universidade de aveiro
theoria poiesis praxis

It's time to recognise the Internet as a basic human right, that means guaranteeing affordable access for all, ensuring Internet packets are delivered without commercial or political discrimination, and protecting the privacy and freedom of Web users regardless of where they live.

Tim Berners-Lee

3 ARQUITETURA DE REDES

Atenção!

Todo o conteúdo deste documento pode conter alguns erros de sintaxe, científicos, entre outros... **Não estudes apenas a partir desta fonte**. Este documento apenas serve de apoio à leitura de outros livros, tendo nele contido todo o programa da disciplina de Arquitetura de Redes, tal como foi lecionada, no ano letivo de 2016/2017, na Universidade de Aveiro. Este documento foi realizado por Rui Lopes.

mais informações em ruieduardofalopes.wix.com/apontamentos

Como continuação da disciplina de Fundamentos de Redes (a3s1), a disciplina de Arquitetura de Redes (a3s2) mostra agora uma aplicação dos conhecimentos adquiridos agora na estruturação e desenho de uma rede empresarial. Quando falamos de **redes empresariais**, referimo-nos a redes montadas entre vários pólos de uma empresa como vários departamentos de uma universidade ou vários pólos clínicos de um hospital.

redes empresariais

Saber desenhar eficiente e eficazmente uma rede computacional para uma empresa poderá tornar-se numa tarefa tão ou mais importante quanto a própria construção dos edifícios onde ela se pretende implementar: num hospital, inclusive, poderá ser decisivo na sobrevivência de algumas pessoas, dado que as máquinas de suporte à vida se encontram ligadas em rede.

1. Tópicos de Desenho de Redes Empresariais

No desenho de uma rede empresarial temos então que ser capazes de conseguir uma montagem eficiente e eficaz, de forma a que a rede continue viável, mesmo depois de acontecimentos inesperados que possam mudar a integridade do edifício (ou edifícios). Em resposta a isto, as nossas redes terão de possuir algumas qualidades como a modularidade, resiliência e flexibilidade: a **modularidade** está presente quando temos inúmeros componentes na rede, cada um com um conjunto finito de responsabilidades e com papeis importantes na rede global, permitindo assim a mudança de escala de atuação e crescimento/evolução dos equipamentos - esta capacidade permite que, ao invés de re-desenhos da rede, sejam adicionados ou removidos componentes desta; em termos de **resiliência**, devemos ser capazes (como engenheiros de redes) de garantir que a rede se encontra ativa aproximadamente 100% do tempo; a **flexibilidade** encontra-se quando a rede é capaz de se adaptar às várias situações/casos de uso de forma rápida.

modularidade

resiliência

flexibilidade

Equipamentos de rede e como os escolher

Como já foi referido, existem vários **equipamentos** que nos trarão várias funcionalidades ao longo da montagem ou do desenho de redes empresariais. Ao longo deste documento iremos usar os ícones da Figura 1.1 para os retratar.

equipamentos

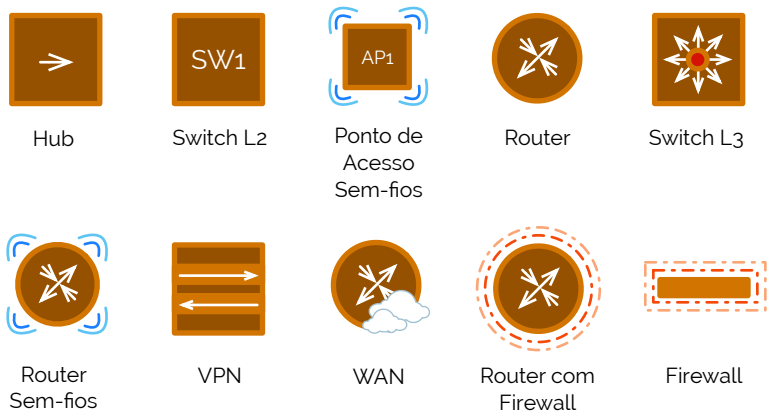


figura 1.1
legenda dos vários equipamentos

Mais equipamentos para além dos especificados na Figura 1.1 poderão aparecer, sendo que serão devidamente legendados caso tal aconteça.

Entre todos estes equipamentos é importante saber escolhê-los, não só pelas suas funções, como pelas suas funcionalidades, através das especificações dos fabricantes. Note-se que o primeiro ponto a verificar não é, de todo, a marca! Pelo contrário, primeiro devemos considerar pela fiabilidade mais elevada (identificação de tempo médio entre falhas - MTBF), tendo sempre em conta que quanto maior for a fiabilidade, maior é o preço. E é precisamente com este indicador que devemos jogar de seguida - o preço -, seguido da assistência técnica do equipamento pelo fabricante em caso de falha.

Do lado mais técnico, devemos averiguar primeiramente as velocidades de processamento e de comutação entre as funções de *routing* e de *switching*, geralmente expressas em termos de número de bytes (ou de pacotes) processados (ou comutados) por segundo. Só de seguida é que é aconselhável verificar se os equipamentos se regem pelos nossos requisitos como número e tipos de portas, tal como saber se possui portas de expansão (novos *slots* para expansão).

Modelo hierárquico de redes

Com o estudo de todos estes problemas para o desenho de redes empresariais, a Cisco avançou com uma boa organização de distribuição de responsabilidades numa rede. Criou-se assim o **modelo hierárquico** o qual possui, em grande escala, três grandes níveis bem designados e discriminados. De baixo para cima, onde o baixo é a raiz de toda a hierarquia, começamos pelo **núcleo** da rede, onde se encontra o ponto fulcral de toda a rede e conteúdos, sendo considerado como o **backbone**, isto é, o ponto crítico para a conetividade, que deverá fornecer um nível de disponibilidade mais alto possível e facilmente adaptável às mudanças que possam ocorrer.

modelo hierárquico

núcleo

backbone

Num nível superior é então possível ser feita a agregação de vários dispositivos de uma rede de área local (LAN), segmentar grupos de trabalho, isolar problemas de rede, agregar conexões WAN (acrónimo de *Wide Area Network*, significando um ponto de extensão da rede num local remoto), aplicar ligações de rede com base em políticas e aplicar políticas de qualidade de serviço (QoS). A esta camada damos o nome de camada de **distribuição**.

distribuição

Finalmente, após uma distribuição de responsabilidades, cabe à última camada garantir o acesso dos equipamentos à rede, numa camada denominada de camada de **acesso**.

acesso

Na Figura 1.2 podemos ver esta arquitetura de estrutura hierárquica.

figura 1.2
modelo hierárquico

Na Figura 1.3 podemos ver um exemplo mais prático, de aplicação desta mesma arquitetura-base numa rede empresarial (mais em particular num pólo de uma empresa). Note-se que nesta figura mostramos um equipamento novo denominado de **PSTN** (sigla para *Public Switched Telephone Network*), sendo a rede telefónica da empresa.

PSTN

Arquitetura de uma rede empresarial

Na arquitetura de uma rede a arquitetura dos edifícios onde esta se vai implementar é muito importante do processo de desenho. Por outras palavras, todos os elementos que, de alguma forma, compõem uma empresa - alojada em um ou mais edifícios - deverão estar de igual forma patentes no desenho da rede. Temos assim cinco grandes pontos de contacto que poderão ser vistos na Figura 1.4, entre eles o campus, o datacenter, as filiais (ou pólos), as redes WAN e os utilizadores remotos, com acesso à rede da empresa em qualquer local do globo.

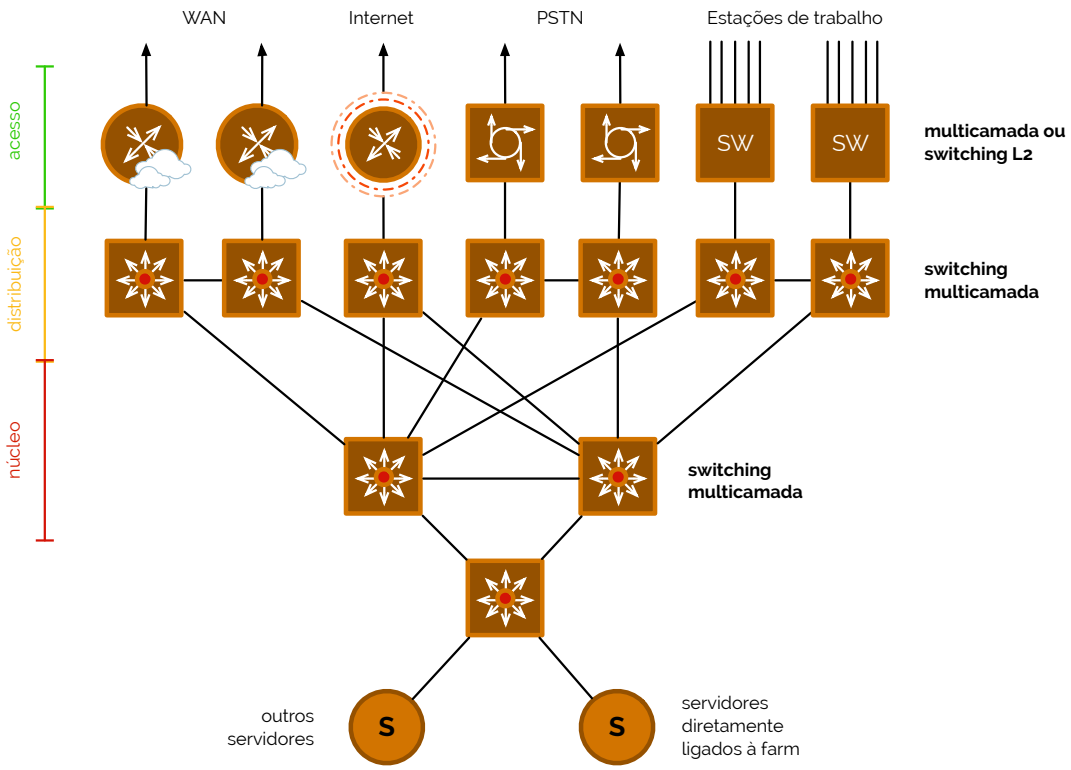


figura 1.3

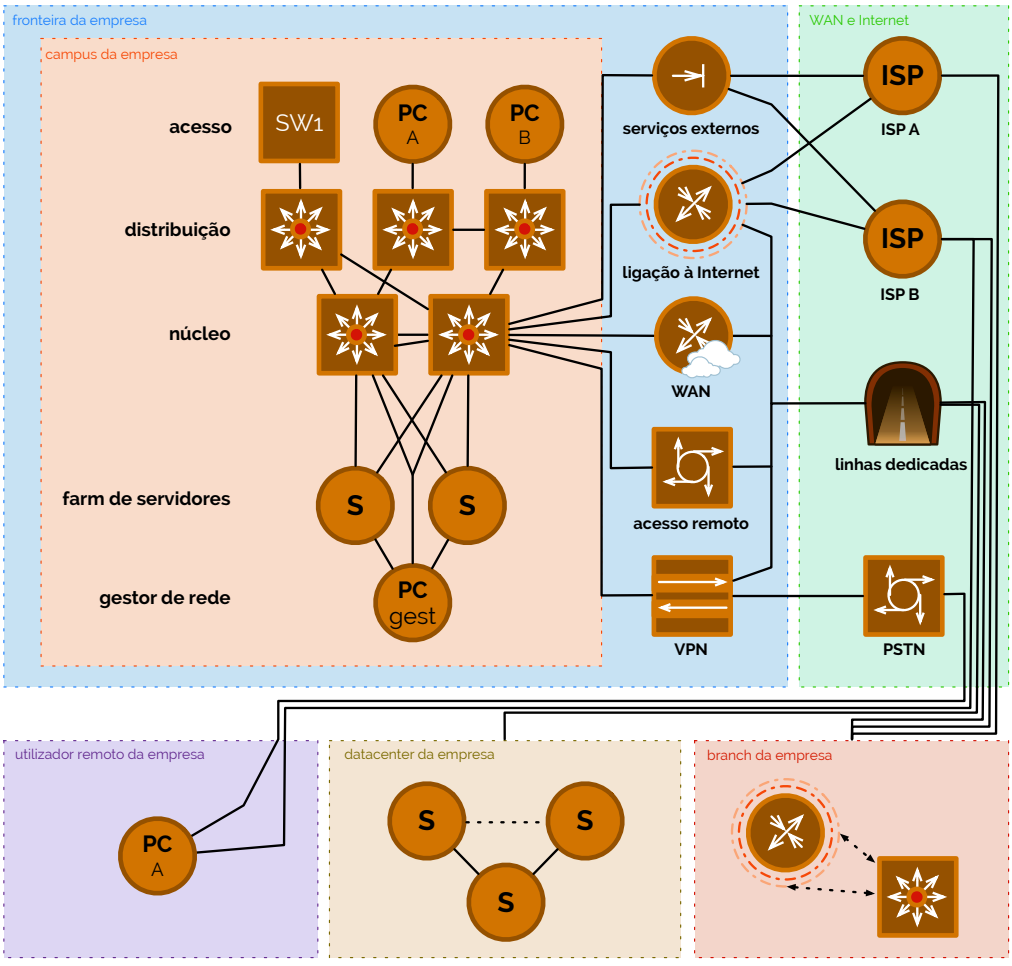


figura 1.4

campus

O **campus** de uma empresa é onde está o cerne de toda a gestão de rede. Seja um campus ou campi de uma mesma empresa, devemos sempre assumir que a maior parte dos utilizadores irão aceder à rede através deste ponto de contacto. Como lhe cabe a gestão da rede, é natural que nesta estejam alojados, em termos de infraestruturas, todos os recursos necessários para complementar o núcleo da mesma, com mecanismos de segurança avançados, capacidades de *switching* e de roteamento com mobilidade.

datacenter

O **datacenter** é um ponto de contacto com a rede empresarial onde se estabelecem todos os critérios de persistência dentro desta. Por outras palavras, estando alojada num bloco à parte - de preferência com altas restrições de segurança -, todos os dados que necessitam de ser preservados por longos intervalos de tempo deverão ser guardados no datacenter. Para que isso aconteça e para evitar fenómenos de *bottleneck*, um datacenter precisará sempre de um conjunto de componentes com um nível de redundância considerável. O facto dos dados serem alojados fora do contexto normal e central de uma rede, permite que a rede evolua de forma totalmente independente da forma como o datacenter é gerido, permitindo um nível maior de flexibilidade.

branch

Um **branch** (em português pólo ou filial) de uma empresa é uma pequena estação de serviços que não é contígua aos edifícios centrais da mesma, localizados no campus (ou campi). Este bloco premite que as empresas extendam os seus serviços a locais mais remotos e mais próximos dos clientes, não obstante, com um nível de segurança específico para o acesso à rede empresarial.

utilizadores remotos

As redes empresariais muitas vezes também possuem WAN, que como já foi referido, permite que se estabeleça uma comunicação interna entre dois ou mais redes da mesma empresa. Mais, muitas vezes acontece que existem **utilizadores remotos**, individuais, que necessitam de comutar nos seus trabalhos e precisam de acesso remoto à rede da empresa. Este é um quinto ponto de contacto, sendo que deverão possuir uma ligação segura para a rede através de um serviço VPN (iremos abordar mais à frente).

Ao desenhar uma rede é importante manter em mente dois aspetos [2]: manter um nível de redundância apropriado (geralmente usa-se redundância 2 ou 3, em raros casos) para contornar possíveis problemas dentro da rede, e evitar encadeamento de equipamentos dentro de uma determinada camada. Referir aumentar a redundância para contornar problemas na rede pode significar que poderão ocorrer problemas de ligação, como até mesmo considerar cenários de incêndios ou inundações num edifício. Se um destes cenários ocorrer, tendo redundância nas ligações permite-se que se algo aconteça a uma das ligações, a outra permaneça ativa e assim nada deixe de funcionar. [1]

Planeamento de uma rede

planeamento

Tendo já alguns conceitos em termos de modularidade e hierarquia de uma rede empresarial ficamos assim com uma base sólida para avançarmos para o verdadeiro **planeamento**. Para simplificarmos a forma como abordar este tópico, iremos começar pela camada mais próxima do cliente final até à camada mais próxima do administrador de redes, isto é, da camada de acesso para a camada de núcleo.

camada de acesso

Começando então pela **camada de acesso** temos que garantir, com esta, a maior disponibilidade de todas, dado que é aqui que os nossos clientes se vão conectar. Para isso devemos fazer várias ligações redundantes entre os nossos equipamentos desta camada com os devidos *gateways* na camada de distribuição (quem liga ao acesso). Claro está que esta redundância não poderá passar apenas pela questão da ligação de rede, mas também deverá passar em termos de energia, sendo que deverão ter fontes redundantes de energia.

Como podemos ver na Figura 1.5 temos a redundância de ligação de rede entre as duas camadas de acesso e distribuição.

Note-se que há mais aspetos com que nos devemos preocupar no caso das camadas de acesso. Consideremos o cenário de uma empresa real: será que um visitante da mesma poderá entrar num dos seus edifícios e conectar o seu computador à rede de forma direta, isto é, sem qualquer tipo de autenticação ou verificação da sua identidade? Ora, na verdade, numa rede que se preze, isso nunca acontece - e se acontecer será uma secção

própria da rede que permite que isso aconteça sem que sejam postas em causa as demais secções (havemos de voltar a este tópico mais à frente, em Partições de acesso à rede por VLANs). Este critério de autenticação serve assim para garantir alguma segurança na rede, e deverá ser feita através de algumas normas que já existem como a IEEE 802.1X, entre outros, que abordaremos mais à frente, quando discutirmos a segurança em redes.

‹ IEEE 802.1X

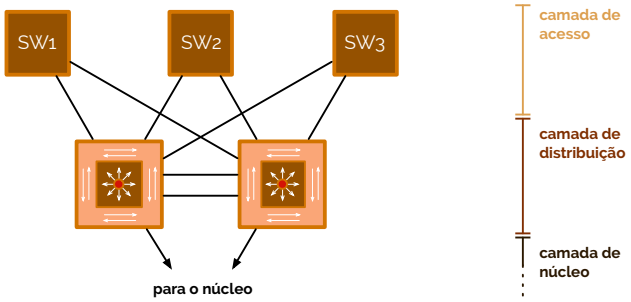


figura 1.5

Mais que a própria segurança, também é necessário saber estipular critérios que permitam uma boa qualidade de serviço (assunto a abordar mais à frente), isto sendo que deverá haver uma diferenciação de tráfego por classificação de importância deste, formando filas de espera com pacotes não tão importantes, até que haja disponibilidade por parte de *routers* e clientes para transmitir/receber a informação, ou capacidade de ligação para efetuar a troca de mensagens.

Na Figura 1.5 temos uma representação de um equipamento que não vimos até ao momento. O que se pretende representar por esta figura é um *switch layer* 3 (L3) com capacidade de roteamento para distribuição, que usa uma combinação de *switching* de camada 2 com multi-camada, de forma a poder segmentar grupos diferentes de trabalho e isolar possíveis problemas de rede, da mesma forma que prevenir que estes causem impactos na camada de núcleo.

Sendo um equipamento da **camada de distribuição**, este é responsável por ligar os serviços de rede à camada de acesso e assim implementar critérios e regras de qualidade de serviço, segurança, balanceamento de tráfego, entre outras... Claro está que para isto os requisitos de disponilidade e resiliência deverão ser ainda mais apertados.

camada de distribuição

Na Figura 1.6 podemos então ver o acrescento da camada de distribuição, agora mais completa, onde vemos dois conjuntos de módulos de distribuição ligados ao núcleo. Esta ligação ao núcleo deverá ser feita através de protocolos dinâmicos de roteamento, como é o caso de um protocolo RIP (como abordámos em Fundamentos de Redes (a3s1)), embora não seja a melhor abordagem, conforme iremos verificar mais à frente. Do lado contrário, é normal que seja aqui que as VLANs da camada de acesso estejam ultimamente definidas, tal como as suas devidas regras para a qualidade de serviço, roteamento controlado ou filtragem de tráfego.

