Inc., 1st edition.
- Wischik, D., Raiciu, C., Greenhalgh, A., and Handley, M. (2011). Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, volume 11, pages 8–8.
- Xie, H., Yang, Y. R., Krishnamurthy, A., Liu, Y. G., and Silberschatz, A. (2008). P4p: provider portal for applications. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 351–362, New York, NY, USA. ACM.
- Zaragoza, R. (2002). *The Art of Error Correcting Coding*. Addison-Wesley.
- Zec, M., Rizzo, L., and Mikuc, M. (2012). Dxr: Towards a billion routing lookups per second in software. *SIGCOMM Comput. Commun. Rev.*, 42(5):29–36.
- Zhang, C., Dhungel, P., Wu, D., and Ross, K. W. (2011). Unraveling the bittorrent ecosystem. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1164–1177.
- Zimmermann, H. (1980). Osi reference model—the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 28(4):425–432.