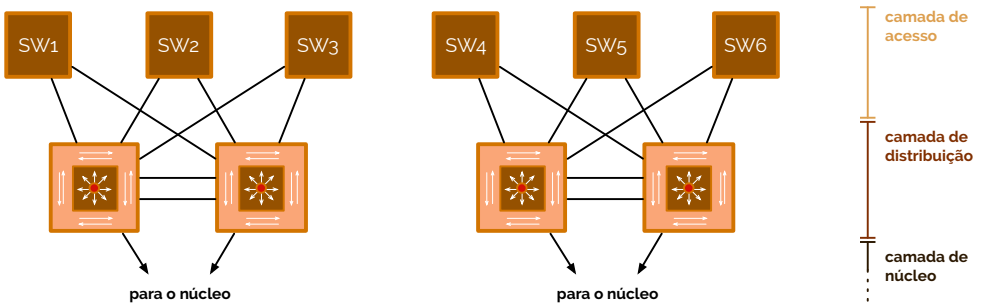


figura 1.6

Então e porque não ligar diretamente o SW3 da Figura 1.6 diretamente com o SW4, para fornecer uma rota mais curta entre terminais no acesso? Ou então, porque não ligar dois conjuntos de distribuição diretamente, ao invés de ter rotas que vão ao núcleo primeiro para voltar para uma distribuição? Estas opções são denominadas de **daisy chain-**

ing, tal como abordámos em Arquitetura de Computadores II (a2s2), e estão representadas na Figura 1.7. [1]

daisy chaining

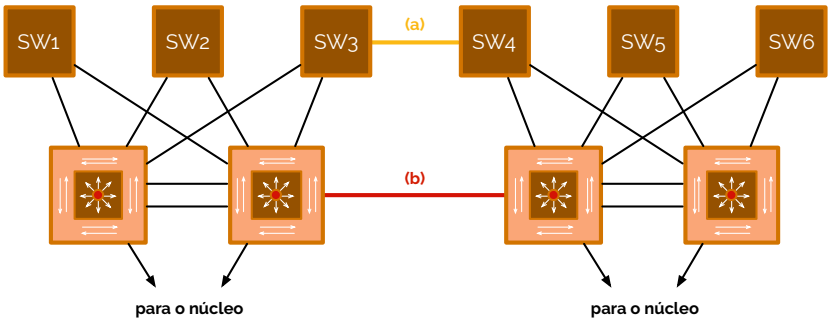


figura 1.7

Ora, o *daisy chaining*, embora não seja válido em muitas situações, não é impossível. Por exemplo, no caso assinalado na Figura 1.7 com (a) fazer uma ligação em camada 2 torna a hierarquia inválida.

Pelo contrário, no caso assinalado com (b), na Figura 1.7, numa camada de acesso, qualquer caminho proveniente de um *switch* não requerirá outro *switch* da mesma camada. Já numa camada de distribuição, qualquer caminho entre os *switches* desta não requerirão um *switch* na camada de acesso. Mais, note-se que com a mesma ligação, mas de camada 2, embora continue a não ser impossível, os *switches* de acesso poderão ser sobrecarregados e o protocolo de *spanning-tree* poderá tornar-se mais moroso até à sua convergência, em caso de falhas.

↬ IEEE 802.1D

Não obstante de algumas vantagens e desvantagens, fique-se sempre com a ideia que deverá ser regra comum evitar o *daisy chaining*.

Tendo já referido tanto a camada de acesso, como a camada de distribuição, precisamos agora de entrar na camada final - a de núcleo. A **camada de núcleo** (vulgarmente designada de *core*) é o ponto central de ligação num *campus* e ponto de agregação para as outras camadas. Como podemos ver nas figuras anteriores (vide Figura 1.7, por exemplo), os módulos da camada de distribuição apontam todos para o núcleo. Mas será que todos os casos de redes necessitam de um núcleo e, quando precisam, por que módulos é este constituído?

camada de núcleo

Vamos começar pela última parte da questão: "por que módulos é o núcleo constituído?". Ora, a camada de núcleo pode ter um de dois tipos: poderá ser constuído por *switching* de camada 2 (L2) ou por *switching* de camada 3 (L3). Num *switching* de camada 2 consideremos primeiramente um caso em que só temos uma VLAN, isto é, somente uma sub-rede IP é usada no núcleo - vide Figura 1.8.

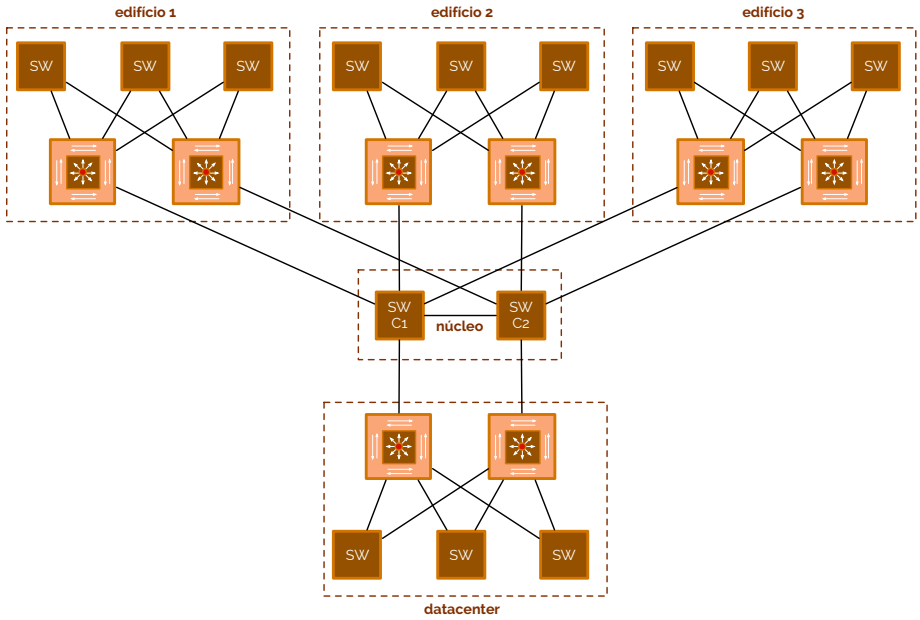


figura 1.8

Analisando este exemplo com mais cuidado podemos ver que até temos algumas vantagens, como o protocolo de *spanning-tree* não cortar nenhuma ligação, uma vez que, por si, a rede com topologia de estrela não possui qualquer ciclo - já é uma árvore abrangente. No entanto, temos uma grande desvantagem - contrariamente ao intuito de uma hierarquia de rede, só temos um domínio de *broadcast*, isto é, cada vez que for feito um envio de pacotes em *broadcast* ou *multicast* de um terminal no acesso, os pacotes irão até ao núcleo - pelo processo de *flooding*. Concluímos assim que esta prática não é a melhor como núcleo - mas será que separar a camada 2 em VLANs em dois núcleos completamente redundantes nos resolve o problema, como se representa na Figura 1.9?

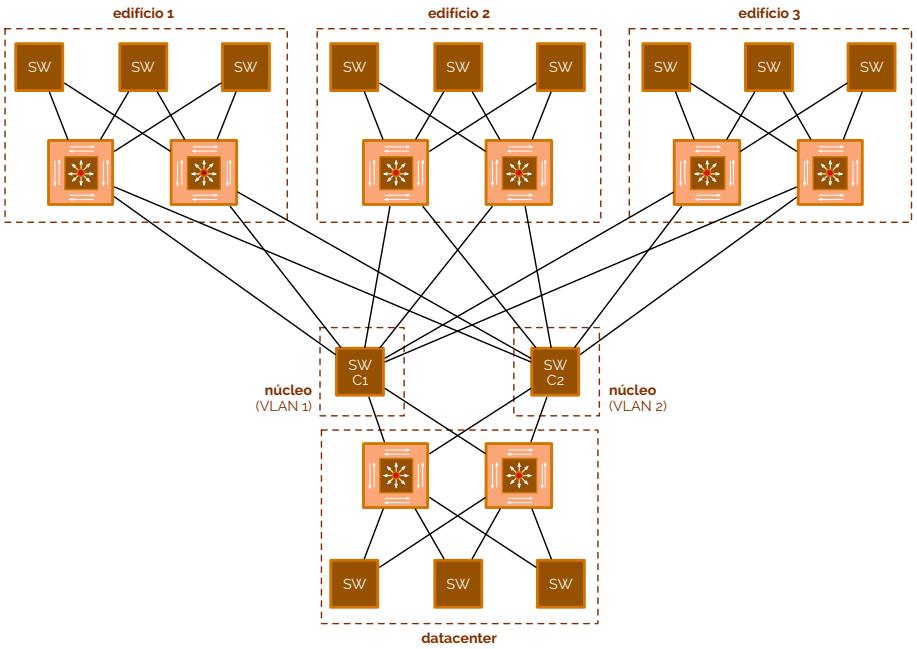


figura 1.9

Consideremos, portanto, que no caso da Figura 1.9 não existe nenhuma ligação *trunk* entre as VLANs. Com este esquema cada módulo da distribuição possui agora dois caminhos distintos para cada outro módulo de distribuição, com o mesmo custo - se o caminho da VLAN 1 for cortado, então o módulo de camada 3 encarregar-se-á de encaminhar de imediato todo o tráfego para a VLAN 2. Isto permite que haja maior convergência em termos de casos de falha. No entanto, continua a não ser uma muito boa opção, dado que precisamos de ligações extra de cada *switch* da distribuição para cada *switch* do núcleo.

Voltamo-nos assim para o núcleo formado com *switching* de camada 3 (L3), onde podemos ter algo como o representado na Figura 1.10. Este caso é bastante análogo ao segundo caso analisado em camada 2, mas possui agora o processamento de uma camada 3. Como temos caminhos duplos para os diferentes módulos de distribuição, tal como no exemplo de camada 2, agora podemos afirmar ainda mais, dizendo que podemos criar VLANs de interligação entre os demais módulos do núcleo e distribuição, para efetuar as várias interconexões entre VLANs no acesso.

Resumindo, temos que os núcleos compostos por *switching* L3 têm uma topologia mais flexível - sem ciclos de *spanning-tree* -, possuem um maior controlo de *broadcasts* e *multicasts* em pleno núcleo e permitem uma maior escabilidade da rede, numa escala arbitrariamente maior.

De facto, estas escolhas são as melhores porque é papel do núcleo conter o processamento mais eficaz e eficiente de toda a rede (se possível com aceleração do *hardware*), ter redundância suficiente para garantir fiabilidade do sistema, possuir capacidade de implementação de protocolos de encaminhamento escaláveis e de novas tecnologias, tal como de balanceamento de tráfego.

Em resposta a uma questão anterior ("mas será que todos os casos de redes necessitam de um núcleo?"), por outro lado, a resposta é não.

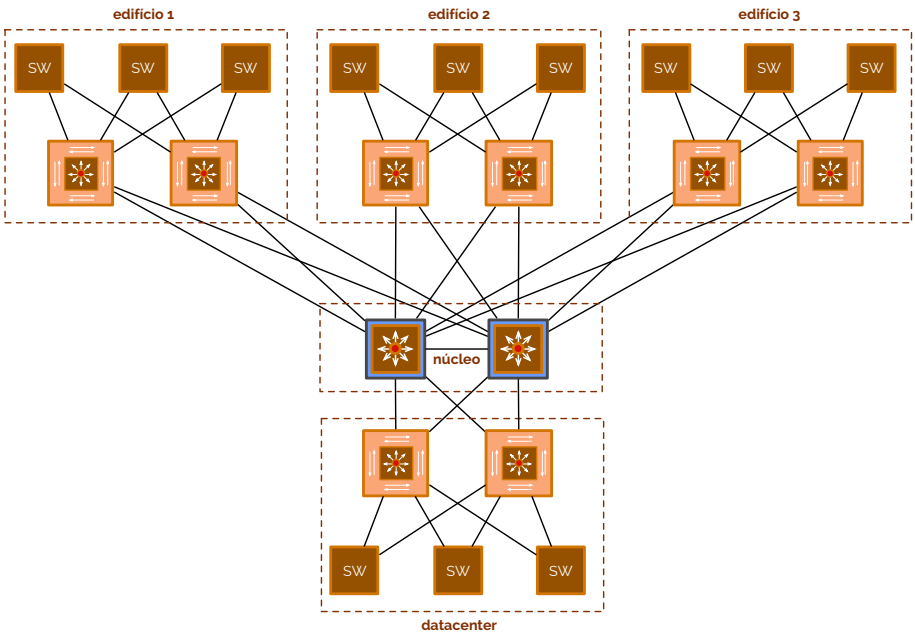


figura 1.10

Nem todos os *campi* necessitam de um núcleo na sua rede, deixando essa responsabilidade para a camada de distribuição. Neste caso dizemos que o núcleo está **colapsado** - em inglês diz-se *collapsed core*. Na Figura 1.11 temos um exemplo de núcleo colapsado.

colapsado

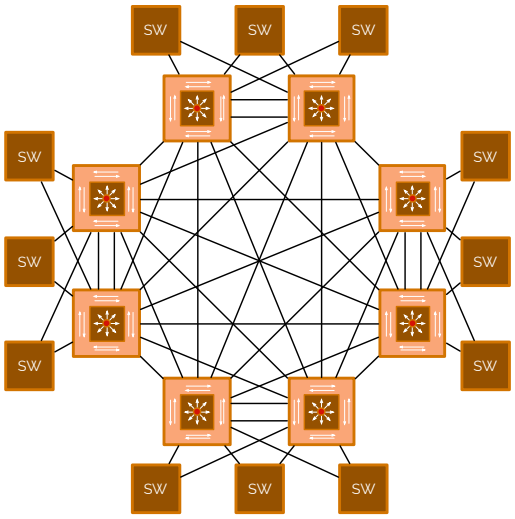


figura 1.11

Analisando a rede da Figura 1.11 podemos verificar logo à partida que a escalabilidade desta solução não é, de todo, a melhor, dado que só a cablagem aumenta muito mais, quanto mais o número de equipamentos também segue a mesma tendência. Por esta mesma razão as arquiteturas de rede com base em núcleos colapsados são úteis para *campi* muito pequenos, com redes igualmente pequenas, o que elimina a necessidade de *hardware* extra para *switching* e simplifica a implementação de toda a rede. Não obstante, com esta arquitetura excluímos as vantagens de uma organização multi-camada, como a isolação de pontos de falha. [2]

Com um desenho hierárquico, nas camadas de núcleo e distribuição os **pontos de falha** podem ser resolvidos pela prórpia redundância inerente à organização (redundância nas ligações, energia, ventilação, ...). Já na camada de acesso, embora seja mais difícil, podemos ter *hardware* redundante, mas como temos maior segmentação em VLANs, podemos isolar o problema, mesmo que dentro de uma VLAN num edifício específico. Note-se, no entanto, que a redundância, *per si*, também não poderá ser muito avultada, dado que um largo número torna as redes complicadas: prefira-se sempre redundância 2.

pontos de falha

Então e os clientes que pretendem associar-se à rede via interfaces sem-fios? As **redes sem-fios** (como iremos revisitar mais à frente) deverão ter o seu ponto de integração no núcleo ou nas camadas de distribuição, sendo que qualquer rede *wireless* de área local (WLAN) pode ser vista como uma LAN convencional[1].

redes sem-fios

Esta integração poderá ser feita através de um equipamento denominado por *Wireless LAN Controller* anexo à camada de distribuição ou núcleo. Estes controladores associam vários **Access Points** (vulgarmente abreviados de AP) com portas *trunk*. [1]

Access Points

Partições de acesso à rede por VLANs

Como já tivemos oportunidade de verificar em Fundamentos de Redes (a3s1) existem pequenas concentrações de equipamentos em redes que formam redes de área locais, isto é, **LAN**s. Uma LAN é assim um grupo de equipamentos que formam um grupo de trabalho, restringindo o domínio de *broadcast* aos membros apenas associados a esta, requerendo *routers* para encaminhar tráfego de dentro para fora (e vice-versa). Com o desenvolver dos tempos e das tecnologias deu-se um passo na virtualização destas práticas e atingiu-se o conceito de **VLAN** - *Virtual Local Area Network*. A grande diferença é que agora não estamos a tratar de um conjunto de equipamentos *per si*, mas antes de um grupo de portas individuais de um *switch* que formam uma área de trabalho lógica - não física como antes. [3]

LAN

VLAN

De uma forma muito simplificada, uma VLAN pega numa LAN e separa o seu domínio de *broadcast* em vários mais pequenos. Esta segmentação, dado que é virtual, é totalmente independente da sua localização.

Existem dois tipos de VLAN, sobre os quais referimos como **segmentação** de uma VLAN: VLANs **end-to-end**, que estão associadas a portos de um *switch* e espalhadas por toda uma rede; VLANs **locais**, que geralmente estão confinadas a uma região especial e única. Esta segmentação existe porque há um conjunto de serviços possíveis de se concretizarem num ambiente empresarial que têm comportamentos e utilidades muito comuns. A rede sem-fios é um exemplo que queremos que seja *end-to-end*, isto porque queremos ser capazes de nos associar à mesma rede independentemente do edifício onde nos encontramos (isto é, independentemente dos *switches* da distribuição a que estamos ligados). Já por VLANs locais são exemplo redes de determinados serviços, por ator e/ou local: por exemplo, podemos criar uma VLAN local que discrimina os terminais de engenheiros de um determinado edifício de uma empresa.

segmentação

end-to-end

locais

O propósito principal para estes tipos de segmentação provém do facto de nos ser útil aplicarmos regras lógicas a conjuntos de equipamentos espalhados pela rede, mas que possuem, entre eles, responsabilidades comuns. Assim sendo cada VLAN deve possuir um IP único de sub-rede. Na verdade, cada VLAN poderá possuir mais que um IP (em IPv4 poderá possuir um privado e um público, por exemplo, e em IPv6 poderá possuir múltiplos endereços).

Alguns serviços em particular, por si, já pedem que sejam instalados dentro de um contexto particular de segmentação. Por exemplo, as VLANs de manutenção da rede são usadas para fazer ações de configuração e ajustes numa rede, pelo que precisam de ser *end-to-end* para que possam chegar a todos os equipamentos relevantes. Já o **VoIP** (*Voice Over Internet Protocol*), com um propósito semelhante ao de um telefone convencional, deverá ser instalado sobre uma VLAN local, de forma a que se possa relacionar a sua identificação com uma parte em específico de uma empresa.

VoIP

Como também já abordámos em Fundamentos de Redes (a3s1), os *switches* L2, como podem ter em si definidas múltiplas VLANs e têm de ser capazes de distinguir pacotes de VLANs diferentes, sem os trocar e sem olhar aos endereços IP (dado que são camada 2), têm dois tipos de ligação: ligações de **acesso** e ligações **trunk**. Uma ligação de acesso é tal que indicamos que VLAN é que se deverá considerar numa determinada porta

acesso, trunk

1 Exceto em cenários de mobilidade, onde a conexão deverá manter-se enquanto há movimento.

ou conjunto de portas, e uma ligação *trunk* é tal que permite que se passem um conjunto de VLANs, cujos pacotes devidamente identificados pelo uso de cabeçalhos modificados pela norma **IEEE 802.1Q**, são analisados e confirmados para o uso dos devidos canais de comunicação. Nestas portas *trunk* podemos assim selecionar todas as redes que pretendemos que passem através do seu identificador da VLAN, denominado de **tag**. Veja-se um exemplo de aplicação na Figura 1.12.

IEEE 802.1Q
IEEE 802.1Q
tag

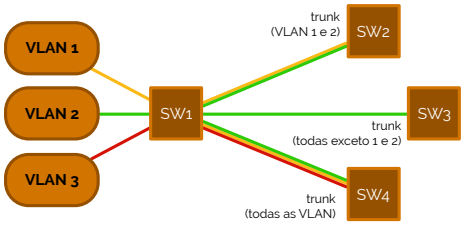


figura 1.12

Para comunicarmos entre duas VLANs diferentes podemos usar várias alternativas, mas precisamos sempre de um equipamento a mais que os representados na Figura 1.12 - precisamos de um *router*. Consideremos um cenário em que queremos comunicar da VLAN 1 para a VLAN 3. Se o *switch* 1 da Figura 1.12 estivesse ligado a um *router* numa ligação *trunk*, então era-nos permitido comunicar da VLAN 1 para a VLAN 3 - o pacote sairia da VLAN 1, passava pelo *switch* 1 e era levado para todas as interfaces onde a VLAN 1 pode passar, depois chegava ao *router*, o qual encaminhava o tráfego para a VLAN 3, modificando no cabeçalho 802.1Q a *tag* de 1 para 3, levando o pacote novamente para a ligação *trunk*, desta vez para fazer circular o pacote por todas as interfaces onde a VLAN 3 pode circular, chegando à VLAN 3. Veja-se a Figura 1.13.

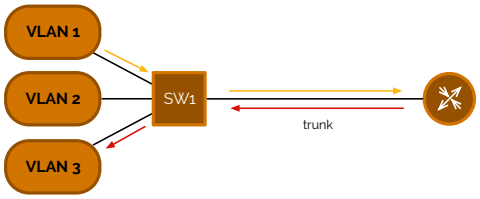


figura 1.13

Se quisermos implementar uma VLAN *end-to-end* no esquema da Figura 1.13 não o podemos fazer, simplesmente porque no *router* não nos é possível ligar diretamente uma rede para *switching* - é um *router*, não faz *switching*. Para isso podemos usar uma extensão do *router* que é um *switch* L3, à qual já podemos anexar um braço diretamente conectado com uma VLAN existente como a VLAN 1 - vide Figura 1.14.

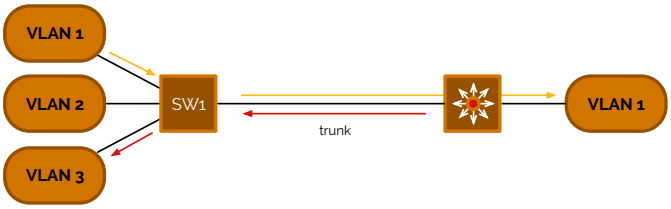


figura 1.14

Se quisermos tornar o caminho ainda mais curto, podemos aplicar precisamente o mesmo *switch* L3 no lugar do *switch* 1, como podemos ver na Figura 1.15.

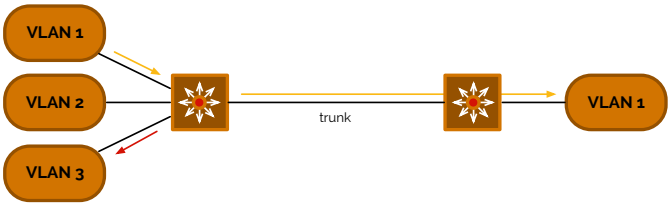


figura 1.15

Como podemos ver na Figura 1.15 temos que, através do encaminhamento, o primeiro *switch* L3 leva o pacote da VLAN 1 para a VLAN 3, não sendo o segundo *switch* L3 a fazê-lo.

Num exemplo agora mais complexo, podemos, com estes mesmos equipamentos, criar duas ligações entre os *switches* L3 - uma de *switching* e uma de *routing*. Consideremos, neste caso, que temos 5 VLANs: 4 locais (VLAN 2, 3, 4, 5) e uma *end-to-end* (VLAN 10). Vejamos a Figura 1.16.

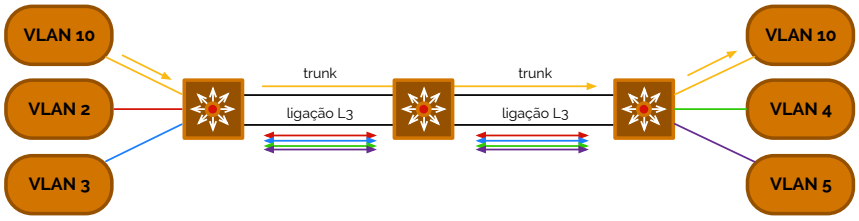


figura 1.15

Neste caso, como temos ambas ligações de *switching* e de *routing* (ligação L3), temos que as VLANs *end-to-end*, desde que permitidas de circulação na ligação *trunk*, transitam na rede por vias de *switching*. Por outro lado, as VLANs locais têm de passar e ser encaminhadas pelos *routers* através das ligações de *routing*.[2]

Mas ainda podemos simplificar isto ainda mais. Através do conceito de **VLAN de interligação** (vulgarmente conhecida como *Core VLAN*) podemos partilhar o mesmo meio físico para comunicar entre VLANs locais, permitindo assim um modo de segmentação híbrido, entre VLANs *end-to-end* e locais. Desta forma, as ligações *trunk* que se sujeitam a VLANs de interligação deverão apenas deixar passar as VLANs *end-to-end* e as VLANs de interligação. Mais, a troca de informação de encaminhamento (dependendo do protocolo em curso) deverá existir apenas na VLAN de interligação, deixando as outras VLANs com as suas interfaces passivas. Caso isto não se verificasse, então haveria um balanceamento de carga entre as várias rotas disponíveis, que seriam a VLAN de interligação, mais todas as VLANs que supostamente a usariam. [1]

VLAN de interligação

Planeamento de alocação de endereços IP

A alocação de endereços é um passo muito importante do planeamento de uma rede. Se este passo estiver bem feito será pouca a necessidade de andar com um dicionário de endereços IP por função dentro de uma rede empresarial, isto porque é possível fazer agregações e sumarizações de endereços IP de forma a que, pelos bits designados, se consiga identificar se o endereço em causa pertence a um terminal, telefone, impressora, televisão, … ou até mesmo onde é que este se encontra ou quem é que o usa. [4]

Como vimos é importante a designação de VLANs para segmentar as nossas redes e isolar partes em caso de manutenção ou falhas. Geralmente quando estamos a tentar alocar endereços para um conjunto de VLANs fazemo-lo em **blocos**, isto é, dada uma gama de endereços IP, tendemos sempre para o cortar em várias partes potências de base 2, aplicando os nossos conhecimentos de criação de sub-redes. Nesta organização, que não está, de todo, incorreta, podemos já começar por designar funções aos blocos: devemos ter blocos para o datacenter, DMZ, para aplicação de NAT, para ligações ponto-a-ponto e para as várias VLANs que vamos ter na nossa rede global. Neste raciocínio, podemos aplicar a **sumarização de redes** sempre que temos um conjunto de sub-redes que usam o domínio completo de uma rede maior. Por exemplo, se tivermos as sub-redes 10.8.0.0/24, 10.8.1.0/24, 10.8.2.0/24 e 10.8.3.0/24 podemos sumariá-las com 10.8.0.0/22. O mesmo se aplica a IPv6, sendo que 2001:A::00:/64, 2001:A::01:/64, 2001:A::02:/64, 2001:A::03:/64, …, 2001:A::F0:/64, …, 2001:A::FF:/64 é equivalente a indicarmos 2001:A::/56.

blocos

sumarização de redes

2 Para garantir este mesmo cenário, nestas condições, é importante que as VLAN end-to-end estejam definidas como interfaces passivas, de modo a que as suas rotas não sejam comunicadas no protocolo de encaminhamento em curso nos routers.

Na alocação de endereços IPv4 privados é importante assim saber **agregar** significados dos mesmos. Por exemplo, tendo uma rede 10.0.0.0/8 podemos criar sub-redes com a forma `10.<aaaabbbb>.<ccccdddd>.0/24` sendo que `aaaa` são 4 bits para a definição do local, `bbbb` são 4 bits para a definição de um prédio, `cccc` são 4 bits para a definição de um grupo de trabalho e `dddd` são 4 bits para a definição de um serviço. O mesmo se pode fazer (e convém) com IPv6.

agregar

Para ligações ponto-a-ponto devemos sempre usar máscaras /30 para IPv4 e /126 para IPv6, e para interfaces *loopback* /32 para IPv4 e /128 para IPv6.

2. Encaminhamento entre Redes num Sistema Autónomo

Na camada 2 do modelo OSI (*link-layer*) os pacotes que são analisados levam o processamento nos respetivos equipamentos a analisar os cabeçalhos Ethernet, que possuem um endereço MAC. Em camada 3 (camada de rede), tendo estudado o protocolo IP, necessitamos tanto de novos equipamentos para efetuar o processamento (agora de gamas de endereços IP), como de novas regras para implementar a forma como selecionar e tratar cada pacote - para isso temos, entre muitas outras coisas, **protocolos de encaminhamento**.

ISO/IEC 7498-1

protocolos de encaminhamento

Protocolos de encaminhamento

Os protocolos de encaminhamento são tais que especificam regras para o trânsito de pacotes entre as unidades mais básicas existentes na camada 3 do modelo OSI - as redes.

A forma mais fácil de encaminhar tráfego é através das **rotas estáticas**. Tal como o próprio nome o indica, estas rotas após a sua ativação, até à sua modificação ou eliminação, manter-se-ão sempre iguais, mesmo que a topologia da rede mude - no pior dos casos as rotas tornam-se inválidas quando um dos caminhos nelas incluídas deixa de existir. Mais, por serem estáticas, a sua escalabilidade não é real, à medida que a rede aumenta de dimensões, pelo que se cria um débito técnico cada vez maior, quanto maior for o aumento.

rotas estáticas

Não obstante, as rotas estáticas devem ser usadas em algumas situações. Por exemplo, quando se pretende ligar a uma rede que apenas possui um caminho podemos usar uma rota estática, tal como quando não podemos usar outro tipo de rotas, porque o *router* não tem capacidade para suportar tanto processamento. Em algumas circunstâncias, o administrador de rede também poderá usar rotas estáticas se pretender ter controlo total sobre as rotas usadas pelo *router*.

Por vezes, e pode vir a ser bastante útil, o *router* não precisa de reconhecer os detalhes das redes remotas, podendo ser configurado para enviar todo o tráfego para um vizinho em específico. Neste caso dizemos que pretendemos usar **rotas estáticas de defeito** (em inglês *static default routes*). Para as configurar devemos criar uma rota cujo endereço é o sumário de todos os endereços IP: no caso de IPv4 referimo-nos a 0.0.0.0/0 e em IPv6 referimo-nos a ::/0. Por exemplo, se estivermos num *router* da marca Cisco (com sistema operativo IOS) podemos dizer que pretendemos uma rota estática de defeito IPv4 e IPv6 através do endereço `address-next-hop`, com os comandos de Código 2.1.

rotas estáticas de defeito

```
R1(conf)# ip route 0.0.0.0 0.0.0.0 address-next-hop
R1(conf)# ipv6 route ::/0 address-next-hop
```

código 2.1

Como também vimos em Fundamentos de Redes (a3s1), a alternativa às rotas estáticas são as **rotas dinâmicas**. Em suma, enquanto que as rotas estáticas são inseridas através de um administrador de redes, as rotas dinâmicas são automaticamente aprendidas através de partilha de informação para/de vizinhos. Assim é-nos possível ter uma rede que, ao alterar a sua topologia, altere automaticamente a forma como processa o seu encaminhamento interno.

rotas dinâmicas

Contrariamente às rotas estáticas, existem vários protocolos de encaminhamento com base em rotas dinâmicas. Em Fundamentos de Redes (a3s1) abordámos e analisámos com bastante detalhe o **RIP**. Não obstante, existem muitos outros protocolos que não iremos abordar neste documento como o EIGRP ou o IS-IS , mas temos o OSPF, que analisaremos com o maior detalhe.

RIP

- RFC 2453 (RIP)
- RFC 7868 (EIGRP)
- RFC 7142 (IS-IS)
- RFC 2328 (OSPF)

Haver vários protocolos de encaminhamento não significa propriamente que tenhamos de escolher um apenas para aplicar num *router*. Pelo contrário, é-nos permitido executar o processamento de pacotes camada 3 através de um conjunto de protocolos de encaminhamento. Isto acontece porque os protocolos, aquando da sua execução, preenchem um conjunto de estruturas de dados armazenadas nos próprios equipamentos, cujos resultados são sumariados e rapidamente utilizados em **tabelas de encaminhamento** (em inglês *routing tables*). Estas tabelas identificam, entre muitas outras coisas, qual é o protocolo em curso para uma determinada rota existente.

tabelas de encaminhamento

Nestas tabelas de encaminhamento, o *router* apenas terá, após o processamento das rotas e escrita destas nesta tabela, de saber discriminar escolhas de rotas para endereços que lhe sejam dados como entrada em argumento. Um dos critérios que esta deverá saber respeitar é a escolha dos enderços IP, escolhendo sempre aquele com máscara maior, isto é, que seja mais específico. Por exemplo, se uma tabela de encaminhamento apresentar as seguintes rotas para o endereço 192.168.1.12 (com o mesmo protocolo de encaminhamento) - 192.168.1.0/25, 192.168.1.0/24, 192.168.0.0/23, 192.168.0.0/19 - o *router* deverá escolher o endereço 192.168.1.0/25. Se nesta discriminação houver n caminhos para o endereço dado, então o *router* deverá saber **balancear** o tráfego entre os n caminhos diferentes.

balancear

Se no exemplo dado anteriormente o protocolo de encaminhamento fosse diferente de rota para rota, então a primeira coisa que os distinguiria seria o próprio protocolo, que se faz representar de uma **distância administrativa** diferente. Como nos devemos recordar de Fundamentos de Redes (a3s1), a distância administrativa do protocolo RIP era 120 e a das rotas estáticas é 1. Comparando valores destas distâncias prefere-se sempre a rota de menor valor.

distância administrativa

Geralmente a rota é acompanhada não só da distância administrativa, como de um **custo** associado. Nos *routers* Cisco temos a representação `RIP [120/1]` para identifcar que uma determinada rota foi calculada através do processamento RIP, com distância adminstrativa 120 e custo 1. Este custo, embora possa ser alterado, tem critérios-padrão para a sua utilização dependendo do protocolo de encaminhamento em causa: por exemplo, no caso do RIP, o custo é o número de saltos, sendo que custo 16 identifica uma situação em que se atinge o infinito.

custo

Da mesma forma que o valor do custo é passível de ser alterado, o valor da distância administrativa também o é, daí que as rotas permanecem com a identificação do seu protocolo-mãe nas tabelas de encaminhamento dos *routers*. Esta troca pode ser-nos útil porque baseando-nos apenas nas distâncias administrativas, então os *routers* escolheriam sempre as rotas estáticas (de distância 1) sobre todas as outras rotas dinâmicas. A estas rotas com distâncias administrativas alteradas - mais especificamente às rotas estáticas - damos o nome de **rotas estáticas flutuantes** (em inglês *floating static routes*).

rotas estáticas flutuantes

Usam-se preferencialmente as rotas estáticas flutuantes, porque a complexidade inerente à alteração de distâncias em protocolos dinâmicos é severamente maior, que em comparação com as estáticas. Dá-se o nome de "flutuante" só é usada quando os protocolos que ocupam distâncias administrativas menores falharem.

Temos falado do RIP como protocolo-exemplo para encaminhamentos dinâmicos, mas dentro deste grupo de protocolos, o RIP faz parte de uma de duas grandes categorias de rotas dinâmicas. As rotas dinâmicas são definidas como *distance vector* ou como *link-state*. Em Fundamentos de Redes (a3s1) verificámos com detalhe o funcionamento global das rotas dinâmicas *distance vector*, onde o RIP é parte constituinte. Agora, nesta disciplina, focar-nos-emos em estudar essencialmente protocolos **link-state**.

link-state

Mas afinal qual é a diferença essencial? Ora, nos protocolos *distance vector*, cada *router* aprende redes e os seus melhores caminhos com base em informação enviada periodicamente dos seus vizinhos (informações como a rede e o seu custo para essa rede). Aqui, cada *router* determina o caminho mais curto para todas as redes através de um algoritmo assíncrono e distribuído de Bellman-Ford, como estudámos em Algoritmos e Complexidade (a2s2). Alguns outros exemplos, para além do RIP, *distance vector* são o IGRP e o EIGRP.

⊙ Richard Bellman
⊙ Lester Ford

Por outro lado, os protocolos *link-state* permitem que os routers aprendam a topologia completa da rede e usem um algoritmo centralizado para determinar caminhos mais curtos para todas as redes conhecidas. A informação necessária para construir e manter, em cada *router*, uma base de dados constante com a topologia da rede é obtida através de um processo de *flooding*, o qual ocorre somente quando há uma mudança na topologia.

Em detalhe, a operação dos protocolos *link-state* é feita do seguinte modo: primeiro, somente quando há uma mudança na topologia é que é despoletada uma mensagem especial que a sinaliza; ao enviar essa mensagem (denominada de **Link-State Advertisement** - LSA), esta é propagada entre os equipamentos vizinhos, através de um endereço *multicast* especial (no caso do OSPF); cada *router* guarda o LSA, encaminhando-o para os vizinhos, enquanto guarda os novos dados na sua base de dados, denominada de **Link-State Database** - LSDB; De seguida, através de algoritmos próprios (como o algoritmo de Dijkstra), são processados os dados preservados na LSDB e transformados numa árvore; cada *router*, neste ponto, tem a capacidade de selecionar o(s) melhor(es) caminho(s) nas suas árvores SPF, colocando-o(s) na sua tabela de encaminhamento. Exemplos deste tipo de protocolos são o IS-IS e o OSPF, que iremos estudar já de seguida. [5]

Link-State Advertisement

Link-State Database

⊙ Edsger W. Dijkstra

Introdução ao protocolo Open Shortest Path First (OSPF)

Como dito, iremos abordar, como principal exemplo de protocolo de encaminhamento *link-state*, o **OSPF** (sigla de *Open Shortest Path First*). Este protocolo encontra-se definido na norma RFC 2328 e é um protocolo que responde muito rápido a mudanças de topologia, enviando atualizações nesses casos e outras periódicas, conhecidas como atualizações *link-state*, em intervalos de tempo bastante longos, como de 30 em 30 minutos.

OSPF

↬ RFC 2328

Os *routers* que executam o OSPF colecionam informação de todos os outros *routers* na rede (ou de uma área definida na rede, como iremos ver mais à frente), calculando o melhor caminho, cada um, para todos os destinos na rede, através do algoritmo de Dijkstra.

Como já referimos, as mensagens que são transmitidas de *router* em *router*, no caso de uma mudança de topologia são as LSA. Estas reportam o estado dos *routers* e as ligações entre eles, mantendo uma sincronização constante desta informação entre todos os equipamentos de rede. Quando estes são recebidos é despoletado um novo envio para o emissor de tal mensagem com um reconhecimento da mesma - um **Link-State Acknowledgement**, LSAck - que por terem um número de sequência permitem a sua fácil correspondência. Mais, como possuem também um tempo de vida designado, cada *router* sabe sempre quando é que possui a versão mais recente de um LSA.

Link-State Acknowledgement

Identificação de equipamentos sob protocolo OSPF

Como todos os *routers* têm informação sobre todos os outros que estão na rede, então tem de haver uma forma fácil e intuitiva de reconhecê-los e identificá-los no processamento do protocolo.

Para começar, cada *router* possui um **Router ID**, vulgarmente abreviado de **RID**. Esta identificação é o maior endereço IPv4 presente no conjunto das interfaces de rede do *router*, no momento da ativação do protocolo OSPF, ou um valor administrativamente definido. Este valor aconselha-se a ser definido ou manualmente, ou através das interfaces de *loopback*, dado que se a interface com maior endereço falhar e o *router* é reiniciado, en-

Router ID, RID

tão o RID irá mudar, ao contrário do que aconteceria com a interface de *loopback*, que nunca muda.

Tendo já uma identificação e sendo o protocolo OSPF um processo que consiste no envio de mensagens aquando de mudanças de topologia, tem de haver um equipamento que seja capaz de iniciar o processamento do protocolo. A esse equipamento damos o nome de **Designated Router** (DR). O DR é sempre o primeiro *router* OSPF a fazer *boot*[3], sendo que o segundo é denominado de **Backup Designated Router** (BDR), passando a DR se o primeiro falhar.

Designated Router

Backup Designated Router

Quando um *router* entra numa rede com OSPF ativo a primeira coisa a fazer é enviar um pacote especial OSPF, denominado de **hello packet**, esperando pela resposta dos outros, com novos *hello packets*. Nestes pacotes estão definidas as várias **adjacências** entre *routers*, designando mais tarde aqueles que serão conceptualizados como **vizinhos**.

hello packet

adjacências

vizinhos

Diz-se que o ID de uma rede OSPF (**OSPF ID**) é o endereço IP da interface de rede do DR.

OSPF ID

Em redes pequenas, o conjunto das ligações entre *routers* não é muito grande (usualmente), pelo que os caminhos para destinos em particular são facilmente dedutíveis. No entanto, quando a própria dimensão da rede começa a ser cada vez maior (pensemos inclusive em termos de escalabilidade), a complexidade de ligações entre *routers* e o número de caminhos para destinos torna-se muito maior. Mesmo estando a calcular os melhores caminhos com o algoritmo de Dijkstra, o tempo para o realizar tornar-se-á bastante significativo. Isto acontece porque na base de dados do nosso protocolo OSPF (LSDB) estão as informações de todos os equipamentos presentes na rede, a qual é atualizada por cada *router*. Como a base de dados é bastante grande, espera-se da mesma forma que a tabela de encaminhamento também fique com o mesmo problema[4].

Como solução a estes problemas de dimensões, os protocolos *link-state*, geralmente, incluem o conceito de **área**, particionando o espaço em várias partes e gerindo cada uma dessas partes como um todo. Assim conseguimos tanto reduzir o número de cálculos no algoritmo de Dijkstra, como o número de mensagens LSA a serem processadas por cada *router*, ou até mesmo o tamanho das tabelas de encaminhamento - agora estas possuem apenas as rotas para dentro da sua área e uma ou outra inter-área.

área

Este conceito de área, em termos de OSPF, é definido tomando nota de uma questão: existe uma **hierarquia** formada por duas áreas - uma área denominada *backbone* e áreas regulares (ou *non-backbone*). As **áreas backbone** (em português, embora não se use tal nome, seria algo como área da coluna vertebral) é tal cuja função principal é processar e movimentar rápida e eficientemente pacotes IP, interligando-se com todas as outras áreas. Sendo sempre a área com identificação 0, tipicamente, os utilizadores nunca se encontram ligados a esta. Por outro lado, as **áreas regulares** (ou *non-backbone*) são todas as outras cuja função é mesmo ligar vários utilizadores à rede e permitir o uso dos seus recursos. Por norma, a definição destas áreas está ligada a questões geográficas. Note-se, não esquecendo, que todas as áreas regulares têm de ter a área 0 como sua vizinha, isto é, precisam de estar constantemente ligadas à área 0. Mais, estas áreas, como iremos ver mais à frente, têm vários subtipos (dentro de áreas regulares), como área *stub*, área totalmente *stubby*, área não tão *stubby* ou totalmente *stubby* não tão *stubby*. Na Figura 2.1 podemos assim ver a representação de uma rede OSPF formada por 3 áreas. Como podemos verificar duas das áreas (para além da área 0) encontram-se ligadas à *backbone*.

hierarquia

áreas backbone

áreas regulares

Na Figura 2.1 também podemos reparar que existem *routers* que assumem compromissos diferentes da rede OSPF. Por exemplo, só nesta figura podemos verificar que existem *routers* que se encontram dentro de áreas, com as interfaces todas dentro da mesma área e existem outros que têm uma interface numa área e outra interface noutra área. Ora, só estes dois exemplos definem dois de vários outros tipos de *routers* que podem co-

3 Se vários routers fizerem boot em simultâneo, então o DR será aquele com maior prioridade (parâmetro administrativamente definido), sendo o BDR o segundo. Mesmo assim, em caso de empate, é DR aquele que tiver maior RID.

4 Note-se que o protocolo OSPF não faz sumarização de endereços por definição - no entanto é configurável.

existir numa rede OSPF. Em particular aos *routers* cujas interfaces se encontram todas dentro da mesma rede damos o nome de **routers internos**. Todos os *routers* internos, na mesma área, têm de possuir a mesma base de dados *link-state* (LSDB). Áqueles que se encontram definidos dentro da área *backbone* damos também o nome de **routers backbone**, pelo que se um determinado *router* for interno e estiver dentro da área 0 é tanto interno, como *backbone* - o mesmo acontece com outros tipos.

routers internos

routers backbone

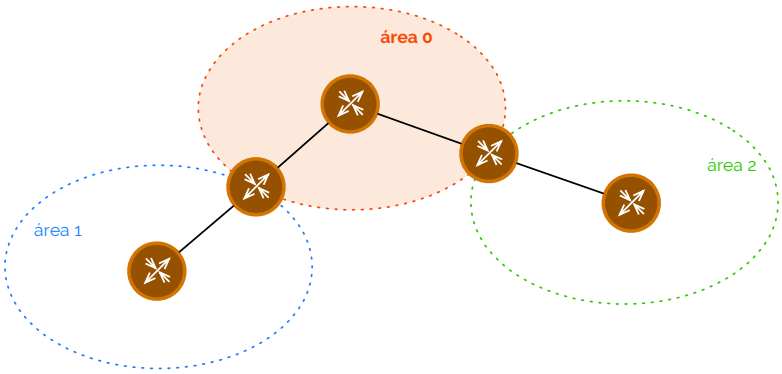


figura 2.1

Aos *routers* que têm várias interfaces em áreas diferentes, fazendo fronteira entre estas, são chamados de **router de fronteira de área** (em inglês *Area Border Router*), vulgarmente identificados pela sua abreviatura **ABR**. Estes *routers* têm a particularidade de possuir LSDBs diferentes para cada área a que estão ligados, tal como rotas para quem vai ou chega de outras áreas. Mais, como todas as áreas necessitam de estar ligadas à área 0, estes são os responsáveis por manter tal ligação. Note-se, no entanto, que estes equipamentos podem ligar outras áreas sem que seja a área 0, embora se recomende fortemente que apenas ligue duas áreas: a área 0 e outra[5].

routers de fronteira de área
ABR

Existe, no entanto, um caso particular dos ABR, que são os **ASBR**, sigla para *Autonomous System Border System* (em português **router de fronteira de sistema autónomo**). Estes *routers* têm pelo menos uma interface conectada a um protocolo de encaminhamento diferente (como outro processo OSPF[6] autónomo ou um domínio com RIP, por exemplo). Estes *routers* também são importantes porque são capazes de redistribuir rotas externas para dentro da rede, como veremos mais à frente.

ASBR
router de fronteira de sistema autónomo

Em suma, na Figura 2.2, podemos ver os vários tipos de *routers* dentro do contexto de uma rede OSPF.

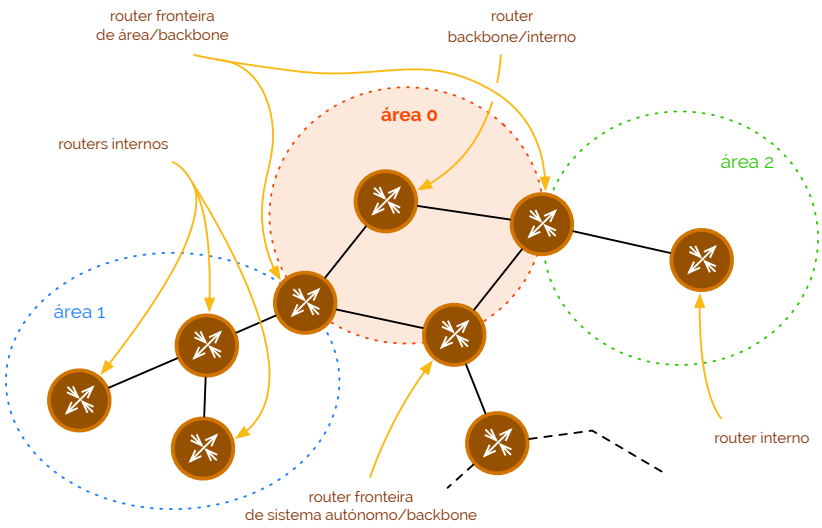


figura 2.2
tipos de routers

5 O limite de áreas ligadas num ABR são 3.

6 Um router pode executar várias instâncias de protocolo OSPF, designadas por processos. Um novo processo é completamente independente dos anteriores, podendo ser executado em paralelo.

Da mesma forma que existem diferentes tipos de *routers* também existem diferentes **tipos de áreas**. Como já vimos as áreas podem ser *backbone* (de nome 0 e serve para ligar todas as outras áreas para trocar e encaminhar informação) e *non-backbone*. Mas dentro das *non-backbone* podemos ter vários tipos, entre os quais os seguintes:

tipos de áreas

- **área normal** (*standard area*) - esta área, por defeito, aceita atualizações de ligações, tal como sumários de rede e rotas externas. As *backbone areas* também são normais;
- **área stub** (*stub area*) - esta área não aceita informação acerca de rotas externas para o sistema autónomo. Se os *routers* precisarem de saber como sair do seu sistema autónomo deverão usar uma rota estática de defeito. Estas áreas não poderão, assim, conter quaisquer ASBR[7];
- **área totalmente stubby** (*totally stubby area*) - este tipo de área é proprietária da Cisco, mas refere-se a uma tal que não aceite quaisquer rotas externas ou sumários de rede de outras áreas internas ao sistema autónomo. Se os *routers* precisarem de saber como sair do seu sistema autónomo deverão usar uma rota estática de defeito. Estas áreas não poderão, assim, conter quaisquer ASBR[7];
- **área não tão stubby** (*not so stubby area*) - vulgarmente denominado de NSSA, este tipo de área oferece benefícios semelhantes aos de uma área *stubby*, não aceitando informação acerca de rotas externas para o sistema autónomo, preferindo usar rotas por defeito para fora. No entanto, estas áreas já permitem que existam ASBRs;
- **área totalmente stubby, mas não tão stubby** (*totally stubby* NSSA) - esta área permite ASBRs, mas não aceita rotas externas ou sumários de rotas provenientes de outras áreas internas. Se os *routers* precisarem de saber como sair do seu sistema autónomo deverão usar uma rota estática de defeito.

Na Figura 2.3 podemos ver um sumário de todos os tipos de áreas (suprimimos as áreas *totally stuby* e *totally stubby* NSSA, sendo que as diferenças são a nível de comportamento, não de estrutura.

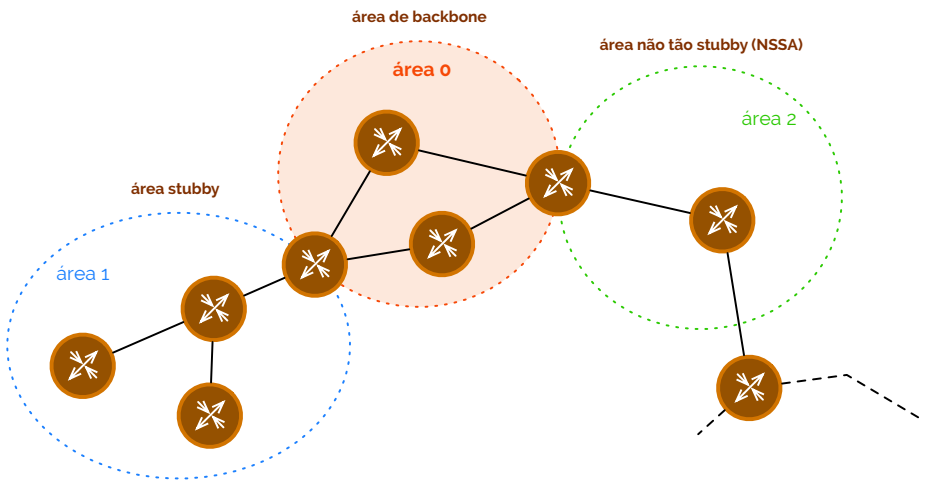


figura 2.3
tipos de áreas

Embora as áreas tenham sempre de estar ligadas com a área 0, isto não significa que estas tenham necessariamente de ser contíguas. Para contornar este problema podemos usar aquilo a que chamamos de **ligações virtuais de área** (*area virtual links*) para ligar uma área à área 0, se forem descontíguas. Na Figura 2.4 podemos ver uma representação deste conceito, onde uma área 2 se liga à área 0 através de uma ligação virtual sobre a área 1.

ligações virtuais de área

7 Há uma exceção, que é quando o ABR que liga a área stub tiver uma ligação externa, sendo assim um ABR/ASBR.

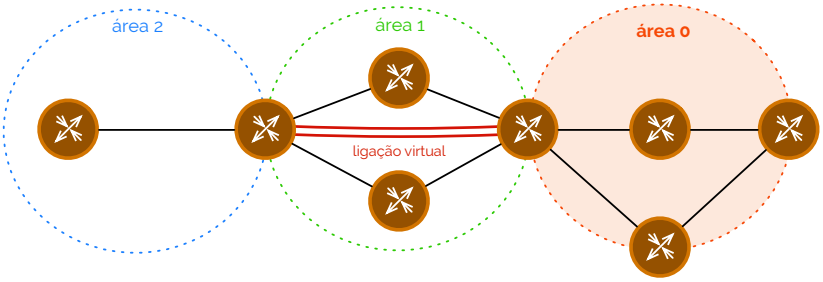


figura 2.4

Pode dar-se o caso, também, da própria área 0 estar descontígua por si, podendo ser usada uma ligação virtual para as unir, como podemos ver na Figura 2.5.

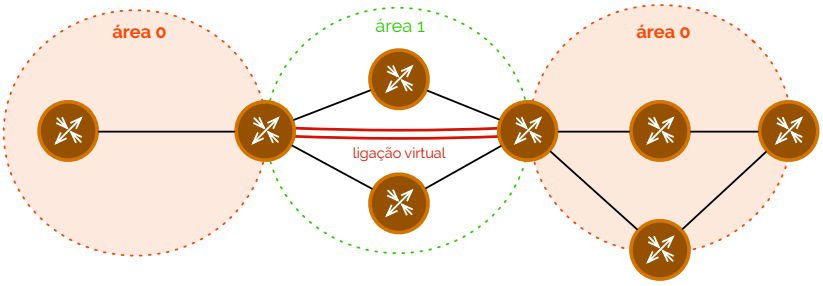


figura 2.5

Base de dados do protocolo OSPF

Como vimos todas as informações partilhadas pelos *routers* são inseridas em bases de dados que se podem encontrar em cada um destes equipamentos na rede. Tal base de dados, como também já tivemos oportunidade de referir, tem o nome de **Link-State Database**, daí a sigla LSDB.

Link-State Database

Esta base de dados está dividida em duas tabelas distintas: a tabela de *routers* e respetiva informação *link-state*, e a tabela de redes/ligações. Na primeira tabela os *routers* são identificados pelos seus RID, enquanto que na segunda as redes são identificadas pelo seu ID de rede. No Código 2.2 podemos ver um exemplo de base de dados.

código 2.2

```
          OSPF Router with ID (20.20.20.1) (Process ID 1)

        Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
20.20.20.1      20.20.20.1      40          0x8000000A 0x00E7FB 2
30.30.30.2      30.30.30.2      69          0x80000006 0x002906 2
30.30.30.3      30.30.30.3      41          0x80000007 0x00283D 2

        Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.10.10.3      30.30.30.3      40          0x80000001 0x00051C
20.20.20.2      30.30.30.2      70          0x80000001 0x00A164
30.30.30.3      30.30.30.3      15          0x80000001 0x00A91C
```

Dependendo do sistema operativo de cada *router* (neste caso estamos a usar o IOS da Cisco), podemos ver cada uma destas duas tabelas com mais detalhe. Vejamos, no Código 2.3, a tabela de *routers* - aqui, para cada *router*, podemos que informações temos sobre quem está diretamente ligado a este.

código 2.3

```
LS age: 321
Options: (No TOS-capability, DC)
LS Type: Router Links
Link State ID: 20.20.20.1
Advertising Router: 20.20.20.1
LS Seq Number: 8000000A
Checksum: 0xE7FB
Length: 48
Number of Links: 2
```

```
Link connected to: a Transit Network
 (Link ID) Designated Router address: 20.20.20.2
 (Link Data) Router Interface address: 20.20.20.1
  Number of TOS metrics: 0
   TOS 0 Metrics: 1

Link connected to: a Transit Network
 (Link ID) Designated Router address: 10.10.10.3
 (Link Data) Router Interface address: 10.10.10.1
  Number of TOS metrics: 0
   TOS 0 Metrics: 1
```

No Código 2.3 conseguimos assim saber que o *router* onde estamos a ver esta tabela tem ID 20.20.20.1 (assinalado a vermelho) e tem duas interfaces ligadas (assinalado a laranja). Com mais detalhe, temos ligações, em ambas as interfaces, para redes de trânsito (assinalado a azul), uma sendo a 20.20.20.1 e outra sendo 10.10.10.1 (assinalado a verde). Pela interface com endereço 20.20.20.1 podemos ir para o DR através de 20.20.20.2 e, pela interface com endereço 10.10.10.1 podemos ir para o DR através de 10.10.10.3 (assinalado a roxo). Ambas interfaces possuem um custo de 1 (assinalado a castanho).

No Código 2.4 podemos ver então a tabela de redes/ligações onde, para cada rede, é mostrada a informação sobre os *routers* que lhe estão diretamente ligados.

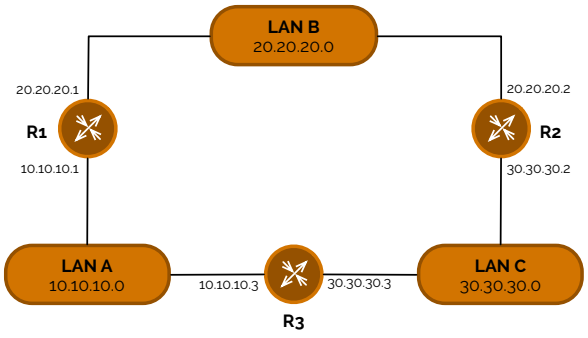
código 2.4

```
Routing Bit Set on this LSA
LS age: 483
Options: (No TOS-capability, DC)
LS Type: Network Links
Link State ID: 10.10.10.3 (address of Designated Router)
Advertising Router: 30.30.30.3
LS Seq Number: 8000001
Checksum: 0x51C
Length: 32
Network Mask: /24
  Attached Router: 30.30.30.3
  Attached Router: 20.20.20.1
```

No Código 2.4 podemos descrever agora melhor a nossa rede, do lado da interface de endereço 10.10.10.1 - que é da mesma rede do DR (assinalado a vermelho). Na verdade, o endereço do DR é mesmo 10.10.10.3, como indicado. E a esta rede está anexa (como vizinha) a rede 30.30.30.0 (representada pelo RID 30.30.30.3) e a rede 20.20.20.0 (representada pelo RID 20.20.20.1), como podemos ver a azul. Note-se que não podemos inferir a rede através dos RIDs, dado que o RID é uma mera *string* que pode ser alterada pelo administrador de rede!

Na Figura 2.6 podemos então ver a rede completa.

figura 2.6



Do lado da interface 20.20.20.1 do R1 a rede deverá estar então descrita da seguinte forma (Código 2.5), na tabela de redes/ligações:

código 2.5

```
Routing Bit Set on this LSA
LS age: 483
Options: (No TOS-capability, DC)
LS Type: Network Links
Link State ID: 20.20.20.2 (address of Designated Router)
Advertising Router: 30.30.30.2
LS Seq Number: 8000001
Checksum: 0xA164
Length: 32
```

```
Network Mask: /24
  Attached Router: 30.30.30.2
  Attached Router: 20.20.20.1
```

Pacotes e operação do protocolo OSPF

As várias mensagens que são partilhadas no protocolo OSPF estão inseridas num conjunto de pacotes que são distribuídos na rede por cada *router*.

Existem vários tipos de pacotes. Alguns deles já referimos, como o pacote *hello* ou até o conteúdo LSA. Assim, podemos já começar por estes. Como já sabemos o LSA, sigla para *Link-State Advertisement* é responsável por transmitir informação acerca do estado da rede aquando de uma alteração de topologia. Esta mensagem em termos globais é muito simples, mas acontece que é descrita em 11 tipos de mensagens diferentes. Vejamos [5]:

- **Tipo 1** (LSA de *router*) - todos os *routers* geram partilhas de dados nas suas ligações para cada área a que pertence. Estes avisos descrevem o estado das ligações do *router* à área e são levados por *flooding* dentro de um espaço restrito. Todos os tipos de LSAs têm cabeçalhos de 20 bytes, sendo um dos seus campos o *link-state* ID, neste caso, o RID;
- **Tipo 2** (LSA de rede) - os DR geram avisos de rede (*advertisements*) para redes de múltiplo acesso que descrevem o conjunto de *routers* ligados a estas. Estes avisos descrevem o estado das ligações do *router* à área e são levados por *flooding* dentro de um espaço restrito. Neste caso o *link-state* ID é o endereço da interface do DR;
- **Tipo 3** e **4** (LSA sumário) - os ABR geram *advertisements* com sumários de ligações, que descrevem rotas inter-área, no tipo 3, para as redes da área e, no tipo 4, rotas para os ASBRs;
- **Tipo 5** (LSA exterior ao sistema autónomo) - os ASBRs geram *advertisements* de ligações externas ao sistema autónomo que são levados para todo o lado, por *flooding*, exceto a qualquer tipo de área *stub*;
- **Tipo 6** (LSA *multicast*) - estes LSAs são usados em aplicações OSPF *multicast*;
- **Tipo 7** (LSA para NSSAs) - estes LSAs são usados em NSSAs;
- **Tipo 8** (LSA para atributos externos de BGP) - estes LSAs são usados para interligar OSPF com as rotinas de BGP, como iremos abordar em Arquitetura de Redes Avançada (a4s1);
- **Tipos 9, 10** e **11** (LSAs opacos) - estes LSAs estão guardados para futuras alterações ao OSPF para distribuição de informação específica para aplicações através de um domínio OSPF. Os mecanismos normais de LSBD *flooding* são usados para a distribuição de LSAs opacos. O tipo 9 não permite que os LSAs passem para além da sua rede ou sub-rede. O tipo 10 não permite que os LSAs passem para além das suas fronteiras ou área associada. Por fim, o tipo 11 permite que os LSA passem pelo sistema autónomo completo. [6]

RFC 5250

hello packet

Um segundo pacote que existe no OSPF é o **hello packet** que permite a descoberta de vizinhanças e cria adjacência entre *routers*. Este pacote contém informações como o RID (um número de 32 bits que identifica unicamente um *router*), intervalos *hello*/*dead* (o intervalo *hello* é o quão frequente, em segundos, um *router* deve enviar pacotes *hello* - 10 segundos é o valor por defeito - e intervalo *dead* é a quantidade de tempo que um *router* espera pela mensagem de um *router* até que este seja declarado como inoperacional - por defeito é 40 segundos, quatro vezes mais que o tempo de *hello*), vizinhos (lista de *routers* adjacentes com quem o *router* efetuou já comunicações bi-direcionais), ID da área, prioridade do *router* (número de 8 bits que indica a prioridade do *router*, especialmente para a eleição do DR e BDR), palavra-passe de autenticação (se a autenticação estiver ativa os dois *routers* precisam de trocar informação) e *flag* em caso de área *stub* (indica o caso em

que é necessário reduzir as atualizações de rotas, substituindo-as por rotas por defeito. Para que dois *routers* estabeleçam uma relação de adjacência é necessário que concordem com o intervalo de *hello*/*dead*, com o ID da área, com a palavra-passe de autenticação e com a *flag* de área *stub*.

As outras mensagem que existem são as **DBD** (de *Database Description*, em português descrição de base de dados), servindo para verificar a sincronização de dados guardados entre *routers*, os **LSR** (de *Link-State Request*) para pedir registos específicos de *link-state* a outro *router*, os **LSU** (de *Link-State Update*) para enviar registos pedidos - resposta a LSRs - e os **LSAck** que servem para reconhecer os pacotes recebidos.

DBD

LSR
LSU
LSAck

Em termos gerais, no entanto, os pacotes têm todos uma estrutura idêntica. Através da identificação do protocolo com o número 89 para o OSPF, o pacote OSPF tem a estrutura da Figura 2.7.

figura 2.7
pacote OSPF

No campo `DADOS` do pacote OSPF a informação muda consoante o tipo de pacote a que nos referimos: num pacote *hello* colocam-se a lista de vizinhos; num pacote DBD coloca-se um sumário de LSDBs; num pacote LSR coloca-se o tipo de LSU necessário e o RID de quem tem o LSU necessário; num pacote LSU colocam-se as entradas LSA completas, sendo que podem ser enviadas múltiplas num só pacote; e no LSAck este espaço é deixado vazio.

Para a descoberta das rotas de uma rede consideremos inicialmente a seguinte rede, onde o R1 pretende conectar-se à rede com o DR, que está ligado a 172.16.6.0/24 (com o endereço 172.16.6.1) - Figura 2.8.

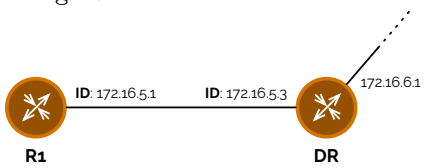


figura 2.8

Em termos de operação para a descoberta de rotas, uma relação *master*/*slave* deve ser criada entre cada *router* e o seu DR e BDR adjacentes, embora somente o DR troque e sincronize a informação de *link-state* com os *routers* que lhe pediram adjacência - esta fase completa-se com uma troca, assim sendo, de pacotes *hello*. Tendo completado esta fase, então agora segue-se uma troca de um ou mais pacotes DBD que incluem informações acerca dos cabeçalhos de entradas LSA que aparecem nos LSDBs do *router*, estas, que podem ser referentes tanto a ligações, como a redes. Cada um destes cabeçalhos inclui informação acerca do tipo de *link-state*, o endereço do *router* que faz o anúncio, o custo da ligação e um número de sequência, este, que serve para aferir o quão recente é a informação armazenada. De seguida, com pacotes LSAck, ambas as partes reconhecem os DBDs recebidos, seguindo-se para a parte em que se o DBD tiver uma entrada mais atual, então envia-se um LSR a pedir as informações mais recentes, que vêm num LSU, voltando a trocar um LSAck para confirmar a receção. Na Figura 2.9 podemos ver uma versão desta operação mais sumária.

O processo de *flooding* em OSPF é feito exclusivamente em *multicast* com o envio de um LSU em caso de mudança de topologia, que leva uma entrada LSA atualizada e com um número de sequência devidamente incrementado, para o endereço 224.0.0.6 - endereço de todos os DRs e BDRs[8]. Note-se que cada LSU poderá levar um ou mais LSAs!

Tendo o DR recebido o LSU e processado-o, então envia um LSAck e faz *flood* o LSU para os outros *routers* com 224.0.0.5. O *router* agora já pode atualizar a usa base de dados *link-state* usando o LSU que inclui o LSU atualizado, recalculando todas as rotas através do algoritmo de Dijkstra, mostrando-se pronto passado uns instantes.

8 Em ligações ponto-a-ponto este endereço seria 224.0.0.5

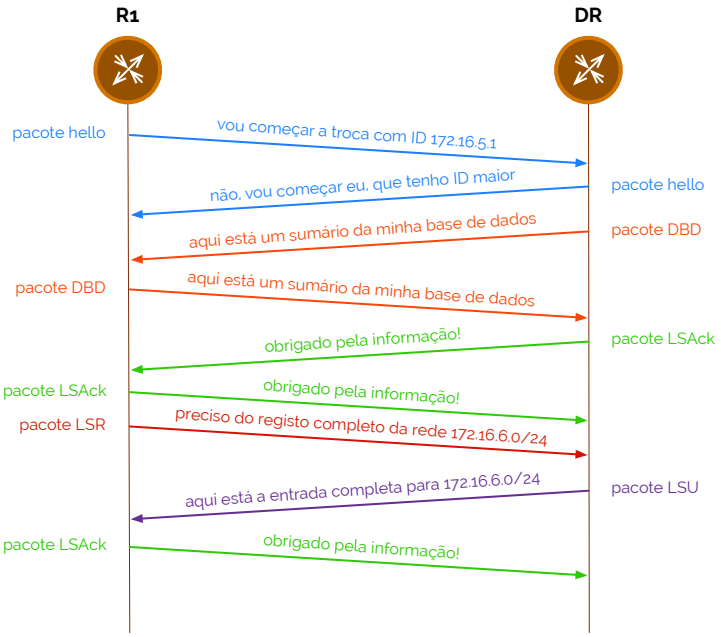


figura 2.9

Portanto, como já devemos perceber até agora, os LSAs estão incluídos nos LSUs. Quando um *router* recebe um LSU verifica primeiro se a entrada descrita no LSA já existe na base de dados. Caso exista, então há que verificar se o número de sequência é o mesmo, descartando o LSA caso seja, e verificando se o número de sequência é maior. Caso não seja o *router* deverá enviar um LSU para a origem deste com a informação mais atual. Caso contrário, isto é, caso seja um número de sequência maior, faz-se o mesmo do caso em que a entrada não está na base de dados - adiciona-se a entrada na base de dados, envia-se o LSAck, faz-se o *flood* do LSU e executa-se o algoritmo de Dijkstra para recalcular a tabela de encaminhamento.

Custos de caminhos e rotas externas sob o protocolo OSPF

Em Fundamentos de Redes (a3s1) estudámos o protocolo de encaminhamento RIP no qual o custo era feito contando o número de saltos desde uma origem até um destino. No caso do OSPF o mesmo raciocínio não pode ser aplicado, uma vez que o **custo** é calculado somando o custo de cada interface de saída ao longo de um caminho. Note-se, não obstante, que as ligações diretas não contam com custo porque não interagem com protocolos de encaminhamento nas tabelas de encaminhamento.

custo

Consideremos assim a Figura 2.10 onde temos um exemplo bastante básico de OSPF para calcular os custos, onde se expõem todos os caminhos possíveis com os respetivos custos (os números das interfaces são os respetivos custos). [5]

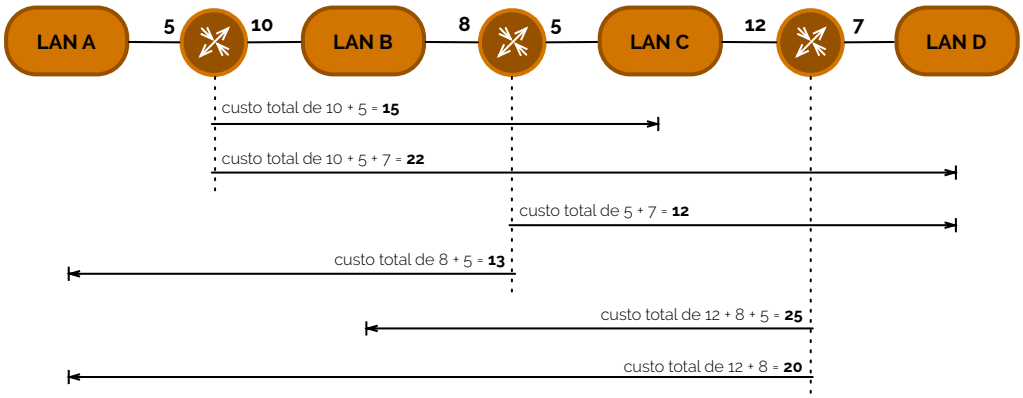


figura 2.10

Mas as rotas OSPF não são exclusivamente dentro de um sistema autónomo. Sempre que temos rotas provenientes de ASBRs temos um conceito já algumas vezes referido atrás chamado de **rotas externas**. Em OSPF podemos ter dois tipos de rotas externas: as E1 e as E2. Basicamente o que as distingue é manipulação de custos: por defeito as rotas externas são do tipo **E2** o que significa que o custo acrescido por ser rota é apenas o custo externo do domínio de OSPF. Se houver apenas um ASBR a transmitir esta rota, então até vale a pena a sua escolha, caso contrário não estamos a discriminar uma linha de trânsito específica, dentro da nossa rede, para os pacotes de/para o exterior passarem. Para o fazer podemos usar o tipo **E1**, que já usa uma combinação (soma) do custo exterior com o custo interno do domínio OSPF. [5]

rotas externas

E2

E1

As rotas em OSPF não são sumariadas por defeito, pelo que precisam da sua configuração para tal. Aconselha-se a sumarização de forma a reduzir o número de anúncios de ligações ao longo de toda a rede, passando a anunciar apenas os prefixos de rede[9]. Por esta mesma razão há que ter o cuidado de atribuir endereços contíguos às redes das áreas, o que permitirá que se passse de tabelas com listas de rotas OSPF para poucas OSPF de inter-área (nos *routers* da Cisco fazem-nas representar de `IA` ao invés do `O` convencional para OSPF). [5]

Encaminhamento com OSPF em IPv6 (OSPFv3)

Com base clara nos desenvolvimentos anteriores de OSPF até à sua versão 2, criou-se a versão 3 (**OSPFv3**) para ser aplicado em redes IPv6. Este protocolo de encaminhamento usa assim os endereços *multicast* FF02::5 para as transmissões de mensagens normais e o FF02::6 para comunicação com os DRs - analogamente ao que acontecia em IPv4.

OSPFv3

↳ **RFC 5838**

Ao invés de correr sobre uma sub-rede, agora corre sobre ligações, podendo haver várias instâncias por ligação e usando endereços *link-local* como endereços de origem de equipamentos IPv6.

Com esta adenda ao protocolo de encaminhamento também foram alterados alguns dos tipos LSA:

- **Tipo 8** (LSA de ligação) - este tipo serve para informar os *routers* vizinhos dos endereços *link-local* e dos prefixos IPv6 da ligação;
- **Tipo 9** (LSA de prefixo inter-área) - serve para associar prefixos IPv6 com uma rede ou *router*.

Mais, as regiões de *flooding* foram agora generalizadas, havendo *flooding* para endereços *link-local*, para áreas ou para sistemas autónomos num todo.

Isto tudo foi possível porque a codificação do tipo de LSA aumentou de tamanho para 16 bits, incluindo as regiões de *flooding*.

Redistribuição de rotas (técnicas e problemas)

Anteriormente referimos que, por questões de conveniência, algumas partilhas de rotas não nos são muito interessantes - dá-se o exemplo de rotas de VLANs end-to-end em VLANs do núcleo. Para evitar este tipo de situação usamos o conceito de **interfaces passivas**.

interfaces passivas

O contrário também pode ser feito - ao invés de cortarmos a partilha de mensagens de rotas numa direção, podemos querer que algumas rotas sejam partilhadas num diferente processo de encaminhamento. Por outras palavras dizermos que queremos **redistribuir rotas**.

redistribuir rotas

9 Os ABR sumariam os LSAs do tipo 3, enquanto que os ASBR sumariam LSAs do tipo 5.

Esta redistribuição pode ser feita de um processo de encaminhamento A para B e vice-versa, mas há alguns problemas que podem acontecer, como perder as métricas numa redistribuição ou criar situações de ciclos. Em relação ao primeiro problema não há uma solução ótima para a sua resolução, no entanto, quanto ao segundo problema eis uma razão pela qual não se aconselha a redistribuição em duas direções em simultâneo.

Para efetuar uma boa redistribuição primeiro há que verificar se a nossa arquitetura segue o padrão onde existe um protocolo de encaminhamento em curso no núcleo da nossa rede e um protocolo de encaminhamento diferente na fronteira da nossa rede.

Existem depois várias redistribuições passíveis de serem feitas. Vejamos:

- redistribuir uma rota estática de defeito do núcleo para a fronteira do nosso sistema autónomo e redistribuir as rotas da fronteira para o núcleo. Esta técnica previne o *feedback* de rotas, o encaminhamento subótimo e os ciclos;
- redistribuir múltiplas rotas estáticas do núcleo para a fronteira e redistribuir as rotas da fronteira para o núcleo. Esta técnica funciona apenas se houver um único ponto de redistribuição;
- redistribuir rotas do núcleo para a fronteira tendo preocupação em filtrar rotas inapropriadas. Por exemplo, quando existem múltiplos *routers* de fronteira, as rotas redistribuídas não deverão ser redistribuídas de volta à fronteira, desde o núcleo, em qualquer ponto de redistribuição;
- redistribuir todas as rotas do núcleo para a fronteira, e vice-versa, e modificar a distância administrativa associada com as rotas de redistribuição, para que não sejam selecionadas quando existem várias rotas para o mesmo destino. [5]

3. Endereçamento IPv4 e IPv6

A área de redes é muito análoga a muitas das nossas ações no dia-a-dia. Os transportes que fazemos, os caminhos que escolhemos para chegar a um local, ... porque no fundo todos os dias trabalhamos com muitas redes, embora no sentido mais lato possível, em comparação com o nosso estudo. Mas nós não nos podemos ficar apenas pela forma como nos movemos e no que pensamos no entretanto - temos de saber caraterizar a forma como identificar os locais, principalmente aqueles de onde saímos e aqueles para onde vamos.

Nós, todos os dias, usamos um mecanismo que até à data se mostrou infalível para identificar locais - chamamos-lhe **endereços**. E fazer o endereçamento destes, por enquanto, passa por dizer uma rua, uma porta, um código-postal e uma localidade. E nas nossas redes? Será que conseguimos fazer isso?

endereços

Em Fundamentos de Redes (a3s1) já vimos que era possível fazê-lo, através de endereçamentos na camada 3 do modelo OSI - a camada de rede.

Abstração de endereçamento

As pessoas que vivem na nossa rua (ou no nosso prédio, como preferirem) têm coisas em comum connosco: para além de partilharem a mesma rua (ou prédio), têm uma casa (ou apartamento) que é única(o). No entanto a rua (ou prédio) é de todos e às tantas até existem outras ruas (e prédios) semelhantes ao nosso.

Quando queremos enviar uma mensagem para as pessoas destas instâncias podemos querer mandar para uma pessoa em particular, para todas, para um grupo de pessoas (como as pessoas da minha rua (prédio) somente) ou até para só uma dessas pessoas!

Em redes, analogamente, temos também essas opções para identificar os objetos recetores de mensagens, pela mesma ordem, denominando-os de **unicast**, **broadcast**, **multicast** e **anycast**.

unicast, broadcast multicast, anycast

Aqui só estamos a tentar relembrar, mas nos apontamentos da disciplina de Fundamentos de Redes (a3s1) podemos ver melhor os quatro conceitos e de que modo é que se estabelecem os vários domínios.

Recordar o endereçamento IPv4

Contrariamente às moradas extensas que usamos no quotidiano, em redes usamos números para identificar os vários componentes de uma rede. Uma das organizações numéricas mais usadas é conhecida por **endereçamento IPv4**, que é um número identificador de 32 bits, que codifica um prefixo identificador de rede e um número do terminal.

endereçamento IPv4

A sua representação é feita através de um formato denominado de **dotted decimal notation**, onde cada byte é denotado por um número decimal no intervalo de 0 a 255. Exemplos de endereços são o 10.0.0.0 ou o 192.168.1.0.

dotted decimal notation

Os vários endereços, representados por quatro números decimais, tinham uma classificação própria nos anos '80, onde através dos primeiros 1 a 3 bits se designavam um de três espaços de endereçamento existentes até então: se o primeiro bit fosse '0', então dizia-se ser de **classe A**; se os primeiros dois bits fossem "10", então dizia-se ser de **classe B**; e se os primeiros três bits fossem "110", então dizia-se ser de **classe C**.

classe A

classe B, classe C

No início dos anos '90 começaram a surgir problemas de atribuição de endereços IPv4 e a solução foi criar uma nova organização para o espaço de endereçamento, terminando o significado de classes - criou-se assim o **CIDR** (acrónimo de *Classless Interdomain Routing*). O novo que o CIDR trouxe (numa das suas últimas versões) foi o facto do prefixo já não estar ligado ao conceito de classes, que o bloqueava de ser de outro tamanho que não 8, 16 ou 24. Agora, com o CIDR, é possível criar prefixos variáveis, criando assim novos conceitos - o de **sub-redes** e de **máscara**.

CIDR

↬ **RFC 4632**

sub-redes, máscara

As sub-redes, com a ajuda das máscaras, são fruto da possibilidade da segmentação de blocos CIDR a partir de endereços dados a uma empresa ou corporação. Com isto as empresas puderam contar com a capacidade de subdividirem o seu espaço físico de endereçamento em muitos mais espaços lógicos, como já tivemos oportunidade de ver anteriormente.

Para mais informações acerca do endereçamento IPv4, por favor, consuta os apontamentos de Fundamentos de Redes (a3s1).

Recordar o endereçamento IPv6

Depois de alguns problemas em reajustar o endereçamento IPv4, dadas as questões de pouco espaço de endereçamento existente, equipas de investigação começaram a tentar criar um novo método de endereçamento e com ele tentaram também conjugar algumas soluções para outros problemas existentes na rede, como iremos verificar.

A primeira grande alteração foi propriamente o espaço de endereçamento - passámos de ter 32 bits para representação de endereços IP para termos 128 bits. Dado o seu enorme comprimento decidiu-se representar estas quantidades por via de dígitos hexadecimais, em grupos de 4, separados por dois pontos ":". Um exemplo de endereço IPv6 é 2001:0db8:0000:1462:dddd:087c:9982:225c = 2001:db8::1462:dddd:87c9982:225c.

↬ **RFC 6052**

Estes endereços já são usados por alguns serviços e utilizadores, no entanto, dado que ainda há muitas máquinas com o endereçamento IPv4 e a mudança ainda é muito dispendiosa.

Para mais informações acerca do endereçamento IPv6, por favor, consuta os apontamentos de Fundamentos de Redes (a3s1).

Modelos e tipos de endereçamento IPv6

Com o endereçamento IPv6 uma interface mais facilmente pode possuir múltiplos endereços. Mas nem todos os endereços têm o mesmo âmbito e espetro de atuação: existem endereços *link-local* (que apenas são válidos dentro da mesma LAN ou ligação), *unique-local* (que são válidos dentro do próprio domínio privado e não podem ser usados na Internet) e endereços globais. Cada um destes endereços tem um **tempo de vida**.

tempo de vida

Existem também vários **tipos de endereços IPv6**, entre os quais o *unicast* - que é o endereço de uma só interface (um para um) -, *multicast* - que é um endereço de um conjunto de interfaces (um para muitos) - e *anycast* - que é um endereço de um conjunto de interfaces (um para um de muitos). Note-se que já não existem endereços *broadcast*.

tipos de endereços IPv6

Estes tipos de endereços têm prefixos muito próprios:

- **endereços globais** - começam por '2';
- **endereços link-local** - FE80::/10;
- **endereços unique-local** - FC00::/8 ou FD00::/8;
- **endereços multicast** - FF00::/16

Os endereços **link-local IPv6** são usados para comunicações locais entre dois dispositivos IPv6 e para efetuar o cálculo dos saltos (*next hops*), e é automaticamente criado quando o IPv6 é ativado. A estrutura deste tipo de endereços pode ser vista na Figura 3.1

link-local IPv6

figura 3.1

Como podemos ver na Figura 3.1 os primeiros 10 bits já estão definidos (sendo que o prefixo de rede para endereços *link-local* é FE80::/10. Os seguintes 54 bits podem ser preenchidos com zeros ou com outro tipo de valor manualmente configurável. Os últimos 64 bits são correspondentes à própria identificação da interface, cujo preenchimento falaremos mais à frente.

Os endereços **unique-local IPv6** são usados para comunicações locais e para VPNs entre *sites* e só pode ser usado dentro do mesmo sistema autónomo. A sua estrutura é a visível na Figura 3.2, onde os primeiros 8 bits são respetivos ao prefixo de rede acima, podendo ser FC00::/8 ou FD00::/8.

unique-local IPv6

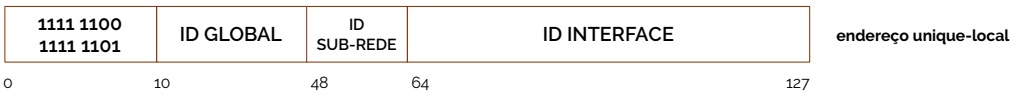


figura 3.2

Os campos seguintes são 40 bits para um identificador global (dentro do sistema autónomo) e um identificador da própria sub-rede onde este se instala (em 16 bits).

Por último, dos endereços *unicast* só nos falta mesmo referir o caso do endereço **global IPv6**. Na Figura 3.3 temos uma representação da sua estrutura, esta, mais complexa que as anteriores. Uma vez que estes endereços podem ser usados no contexto da Internet, é importante que várias entidades tenham campos específicos para preencherem, de modo a que o espaço de endereçamento não se intersete muitas vezes de forma indesejável.

global IPv6

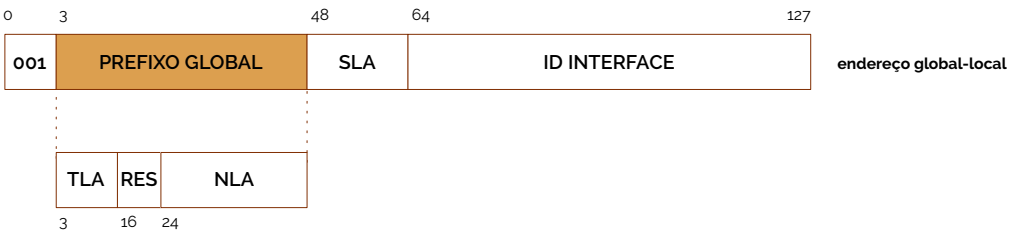


figura 3.3

Na estrutura acima descrita (Figura 3.3) temos várias partes que necessitam de uma explicação. O endereço em si é divido em três partes essenciais: a parte do prestador de serviço (primeiros 48 bits), a parte correspondente aos *sites* (secção **SLA**, sigla de *Site-Level Aggregation identifier*) e a parte final, correspondente ao próprio terminal. Na primeira secção temos o prefixo habitual dos endereços globais (2000::/4), seguido do prefixo global de encaminhamento, que são 45 bits constituídos por três secções: o **TLA** (13

SLA

TLA

bits para o *Top-Level Aggregation*), o **RES** (de reservado, 8 bits que deverão manter-se a zero) e o **NLA** (*Next-Level Aggregation identifier*, de 24 bits). Estes três campos servem para hierarquizar a localização da definição deste endereço. Quase como um diretório, este endereço encontra-se indexado em TLA/NLA/SLA.

RES
NLA

Os endereços **anycast** são endereços atribuídos a conjuntos de interfaces, pelo que tipicamente pertencem a conjuntos de nós numa rede. Um pacote que é enviado para um destes endereços é entregue à interface mais próxima (determinada pelo encaminhamento e temporização) e esta interface tem de ser de um *router*, uma vez que não é entregue a mais nenhum equipamento de rede como terminais.

anycast

Os endereços **multicast** tem o prefixo FF00::/8 e têm a estutura da Figura 3.4.

multicast

figura 3.4

O `TEMPO DE VIDA` é um campo que é preenchido com valor '0' se se pretender ser permanente e '1' se se pretender ser temporário. Já a meia-palavra seguinte descreve o âmbito da aplicação do *multicast* sendo '1' caso se pretenda aplicar a um nó da rede, '2' a uma ligação, '5' a um *site*, '8' a uma organização e 'E' à globalidade dos endereços.

Já o campo que ocupa os últimos 112 bits do endereço *multicast* pertence à identificação do grupo de atuação, sendo que é possível, com este, fazer um mapeamento com um endereço MAC - o endereço MAC são os últimos 32 bits do ID de grupo concatenados com 0x2121 à esquerda. Na Figura 3.5 mostra-se essa correspondência.

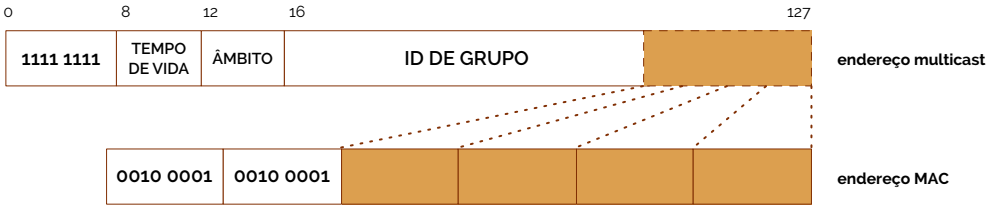


figura 3.5

Alguns usos específicos de endereços *multicast* estão listados abaixo:

- FF02::2 - endereço para todos os *routers*;
- FF02::5 - endereço para todos os equipamentos com OSPF ativo;
- FF02::6 - endereço para todos os DRs em OSPF;
- FF02::9 - endereço para todos os *routers* com RIPng ativo;
- FF02::1:2 - endereço para todos os agentes DHCPv6.

A parte respetiva ao identificador da interface nos endereços IPv6 pode ser feita de vários modos. Contrariamente ao IPv4, com o IPv6 é possível atribuir um endereço automaticamente a um terminal através de um conjunto de mecanismos como o **EUI-64**.

EUI-64

O mecanismo EUI-64 faz parte da auto-configuração do IPv6 (*stateless*) e permite expandir os 48 bits do endereço MAC de um terminal inserindo a palavra 0xFFFE no meio e fazendo assim os 64 bits de endereço final IPv6, respetivos à identificação do terminal.

No entanto existem algumas alternativas ao EUI-64, entre as quais a geração de um número pseudo-aleatório ou até mesmo o **DHCPv6**. Mais à frente voltaremos à questão do DHCPv6, mas primeiro analisemos que mensagens é que são usadas no protocolo IPv6.

DHCPv6

Mensagens em IPv6 (ICMPv6)

Embora não pareça que faça parte do protocolo IPv4, juntamente com as outras melhorias do protocolo para o IPv6 está a integração nativa do protocolo de rede ICMP (mas de forma integral). Tendo sido normalizado no RFC 4443, o **ICMPv6** possui as mesmas funcionalidades do ICMP, mas substitui o protocolo de obtenção de endereços físicos

ICMPv6
‹ RFC 4443

ARP pelo melhorado **NDP** (sigla de *Network Discovery Protocol*), os terminais usam agora o ICMPv6 para efetuar a auto-configuração de endereços, verificar duplicação de endereços com o mecanismo **DAD** (*Duplicate Address Detection*) e para testar a disponibilidade dos equipamentos vizinhos.

NDP

DAD

Para fazer esta última tarefa usa-se uma mensagem especial denominada de **neighbor discovery**, que usa pacotes ICMPv6 lançados através da interface com endereço *link-local* e com um número máximo de 255 saltos. Basicamente consiste num cabeçalho IPv6 com um cabeçalho ICMPv6, um cabeçalho de *neighbor discovery* e as respetivas opções. Existem cinco tipos de mensagens de descoberta de vizinhança, entre as quais o tipo 133 chamada *router solicitation*, o tipo 134 chamado *router advertisement*, o tipo 135 chamado *neighbor solicitation*, o tipo 136 chamado *neighbor advertisement* e o tipo 137 denominado *redirect*. Vejamos cada um com mais detalhe.

neighbor discovery

A mensagem de **Router Solicitation** (em português solicitação de *router*, vulgarmente referida como RS) acontece quando um terminal envia uma questão para a rede sobre a presença de um *router* numa ligação, enviando esta a todos os *routers* através do endereço *multicast* para o efeito, com endereço de origem sendo o *link-local* (ou até mesmo nenhum).

Router Solicitation

A mensagem de **Router Advertisement** (em português anúncio de *router*, vulgarmente referida como RA) é enviada regularmente pelos *routers* ou em resposta a um RS, que inclui informação para possível auto-configuração, um nível de preferência para cada endereço de *router* anunciado e um campo de "tempo de vida", conforme vimos anteriormente.

Router Advertisement

A mensagem de **Neighbor Solicitation** (em português solicitação de vizinhança, vulgarmente referida como NS) é usada para descobrir endereços *link-layer* de um determinado nó de uma rede IPv6.

Neighbor Solicitation

A mensagem de **Neighbor Advertisement** (em português anúncio de vizinhança, vulgarmente referida como NA) é usada para responder a um NS ou para informar de uma mudança de endereço *link-layer* de um determinado nó da rede.

Neighbor Advertisement

Por fim, a mensagem de **Redirect** (em português redireção) é usada pelos *routers* para sinalizar um reencaminhamento de um pacote para uma rota melhor.

Redirect

Mecanismo de auto-configuração IPv6 e DHCPv6

Como já fora referido anteriormente, o IPv6 está capacitado de mecanismos de **auto-configuração**. O que vimos até agora foi o mecanismo nativo deste protocolo a que damos o nome de **stateless**, onde o próprio nó que se liga à rede consegue atribuir a si próprio um endereço (daí não haver um conjunto de estados para a obtenção de endereços).

auto-configuração

stateless

Por outro lado, mecanismos **stateful** como o DHCP como o conhecemos do IPv4, também podem ser usados em IPv6. Na verdade, o protocolo DHCPv6 (DHCP na sua versão para IPv6) é ligeiramente diferente da de IPv4.

stateful

RFC 3315

No fundo, com o DHCPv6, se um cliente pretender receber uma configuração, este enviará uma mensagem *multicast* à procura de um servidor DHCP - aqui usam-se mensagens de solicitação e de anúncio. De seguida, o cliente deverá responder questionando os parâmetros de um servidor disponível que lhe responderá com a informação requirida através de uma mensagem *reply* (à *request* enviada pelo cliente).

O mecanismo de **relay** usado pelo IPv4 não se aplica em DHCPv6. Aqui o agente de *relay* irá encapsular a mensagem recebida do cliente DHCPv6 diretamente ligado. [4] De seguida esses pacotes serão encaminhados, encapsulados, para o servidor DHCPv6. No sentido contrário, o agente de *relay* irá desencapsular os pacotes recebidos do servidor central.

relay

Para além do DHCPv6 existem outros mecanismos de auto-configuração, mas este é, de facto, considerado o mais fiável e mais usado.

Conceito de túnel e tráfego de pacotes em túneis

Por vezes, para resolvermos problemas de encaminhamentos não pretendidos ou até mesmo bloqueios inesperados no envio de pacotes para determinados locais, dar-nos-ia algum jeito que pudessemos mascarar, de alguma forma, os pacotes que enviamos.

túneis

Os **túneis** servem precisamente esse papel, para garantir que um pacote que atinja um nó de uma rede seja levado para uma rede secundária em específico, independentemente dos encaminhamentos em processamento intermédios. Mais, os túneis também garantem a entrega de um pacote a um nó remoto mesmo quando os nós das redes intermediárias não suportam os protocolos que neles seguem, definindo um canal virtual que apenas adiciona funcionalidades de transporte de dados, de forma a poder fornecer qualidades de serviço totalmente diferenciadas, requisitos de segurança ainda melhores e mais reforçados e um encaminhamento mais otimizado.

Não se pense que se criam mesmo "túneis" na verdadeira aceção da palavra. A palavra "túnel" aparece porque no fundo é criada uma forma de mascarar os pacotes, adicionando-lhes um cabeçalho extra de um novo protocolo de rede, como IPv4 ou IPv6 - este é o "túnel". Visualmente, um pacote que esteja em túnel tem a aparência da Figura 3.6.

figura 3.6

Numa rede real, o que acontece é o seguinte - vejamos a Figura 3.7 onde temos um conjunto de redes e queremos passar um pacote de PC A para o PC B. No meio de tantas redes, os protocolos de encaminhamento não nos dão garantias absolutas de que o nosso pacote irá chegar a PC B. Assim, criamos um túnel do R1 para o R5. Na Figura 3.8 podemos ver o pacote e os endereços colocados em cada um dos cabeçalhos.

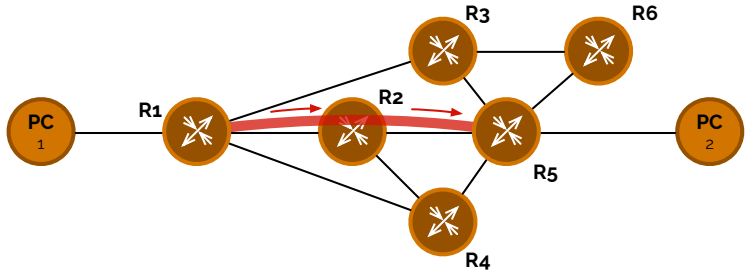


figura 3.7

figura 3.8

interfaces virtuais de túnel

A criação de um túnel exige que o nosso equipamento permita a utilização de **interfaces virtuais de túnel** (VTI, de *Virtual Tunnel Interface*), que são construções lógicas que criam uma interface de rede virtual que podem ser manipuladas como outra qualquer interface. Um túnel não precisa de ter qualquer endereço IP, para além daqueles que já definem o *router* do *end-point*. No entanto, grande parte das implementações exigem que se assine uma interface real à interface túnel, de forma a que se ative o processamento IP nessa mesma interface.

Assim uma VTI apenas precisa de um identificador (*tunnel 1*, *tunnel 2*, ...), uma interface real IP assinada, um modo de túnel especificado (mais à frente abordamos os vários tipos de túnel), uma origem, um destino e outras opções.

Mais uma vez aconselha-se o uso das interfaces de *loopback* para a integração de túneis, uma vez que estas nunca vão abaixo a não ser que o próprio equipamento também se desligue.

Como indicado antes, existem vários **tipos de túnel**, entre os quais túneis que permitem a utilização de protocolos que podem resolver problemas de compatibilidade de transmissão do pacote original.

tipos de túnel

O túnel mais comum e simples é o **IPv4 sobre IPv4**, isto é, trata-se de um pacote normal IPv4 que é transportado com um cabeçalho igualmente de IPv4. A mesma coisa acontceria com o túnel **IPv6 sobre IPv6**, que faz precisamente o mesmo, mas com IPv6.

IPv4 sobre IPv4
↬ RFC 2003
IPv6 sobre IPv6

Caso tenhamos um problema de compatilibilidade em que não conseguimos enviar pacotes em IPv4 ou IPv6 podemos sempre enviar em túneis **IPv4 sobre IPv6**, onde um pacote IPv4 é transportado com um cabeçalho externo IPv6, ou um **IPv6 sobre IPv4**, onde um pacote IPv6 é transportado com um cabeçalho externo IPv4.

IPv4 sobre IPv6
IPv6 sobre IPv4
↬ RFC 2473 (4in6)

Existe ainda um tipo, criado pela Cisco, chamado **GRE** (acrónimo inglês para *Generic Routing Encapsulation*) e que permite um variado conjunto de outras funções como especificar um largo conjunto de protocolos que estão adjacentes ao pacote, tal como possível autenticação, entre outros... O túnel GRE pode ser instalado sobre IPv4 ou IPv6.

GRE
↬ RFC 4213 (6in4)
↬ RFC 2890 (GRE)

Os túneis permitem depois criar conjuntos de ligações virtuais que, num seu todo, formam aquilo a que chamamos de **redes overlay**. Uma rede *overlay* é, no fundo, uma rede virtual sobre uma segunda rede virtual, passível de ser usada para garantir condições de qualidade de serviço ou segurança reforçada.

redes overlay

A esta rede *overlay*, quando se atinge um determinado nível de segurança e privacidade, damos o nome convencional de **VPN**, sigla inglesa para Rede Virtual Privada (*Virtual Private Network*).

VPN

Mecanismos de tradução IPv4/IPv6

Conhecendo o panorama de como é que funcionam os túneis e sabendo que é possível fazer a transição de IPv4 para IPv6 (e vice-versa), talvez já consigamos deslindar que procedimentos são necessários para fazer uma **tradução** entre ambos protocolos. Note-se que uma tradução implica que haja uma correspondência biunívoca entre ambos os endereços IPv4 e IPv6.

tradução

A forma mais "fácil" de implementar uma tradução é criar uma lista de correspondências no planeamento de alocação de endereços e depois configurar manualmente a tradução, através de ligações diretas e permanentes entre dois domínios IPv6 sobre um *backbone* IPv4. Isto seria feito com a ajuda de túneis IPv6 sobre IPv4. Uma segunda forma mais "fácil" seria fazer o mesmo mas com um túnel GRE IPv4.

Mas tornemos isto um pouco menos manual e antes mais automático. Pegando na mesma solução que explorámos até agora consideremos que criamos uma entidade a que chamamos de **tunnel broker**. Um *tunnel broker* é um equipamento instalado na rede que configura um determinado túnel entre dois pontos, a pedido de tradução por via de um *router*. Esta arquitetura até parece resolver-nos os problemas, mas possui uma grande fragilidade - o *broker* é uma máquina que, sendo única, em caso de falha, deixa de poder garantir traduções para nenhum cliente. Esta arquitetura, no entanto, existe implementada em normas como a **Teredo**, criada pela Microsoft, e normalizada no RFC 4380.

tunnel broker

Teredo
↬ RFC 4380

Entramos assim num conjunto de mecanismos mais automáticos para a tradução de endereços IPv4/IPv6 e cria-se um mecanismo, que se usa nos dias que correm, denominado de **automatic 6to4**.

automatic 6to4
↬ RFC 3056

O mecanismo dos túneis automáticos 6to4 são baseados em túneis convencionais IPv4 sobre IPv6, mas onde o prefixo global de encaminhamento (excluindo o TLA) é uma transposição do endereço IPv4 em IPv6. O prefixo, neste caso, terá de ser, obrigatoriamente 2002::/16, sendo que as redes finais, traduzidas para IPv6 serão de máscara 48, isto porque 16 bits do prefixo IPv6 com os 32 bits do endereço IPv4 dão os 48 bits de máscara.

Por exemplo, consideremos que temos o endereço 192.168.54.2 e queremos passar para IPv6. Com o 6to4 acabamos por ter uma tarefa fácil em mãos, sendo que ao prefixo 2002 apenas temos de juntar *a.b.c.d* em hexadecimal na forma 2002:*AABB*:*CCDD*::/48. Na Figura 3.9 podemos ver as traduções IPv4 para IPv6 e vice-versa.

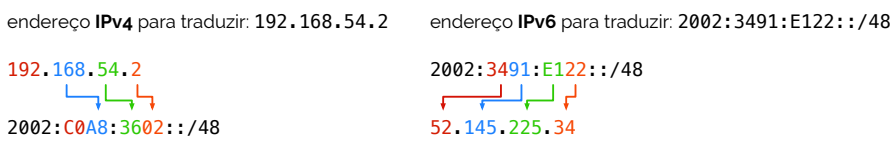


figura 3.9

4. Redes Sem Fios

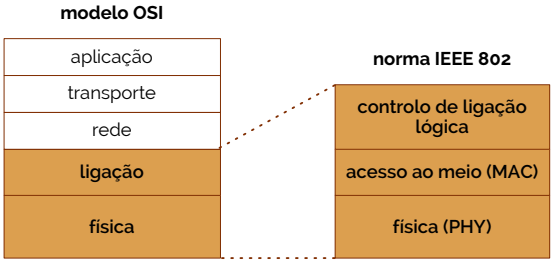
Cada vez mais, no nosso quotidiano, usamos **redes sem fios** para acedermos a meios conetectados a uma rede. Quer a rede seja a Internet ou outra, é cada vez mais um requisito haver interfaces de rede capazes de comunicarem usando o meio atmosférico como canal físico de transmissão e receção de dados.

redes sem fios

Recordando o modelo OSI, as redes sem fios, pelas normas da IEEE encontram-se particionadas de forma diferente, dados os seus requisitos e outras normas de funcionamento. Daí que ao longo deste documento, quando se referir qualquer aspeto ligado a redes sem fios, se estabeleça uma relação direta com as normas **IEEE 802**. As correspondências entre ambos os modelos podem ser vistas na Figura 4.1.

IEEE 802

figura 4.1



Introdução às redes de área local sem fios (WLAN)

As redes sem fios podem ser vistas como LANs convencionais, mas em termos técnicos acabam por ser diferentes porque têm requisitos de funcionamento completamente distintos. Por esta mesma razão denominamos estas LANs de **WLAN**s, nome abreviado de *Wireless Local Area Network*.

WLAN

Existem dois grandes tipos de WLANs: as de infraestrutura e as *ad-hoc*. Estas redes possuem um enorme leque de vantagens, mas também incluem um vasto conjunto de desvantagens, como a quantidade de soluções proprietárias que existem - dificultando o desenvolvimento de produtos com estas tecnologias -, as restrições do espetro eletromagnético e as baixas larguras de banda para a transmissão de dados, em comparação com as infraestruturas cabladas.

Não obstante, estas redes têm um grande propósito e uma popularidade enorme, e são constuídas por um pequeno conjunto de **componentes**, entre os quais as **estações** (que recebem os sinais e dados), os **pontos de acesso** (aos quais as estações se ligam e que fornecem um serviço ao cliente final, vulgarmente designadas pela abreviatura **AP**), os **Basic Service Set** (vulgarmente abreviado de BSS, sendo a rede por si que é propagada pela AP e que a estação usa, identificada por um nome - o **SSID**) e os **Extended Service Set**s (ESS), que são, no fundo, conjuntos de BSSs interconectados por APs.

componentes, estações
pontos de acesso
AP
Basic Service Set
SSID
Extended Service Set

No entanto nem todas as redes WLAN possuem estes componentes todos. É o caso das redes **ad-hoc**, caraterizadas por se formarem num curto intervalo de tempo, com um BSS independente e proveniente de uma estação que fornece o serviço - não de um AP.

ad-hoc

Todos estes tipos de WLAN possuem o conjunto de operações e serviços normalizados com o identificador **IEEE 802.11**, que define os parâmetros de conexão e uso de redes sem fios. Nestes serviços incluem-se as capacidades das estações efetuarem **autenticação** (fazerem *login*), **desautenticação** (fazerem *logout*), possuírem critérios de **privacidade** e **entrega de dados**. Como estamos perante um meio igualmente físico, de transmissão de

‹ IEEE 802.11
IEEE 802.11
autenticação, desautenticação, privacidade, entrega de dados

dados, precisamos também de alguns serviços de distribuição, como a **associação** (que se encarrega de efetuar uma ligação lógica entre a estação e um AP - não estando o AP a receber dados de nenhuma estação antes da associação), a **reassociação** (que não passa de uma associação repetida mais que uma vez, possivelmente para alertar o AP de que a estação se encontra a mover para outro BSS) e, claro está, a **dissociação** (onde se disconecta manualmente a estação da AP).

associação

reassociação

dissociação

Operação de ligação a um Basic Service Set (BSS)

Da mesma forma que numa rede com fios nós precisamos de efetuar algumas trocas de informação para obter uma conexão válida, numa rede sem fios precisamos de passar por um conjunto de operações para nos ligarmos com sucesso a um BSS.

Identificando-nos como estação e tirando partido do conjunto de operações normalizadas nos serviços IEEE 802.11, ao procurar um ponto de acesso através de processos de **scanning** (e/ou *probing*), dever-se-á dar início a uma troca de pacotes mais pequenos, aos quais damos o nome de **frames**. Nestes pacotes especiais a estação começa por tentar obter informação acerca de um AP e poderá fazê-lo de duas formas distintas: numa forma **passiva**, a estação apanha um **Beacon frame** (que são *frames* que os APs constantemente lançam para o meio físico, com as suas informações); ou, numa forma **ativa**, a estação tenta procurar um AP específico, enviando um **Probe Request** para o meio, onde se pede informação acerca de outro equipamento de rede. Em reposta ao *probe request* surge um **Probe Response** com a informação pedida.

scanning

frames

passiva, Beacon frame

ativa

Probe Request

Probe Response

Uma vez que a estação já possui a informação do AP ao qual se pretende ligar, então agora deverá passar os critérios de **autenticação**. Existem **redes abertas**, isto é, que só precisam de dois passos livres para efetuar a autenticação, não possuindo grande segurança - os pacotes enviados são meramente *frames* de *acknowledgement*. Por outro lado, grande parte das redes possuem mecanismos de autenticação que precisam de ser respondidos para que nos possamos conectar à rede em causa. Estes processos são feitos através de autenticação por **chave partilhada**. Iremos abordar este tópico com mais cuidado na disciplina de Segurança (a4s1), mas de qualquer das formas o que acontece é que através de um canal seguro que se estabelece, a interface de rede sem fios da estação envia um **Authentication Request**, sobre o qual recebe uma resposta vinda do AP referindo um **desafio**. Este mesmo desafio deverá ser encriptado e enviado pela própria estação num novo *authentication frame*. Neste ponto o AP deverá verificar se o desafio foi bem correspondido, desencriptando-o com a sua própria chave. No fim deste processo devemos obter o estado da autenticação por parte da interface de rede da estação.

autenticação, redes abertas

chave partilhada

Authentication Request

desafio

Findado o processo de autenticação terminamos a nossa tentativa de ligação à rede através de uma **associação**. Esta associação consiste na troca de informações entre a estação e o AP sobre as suas capacidades de conexão e *roaming*. Assim, primeiramente a estação envia um **Association Request** para o AP, permitindo que este aloque recursos e sincronize com a estação. Em resposta, recebe um **Association Response**, onde vem a aceitação ou rejeição das capacidades explícitas nas informações enviadas, por parte do AP. Em caso de aceitação, neste pacote virão também as informações sobre um identificador de associação e a taxa de transferência de dados que foi alocada para a estação, por parte do AP.

associação

Association Request

Association Response

Tendo terminado este processo, então podemos dizer que estamos em condições plenas para transmitir e receber dados.

Ao longo dos anos, desde o ano de 1999, que várias atualizações à norma IEEE 802.11 têm sido feitas, cada uma com melhor largura de banda, taxas de transferência de dados, tal como tecnologias suportadas e técnicas de modulação. Hoje em dia (ano de 2017) ainda temos muitas máquinas com suporte apenas de 802.11b, 802.11g ou 802.11n, mas as máquinas que se vendem há mais de 2 anos já vêm com interfaces de redes capazes de se associarem a redes sem-fios com a norma IEEE 802.11ac, que trabalha na banda dos 5GHz e possui uma taxa de transferência estimada de 6.93 Gb por segundo. [7]

Meio de acesso físico em redes sem fios e problemas associados

Na disciplina de Fundamentos de Redes (a3s1) já investigámos alguns problemas que poderão surgir do uso do meio atmosférico para a transmissão de dados. Em particular, referimos inclusive dois dos problemas mais conhecidos no meio de redes sem fios: o problema dos nós escondidos e o problema dos nós expostos.

CSMA/CD

Contrariamente ao meio Ethernet, onde se usa **CSMA/CD** (*Carrier Sense Multiple Access with Collision Detection*), no meio *Wireless* não nos é permitido efetuar o mesmo processo porque as colisões ocorrem sempre nos recetores, o que não conseguimos prever ao verificar se o meio está livre em redes sem fios. Uma solução que estudámos foi o **MACA** (*Multiple Access with Collision Avoidance*). Neste mecanismo dois pacotes são usados para sinalizar quando se pretende enviar um pacote para o meio e se se pretende aceitar a receção de um pacote por vias de outro terminal - pacotes **RTS** (*request to send*) e **CTS** (*clear to send*), respetivamente. Estes pacotes contêm informações sobre o endereço de quem pretende enviar um pacote, o endereço para onde pretende enviar o pacote e uma quantidade representativa do tamanho do pacote a ser enviado/recebido.

MACA

RTS

CTS

acknowledgement

No entanto, como já sabemos, o meio físico usado nas redes sem fios é muito propício a erros, pelo que o transporte não é, de todo, fiável. Então como é que podemos garantir que o envio de um pacote foi feito sem perdas? Ora, através do envio de pacotes de **acknowledgement**, quando um terminal recebe um pacote de outro terminal, responde-lhe com um ACK. Isto permite que, quando um emissor não recebe um ACK de um recetor para quem enviou um pacote, o volte a enviar. Enquanto isto acontece não são enviados mais RTS (ou respondidos CTS) por parte da máquina emissora. [3]

Segurança em WLANs

Numa rede tão propícia a erros e tão aberta a outros equipamentos se colocarem no centro de várias comunicações é importante que existam vários protocolos de segurança para garantirem menos entropia possível. Esta segurança passa muito pelos processos de autenticação inciais.

cifra

Anteriormente referimos o processo de **cifra**, em que se transforma dados numa forma não-legível a quem não possui uma devida chave para decifrar mensagens. Este processo fornece caraterísticas de confidencialidade e geralmente consiste num algoritmo associado a um conjunto de parâmetros, mas que não deve depender deste (ou da chave de cifra/decifra).

cifra simétrica

cifra assimétrica

par chave pública/privada

Existe duas técnicas de cifra: a simétrica e a assimétrica. Numa técnica de **cifra simétrica** temos uma única chave que permite a cifra e a decifra de dados por parte do emissor e do recetor. No segundo caso, na **cifra assimétrica**, temos um par de chaves, a que chamamos um **par chave pública/privada**. Aqui o emissor e o recetor partilham uma chave secreta, sendo que a mensagem é cifrada com a chave pública no emissor e decifrada com a chave privada no recetor[10].

Tendo este método é importante repararmos que no meio frágil das redes sem fios quaisquer mecanismos que existam para garantir a segurança deverão ser ligeiros, identificando os utilizadores no seu acesso à rede, não sendo possível trocar de identidade e não podendo causar grandes esforços computacionais para o cálculo de mensagens. Assim sendo, na norma IEEE 802.11 foram incluídas, ao longo dos tempos, questões de segurança que estão em constante desenvolvimento.

gateways

Anteriormente já falámos das redes abertas, mas é possível incrementar a segurança destas através de **gateways**. Na verdade, atualmente (ano de 2017), grande parte das redes abertas em meio público costumam possuir segurança a este nível, levando os seus clientes para uma **página web** quando se tentam associar à rede. Note-se, no entanto, que

página web

10 Poderá acontecer também ser criada uma terceira chave com a combinação das duas e só essa permitir a decifra da mensagem.

este método é tudo menos infalível: a página *web* alvo que nos é apresentada para preenchermos credenciais não tem forma de nos garantir que de facto se trata de algo oficial e não refeito por terceiros à busca de credenciais. Outro método usado com *gateways* é através de VPNs, onde o cliente deverá ligar-se a uma VPN de forma a poder conectar-se a uma rede específica do seu exterior.

Uma das primeiras preocupações de segurança em WLANs levou as equipas a integrarem nomes nas próprias redes. Esses nomes, vulgarmente denominados de **SSID** (*Service Set ID*) identificam então o BSS, emitido nos *frames beacon*. Com este mecanismo as redes poderão aceitar ligações com base no seu nome. No entanto este processo também não é infalível, uma vez que torna-se possível, replicar um SSID num equipamento e fazer com que equipamentos habitualmente associados com tal SSID se associem a mim desta vez. Conclui-se assim que este método é uma mais-valia na organização de uma rede, mas não na segurança desta.

SSID

A segurança nas WLANs começa assim a preocupar as equipas de investigação e assim surge, de uma forma algo apressada, um protocolo que resolveria (somente inicialmente) o problema de autenticação: o **protocolo WEP** (*Wired Equivalent Privacy*). Neste processo, enquanto se usa um esquema de chave partilhada, aplica-se segurança ao nível da camada 2 (pelo que se tornava computacionalmente eficiente e auto-sincronizado). Aqui a estação só teria mesmo de saber a chave para aceder o AP, sendo que com capturas de pacotes este processo poderá ser quebrado em segundos, uma vez que tudo passa em aberto - não há cifras dos cabeçalhos dos pacotes.

protocolo WEP

Como melhorias do protocolo WEP que se mostrou muito frágil, surgiu um novo protocolo - o **protocolo WPA** (*Wi-Fi Protected Access*) - que foi feito dentro da norma IEEE 802.11i. Embora apenas necesitasse de uma atualização do *firmware* dos equipamentos de rede já instalados, era usado tanto nas estações, como nos AP, havendo uma palavra-passe constante e partilhada, mas com cheves diferentes criadas por sessão.

protocolo WPA

< IEEE 802.11i

O grande avanço do WPA perante o WEP estava na colaboração de vários componentes, entre os quais a autenticação garantida por **802.1X** (protocolo de transporte que usa **EAP** - *Extensible Authentication Protocol*, um *wrapper* para tráfego específico de autenticação, que assegura que este não passe pelo AP - em conjugação com **PSK** - *Pre-Shared Key*) e uma cifra garantida por **TKIP**. O TKIP, sigla inglesa para *Temporal Key Integrity Protocol*, é uma solução interna para melhores resultados de proteção, que associa o meio de comunicação e usa chaves separadas para melhor integração.

802.1X
EAP
PSK
TKIP

Algum tempo depois surgira a versão 2 para o protocolo WPA (**WPA2**), que já requiria adquirir novos equipamentos. No entanto, já suportaria AES (um protocolo de cifra avançado), mas não cifra *frames* de controlo e de manutenção, pelo que as dissociações e outros continuam não cifrados.

WPA2

Como intermediário entre as estações e os AP podemos ter soluções IEEE 802.1X, tal como já referimos. Estas soluções usam vários métodos de autenticação como o EAP com MD5 (inseguro) ou sobre TLS. Estas tecnologias costumam ser orientadas por vias de um segundo protocolo denominado **RADIUS**.

RADIUS

RADIUS é um acrónimo inglês de *Remote Authentication Dial-In User Service* e permitem a autenticação de utilizadores perante uma rede. No fundo é um intermediário neste processo de autenticação, deixando essa responsabilidade fora do âmbito do utilizador. O que acontece é que o utilizador deixa ao RADIUS a indicação que se pretende conectar à rede, respondendo-lhe depois com instruções sobre como entrar na rede.

Estas transações que ocorrem entre o servidor RADIUS e o cliente são feitas num canal seguro, usando UDP.

Os pacotes RADIUS possuem cinco campos entre os quais um código de 1 byte (responsável como identificar o tipo de pacote em causa entre *Access-Request* (1), *Access-Request* (2), *Access-Reject* (3), *Accounting-Request* (4), *Accouting-Response* (5) e *Access-Challenge* (11)), um identificador de 1 byte (que permite ao RADIUS fazer corresponder mensagens que recebe e envia - é um contador), um campo de tamanho do pacote (2 bytes), um autenticador (que nos pedidos é preenchido com um número aleatório e nas

respostas é preenchido com um *hash* MD5 do tuplo `(código,ID,tamanho,pedido-de-autenticação,atributos,segredo-partilhado)`) e outros atributos. Na Figura 4.2 podemos ver a estrutura do pacote RADIUS. [7]

figura 4.2

5. Dinâmica de Redes Complexas

Como podemos verificar, à medida que vamos avançando neste documento, as redes que estamos a estudar aumentam cada vez mais a sua **complexidade**. Anteriormente também referimos, a propósito das rotas estáticas, que haver conteúdo estático torna-se inibidor de escabilidade, isto porque material estático (exceto em casos pontuais em que é mesmo necessário) indica que é necessário a reconstrução manual de todos os passos caso a escala da rede aumente ou diminua.

complexidade

Para resolver estes dilemas o melhor que temos a fazer será mesmo tornar tudo o mais **dinâmico** possível. Para isso já investigámos alguns dos conceitos que nos permitem fazer essa mudança, como o DHCP, DNS ou até mesmo o NAT, mas agora revisitemos os mesmos e tentemos aplicá-los em ambientes de complexidade muito mais elevada.

dinâmico

DHCP e NAT em redes complexas

Embora de forma e intuitos diferentes, ambos DHCP e NAT possuem uma vertente igual - a atribuição de endereços. Por um lado, o DHCP atribui endereços dentro de uma gama fornecida pelo administrador de redes, tal como o NAT, mas o NAT limita-se a oferecer endereços a entidades privadas que pretendam sair do âmbito da sua rede para o exterior, enquanto que com o DHCP a atribuição feita depende exclusivamente da gama de endereços fornecida (a **pool** de endereços).

pool

Como já devemos saber de Fundamentos de Redes (a3s1), e referindo-nos à versão IPv4 do DHCP, este protocolo funciona tendo base em agentes de *relay* que correm processos **BootP**. Assim, numa rede complexa, estando um servidor DHCP normalmente alojado no datacenter ou zona DMZ, os *routers* que o separa das VLANs onde se pretende que se use o DHCP deverão ter o *relay* por BootP ativo, indicando o endereço IP-alvo do servidor DHCP para onde os pacotes BootP deverão ser guiados.

BootP

Em termos de **NAT** já sabemos que é responsável por criar, quando necessário (após pedido), uma correspondência entre um endereço privado e um endereço público, não havendo garantias, no entanto, que o mesmo endereço esteja sempre disponível (a não ser que seja feita uma correspondência de tradução estática). Com *pools* de endereços pequenos também vimos em Fundamentos de Redes (a3s1) que poderíamos usar um complemento deste mecanismo denominado de **NAT/PAT**, que junta a capacidade de formar traduções de conjuntos endereço-porto, permitindo assim, com um mesmo endereço, haver mais oportunidades de tradução.

NAT

RFC 2663

NAT/PAT

RFC 3022

Nas nossas redes empresariais o NAT será aplicado à saída do espetro privado, numa comunicação com o ISP. Veja-se a Figura 5.1.

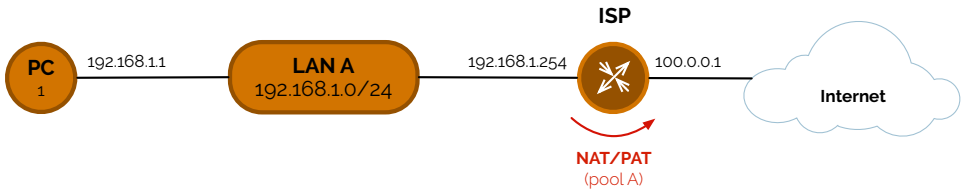
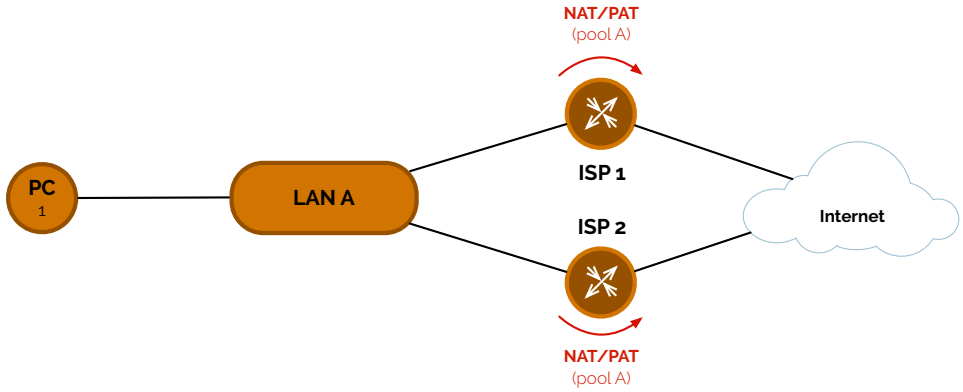


figura 5.1

Na Figura 5.1 podemos ver um exemplo em que se o PC 1 precisar de comunicar com a Internet, então quando passará no ISP irá receber um endereço da mesma rede que o 100.0.0.1, mas definido na *pool* A do processo NAT/PAT.

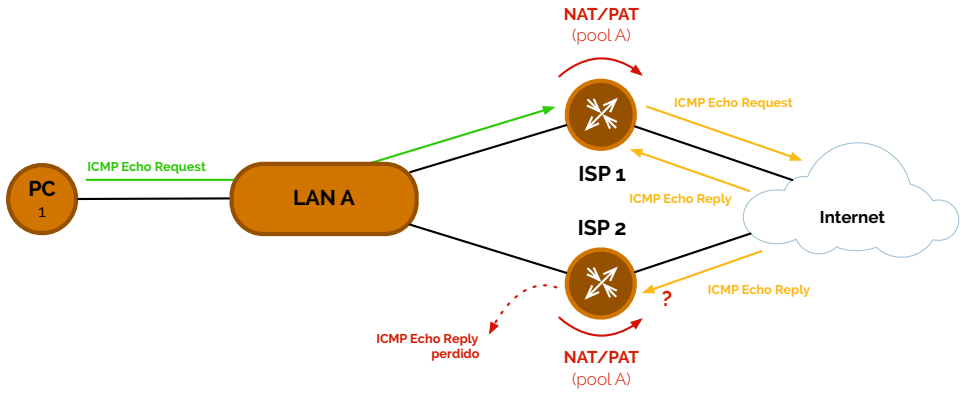
Mas a Figura 5.1 mostra um exemplo em que só temos um ISP. Quando tivermos ligação a mais que um ISP podemos aplicar NAT com *pools* diferentes em cada ISP, mas tentando usar a mesma *pool* nos vários ISPs será que é possível obter uma solução? Vejamos então a Figura 5.2 onde tentamos aplicar esta mesma lógica.

figura 5.2



Ao implementar uma rede semelhante à representada na Figura 5.2 incorremos num erro em que não há sincronia entre os dois ISP a respeito da tabela NAT. Porquê? Ora vejamos que, quando o PC 1 envia um pacote ICMP para a Internet passa num dos ISPs, recebendo um IP público - consideremos que passou pelo ISP 1, para efeitos de demonstração. A resposta da Internet ao *ping* enviado para si, dado que se publicita para ela a rede pública NAT pela interface ligada ao ISP 1 e ISP 2, poderá ser levado por balanceamento (ou na totalidade) para o ISP 2. Este, por sua vez, recebe um endereço e identifica-o como sendo da sua rede NAT, pelo que vai à sua tabela NAT procurar o registo de tradução e não o encontra - o registo foi feito no ISP 1, não no ISP 2 (Figura 5.3).

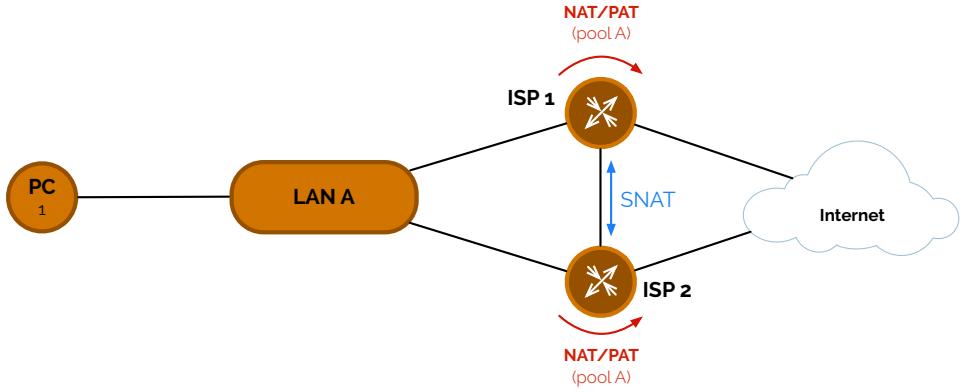
figura 5.3



Para **sincronizar** as duas tabelas de NAT dos ISPs podemos usar uma pequena adptação do mesmo mecanismo, num modo **stateful**. Usando o NAT *stateful* podemos então manter a mesma tabela entre os dois *routers*. Note-se que esta solução é proprietária da Cisco e tem a sincronização a ser feita sobre TCP. Veja-se a Figura 5.4.

sincronizar
stateful

figura 5.4



Introdução ao Domain Name System (DNS) e hierarquia de domínios

O NAT faz traduções de endereços privados para endereços públicos (e vice-versa, salvo casos em que um terminal privado não comunicou recentemente para o exterior). Quer do domínio privado, como do domínio público, por muito boa que a organização e planeamento da alocação de endereços seja, é sempre difícil lembrarmo-nos dos endereços IP dos equipamentos. E se lhes atribuíssemos **nomes**? Para lhe atribuirmos nomes, por outro lado, também precisaríamos de ter um sistema que nos efetuasse uma tradução de tais identidades para endereços IP e vice-versa. Esse sistema é denominado de **DNS**, sigla para *Domain Name System* (em português Sistema de Domínios de Nome).

nomes

DNS

De forma a este sistema poder organizar a sua informação em termos globais foi criada uma **hierarquia** de domínios de nome. Na Figura 5.5 podemos ver um conjunto de níveis de domínios entre os quais os **Top-Level Domains** (**TLD**) (genéricos (gTLD), de código de país (ccTLD) e novos como .xyz ou de marcas como .apple.

hierarquia

Top-Level Domains, TLD

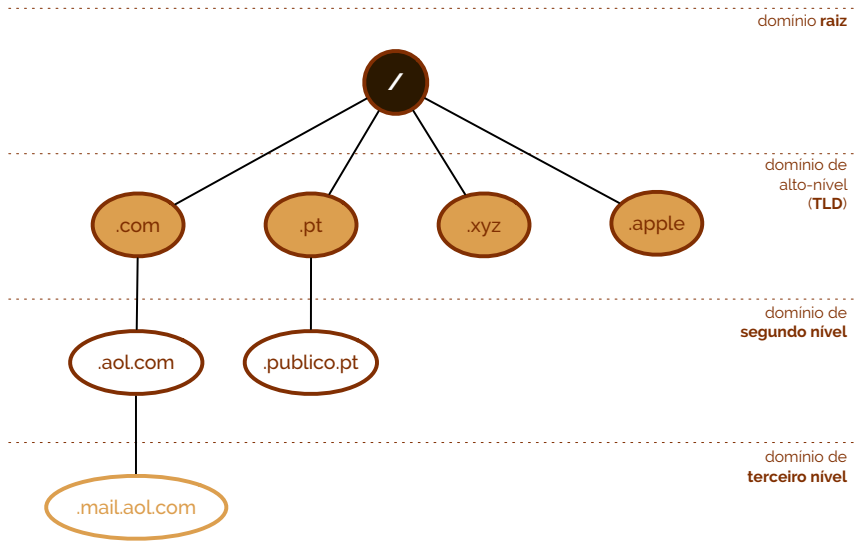


figura 5.5

Estas hierarquias têm a sua **raiz** em servidores raiz em todo o mundo. Estas máquinas ligam-se a outras um nível abaixo que estruturam os nomes de domínio **genérico** (como .com, .edu, .mil, .net, .org, .int, .aero, .biz, .coop, .info, .museum, .name, .pro, .cat, .jobs, .mobi, .travel, .tel e .asia), os nomes de domínio de **código de país** - segundo a norma ISO 3166 - com duas letras (como .pt, .us, .es, .fr, .it, ...) ou outros novos genéricos que vão surgindo, como .xyz, .apple, .site, .club, .hotel, .tourism, .fly, .university, ...

raiz

genérico

código de país

Gestão e pedido de nomes

A hierarquia de nomes é devidamente regulada por um conjunto de entidades que se responsabilizam especificamente para o efeito. A entidade mais importante neste conjunto é o **ICANN** - *Internet Corporation for Assigned Numbers and Names*. Como não se haveria de esperar diferente, é ICANN que regula o uso e atribuição dos nomes genéricos. Pelo contrário, está ao cargo dos overnos de cada país regulamentar o uso dos domínios que usam o(s) código(s) do seu país - embora existam países, como os Estados Federativos da Micronésia, que permitem a gestão do seu domínio .fm a empresas privadas, de lucro total. [8]

ICANN

Abaixo dos corpos que regulamentam os domínios, como o ICANN e as várias instituições em representação dos vários governos mundiais encontram-se os **registries**, que mantêm ficheiros com a lista de nomes para domínios específicos - ficheiros esses denominados de **ficheiros de zona**, como iremos ver mais à frente. Os registries, por sua vez, de-

registries

ficheiros de zona

legam aos **registrars** a gestão operacional e a venda dos domínios disponíveis através de **revendedores**.

registrars
revendendores

Os nomes de domínio têm, no entanto, um **ciclo de vida**. Quando é feito o registo de um domínio, este pode manter-se registado durante um período de 1 a 10 anos. Quando este prazo termina há que **renovar** o registo. Em caso de falha de renovação, inicia-se um processo de eliminação do nome da base de dados DNS, se bem que hoje em dia (ano de 2017) os registrars não largam de imediato o registo depois do **período de redenção**, sendo que iniciam uma revenda (usualmente através de um leilão) do domínio num mercado secundário. Veja-se a Figura 5.6.

ciclo de vida

renovar

período de redenção

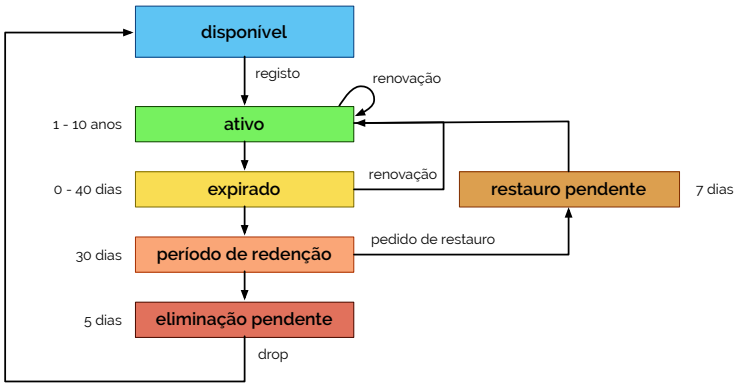


figura 5.6

Uma vez que os registos dos domínios estão feitos, neles têm de estar todas as informações, principalmente de contacto de índole geral, administrativo, técnico e de faturação. Estas informações podem ser obtidas pelo serviço **WHOIS**, que junta estas informações, tais como os seus servidores de nome, o estado do domínio e as datas de criação, expiração e de úlitma atualização. No Código 5.1 podemos ver um exemplo de informação obtida pelo serviço WHOIS para o nome `www.washingtonpost.com`, um jornal de Washington D.C., Estados Unidos.

WHOIS

código 5.1

```
Domain Name: washingtonpost.com

Registry Domain ID: 1707992_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.corporatedomains.com
Registrar URL: www.cscprotectsbrands.com

Updated Date: 2016-11-08T06:48:51Z
Creation Date: 1995-11-13T05:00:00Z
Registrar Registration Expiration Date: 2017-11-12T05:00:00Z

Registrar: CSC CORPORATE DOMAINS, INC.
Registrar IANA ID: 299
Registrar Abuse Contact Email: domainabuse@cscglobal.com
Registrar Abuse Contact Phone: +1.8887802723

Domain Status: serverTransferProhibited
Domain Status: serverDeleteProhibited
Domain Status: clientTransferProhibited
Domain Status: serverUpdateProhibited

Registry Registrant ID:
Registrant Name: Domain Administrator
Registrant Organization: WP Company LLC
Registrant Street: 1301 K Street, NW
Registrant City: Washington
Registrant State/Province: DC
Registrant Postal Code: 20071
Registrant Country: US
Registrant Phone: +1.2023347868
Registrant Phone Ext:
Registrant Fax: +1.2023345075
Registrant Fax Ext:
Registrant Email: admin.contact@digitalink.com

Registry Admin ID:
Admin Name: Domain Administrator
Admin Organization: WP Company LLC
Admin Street: 1301 K Street, NW
Admin City: Washington
```

```
Admin State/Province: DC
Admin Postal Code: 20071
Admin Country: US
Admin Phone: +1.2023347868
Admin Phone Ext:
Admin Fax: +1.2023345075
Admin Fax Ext:
Admin Email: admin.contact@digitalink.com

Registry Tech ID:
Tech Name: Administrative Technical Contact
Tech Organization: WP Company LLC
Tech Street: 1301 K Street, NW
Tech City: Washington
Tech State/Province: DC
Tech Postal Code: 20071
Tech Country: US
Tech Phone: +1.2023344530
Tech Phone Ext:
Tech Fax: +1.2023345075
Tech Fax Ext:
Tech Email: tech.contact@digitalink.com

Name Server: pdns3.ultradns.org
Name Server: ns-1840.awsdns-38.co.uk
Name Server: pdns115.ultradns.net
Name Server: pdns6.ultradns.co.uk
Name Server: ns-404.awsdns-50.com
Name Server: pdns5.ultradns.info
Name Server: pdns2.ultradns.net
Name Server: pdns4.ultradns.org
Name Server: ns-1027.awsdns-00.org
Name Server: ns-666.awsdns-19.net
Name Server: pdns1.ultradns.net

DNSSEC: unsigned

URL of the ICANN WHOIS Data Problem Reporting System: http://wdprs.internic.net/

>>> Last update of WHOIS database: 2016-11-08T06:48:51Z <<<
```

Note-se que é possível ocultar algumas das informações exibidas acima, mas isso carece de uma taxa e serviço extra.

Resolução de nomes

Nem sempre é útil, pelo menos para as instituições governamentais ou fornecedores de serviço Internet (ISP), tornar acessíveis todas as correspondências DNS no mundo. Para tal, em países como Portugal, em que há um conjunto de páginas Web que se encontram bloqueadas pelos ISP, por lei, aplica-se um **local name server**, isto é, um servidor que não tem de pertencer à hierarquia DNS, mas que faz parte do ISP e que fornece algumas correspondências pedidas de um modo diferente que faria um servidor DNS de níveis mais superiores. Quando o servidor DNS local não tiver a entrada nas suas tabelas, terá de consultar o primeiro acessível em termos hierárquicos.

local name server

Vejamos agora como é que os pedidos por correspondência de endereços passam na hierarquia. Existem dois modos de efetuar a resolução de nomes. Consideremos a Figura 5.7, onde o pedido é feito de modo **iterativo**.

iterativo

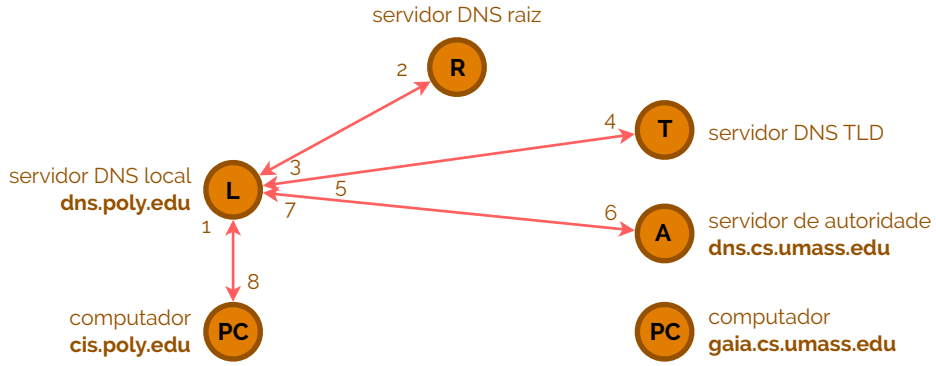


figura 5.7

Considerando a Figura 5.7, podemos verificar que os servidores são contactados a partir de um servidor DNS local. Basicamente, o que aqui acontece é que se está a perguntar, a partir do servidor local, a cada um dos outros, se conhece um determinado nome, ao que eles, caso a resposta seja negativa, dizem: "não sei quem tem esse nome, mas pergunta antes a este servidor". Esta forma de resolver o nome por DNS, apesar de funcionar, é bastante lenta e ocupa o servidor local, mantendo-o à espera de um pedido DNS. Mais, a distância que se encontra o servidor local do servidor de autoridade da figura, por exemplo, pode ser muito grande, o que irá manter o local muito tempo à espera ou mesmo sem resposta.

Uma forma de resolver os problemas das iterações é implementar um algoritmo **recursivo** (Figura 5.8). Implementando um algoritmo recursivo, na prática, o que terá de fazer é: conduzir o pedido até à raiz e descer pelo menor caminho. Embora use recursos de vários servidores, o tempo despendido para a resolução do nome DNS é muito curto.

recursivo

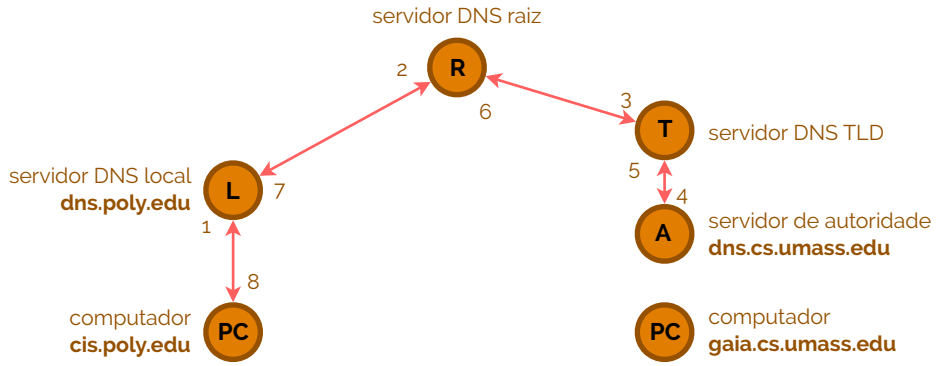


figura 5.8

Em suma, a resolução de forma recursiva é muito mais eficiente que a forma iterativa, pois minimiza o tempo entre os pedidos e as respostas, no entanto requer mais poder de processamento por parte dos servidores DNS, pelo que cada servidor terá de ser capaz de processar mais pedidos em simultâneo. Já a resolução de nomes por iterações é menos eficiente porque o tempo entre o pedido e a resposta pode ser muito largo (que o é, em média), no entanto, minimiza-se o poder de processamento requerido nos servidores DNS, pelo que cada servidor responde imediatamente a cada pedido.

Se uma determinada máquina já efetuou um pedido DNS este ficará preservado nela durante algum tempo em cache (depois desaparece). Assim, se de seguida, com a entrada ainda em cache, precisarmos de visitar uma página onde estivemos há pouco, não haverá pesquisa em servidores DNS pela resolução do nome. Antes, será revisitada, a tabela DNS da cache, presente localmente, na máquina. [3]

Tipos de servidores de nomes DNS

Dentro do DNS existe um vasto conjunto de servidores de diferentes tipos. Um **servidor master** (ou primário) é tal que define um ou mais ficheiros de zona para os quais este DNS é *authoritative* (em português "oficial"). Estes ficheiros de zona estão definidos localmente.

servidor master

Abaixo de um *master*, tal como em qualquer outra implementação de uma arquitetura *master-slave*, está um **slave** (ou secundário), que obtém os seus ficheiros de zona através de uma operação de transferência, tipicamente proveniente de um servidor primário e de forma automática. Este servidor responde como sendo o oficial para as zonas onde está definido como sendo o *slave* e para as quais tem uma configuração de zona válida para a altura - isto acontece porque não é possível identificar a origem das consultas DNS, entre sendo de um servidor *master* ou *slave*.

slave

Um outro tipo de servidor de nomes é uma **cache**, tal como já referimos acima. Não sendo uma fonte oficial, permite que registos procurados recentemente fiquem armazenados num local mais próximo do cliente durante algum tempo. De facto a cache não

cache

é uma fonte oficial de dados, mas quando pede dados diretamente ao *master* e tem de responder por ele diz-se oficial.

Diferente de uma cache é um servidor de nomes que só redireciona os pedidos e consultas - chamamos-lhe de **proxy**. Este servidor também possui capacidade de caching, mas torna-se melhor uma vez que é bastante útil quando o acesso aos dados é lento ou demasiado caro, pelo que reduz o acesso externo, aumentando o tempo de resposta e descartando tráfego desnecssário. Note-se que, neste caso temos um equipamento que faz caching, pelo que os servidores DNS devem apenas efetuar uma função - ou é um servidor oficial, ou é um agregador (cache), mas nunca as duas coisas no mesmo servidor (embora possam existir em empresas com dimensões muito grandes).

proxy

Os servidores que possuem funções apenas oficiais têm duas caraterísticas muito próprias - entregam respostas oficiais, isto é, zonas definidas num *master* ou num *slave* - e não fazem cache. Por esta mesma razão estes servidores costumam ser usados em um de dois cenários: como servidores DNS de elevado desempenho ou como servidor público (ou externo) numa configuração stealth, fornecendo um maior perímetro de segurança.

Uma configuração **stealth** (ou de separação) DNS é tal que existe uma fronteira entre um "servidor interno" ou mais e um (ou mais) servidores externos/públicos. Ao servidor interno damos o nome de **servidor stealth**, os quais fornecem um conjunto de serviços a utilizadores internos, entre os quais caching e consultas recursivas, sendo configurado como servidor *master*. Por outro lado, os servidores externos podem fornecer serviços mais limitados e, tipicamente, são configurados como servidores somente oficiais.

stealth

servidor stealth

Num servidor de separação aparecerão informações tanto de terminais privados, como de públicos, enquanto que no servidor público apenas aparecem os terminais públicos.

Conceito de zona: atualizações, tipos e configuração (BIND)

Uma **zona** é um conjunto de configurações definidas para localizar um determinado nome num equipamento de rede (inclusive na própria máquina). Esta é definida através de declarações de zona, que descrevem o tipo, um ponteiro para o ficheiro de zona e alguns atributos especiais (opcionais) ou através de um simples ficheiro de zona, onde se preservam os registos de recursos DNS para todos os nomes de domínio associados com a zona.

zona

Estas configurações, como referido anteriormente, são atualizadas nos *slaves* através de pedidos feitos por este numa lógica AXFR, automaticamente através de mensagens informativas após alteração de registos, numa lógica IXFR (incremental) ou, pelo RFC 1912, após um pedido feito depois do alerta REFRESH despoletado automaticamente pelo *master* (em intervalos de até 12 horas).

↬ **RFC 1912**

As próprias zonas têm um conjunto de **tipos** de definição. Estas podem ser **zonas master** se o servidor lê os dados da zona diretamente do armazenamento local e fornece informações oficiais para a mesma. Se for réplica desta, então é uma **zona slave** e obtém os dados através de atualizações ao *master*.

zonas master

zona slave

Existe também o tipo de **zona hint** (que é usado quando os servidores DNS arrancam, para obter a lista de servidores-raiz), o tipo de **zona forward** (um tipo de zona que cria uma forma simples de configurar o reencaminhamento por domínio, ou por zona), o tipo de **zona stub** (zona semelhante à *slave*, à exceção do facto de replicar um pequeno conjunto de registos na zona, mas antes todos os registos presentes no *master*) e a **zona de delegação** (que fornece referências a servidores DNS mais abaixo na hierarquia).

zona hint

zona forward

zona stub

zona de delegação

Um exemplo de configuração de zonas, pode ser visto no Código 5.2, onde usamos o mecnismo **BIND** para as declarar.

BIND

código 5.2

```
zone "domain.com" {
  type master;
  file "zones/domain.com";
};
```

```
zone "200.136.193.in-addr.arpa" {
  type master;
  file "zones/193.136.200";
};

zone "example.com" in {
  type slave;
  file "slave.example.com";
  masters {192.168.2.7; 10.2.3.15 port 1127; 2001:db8:0:1::15;};
};
```

Os **ficheiros de zona**, assim, contêm registos de recursos que descrevem um domínio ou um sub-domínio, num formato definido pela norma RFC 1035. Este contéudo é tal que possui os dados que indicam o topo da zona e algumas das suas propriedades, dados oficiais para todos os nós e/ou terminais dentro da zona, dados que descrevem informação global a respeito da zona e, no caso de sub-domínios delegados, os servidores de nomes responsáveis por este.

ficheiros de zona
RFC 1035

Tipo dos registos dos servidores de nomes

Dada a quantidade de informação que está guardada nos ficheiros de zona, é necessário que haja algum tipo de convenção sobre de que forma é que a podemos caraterizar. Para isso foram criados vários tipos de registos, todos especificados no RFC 1035:

- **registo SOA** (*Start Of Authority*) - define o nome da zona, um contacto de correio eletrónico e vários tempos e valores de atualização aplicáveis à zona em questão;
- **registo A** - um registo de endereço IPv4 para um terminal;
- **registo AAAA** - um registo de endereço IPv6 para um terminal;
- **registo NS** (*Name Server*) - define o(s) servidor(es) de nomes oficial(is) para o domínio definido pelo registo SOA;
- **registo MX** (*Mail Exchanger*) - um valor de preferência e um nome de terminal para um serviço *exchanger* ou servidor de mensagens de correio eletrónico;
- **registo CNAME** (*Canonical Name*) - um *alias* para um terminal;
- **registo PTR** - um registo de endereço IP para um terminal (quer versão 4 ou 6), para uso em mapas invertidos (*reverse maps*);
- **registo TXT** - um registo de texto associado com um nome.

Resolução de nomes com reverse DNS

Até agora temos usado os nomes para atingir um endereço IP. Então e o contrário, será que é possível? Ora, se existe uma correspondência biunívoca entre endereços e nomes, então teoricamente é possível indicar um endereço IP e atingir um nome.

Com o **reverse DNS** não só é teoricamente possível, como também o é na prática. O que acontece é que há um TLD chamado .arpa que possui, abaixo deste, duas árvores de endereços - um para endereços IPv4 (o .ip-addr) e outro para endereços IPv6 (o .ip6). Estas árvores têm, nas suas folhas, os nomes dos IPs constituídos pelo caminho inverso em altura da árvore. Por outras palavras, e de forma mais prática, basta pedir o nome para o endereço `<ipv4-invertido>.in-addr.arpa` no caso de IPv4 e o nome para o endereço `<ipv6-invertido>.ip6.arpa` no caso de IPv6.

reverse DNS

Segurança em DNS (DNSSEC)

O mecanismo de DNS, em termos de segurança, tem muitos pontos onde pode ser atacado - desde a dados corrompidos nos ficheiros de zona, a atualizações de dados falsas a vdata modificada nos *slaves* (que são oficiais), ... muitos são os pontos de possível ataque.

Alguns destes pontos podem ser resolvidos se se ativar uma extensão ao DNS denominada de **DNSSEC** (*DNS Security Extensions*). Estas extensões permitem, assim,

DNSSEC

fornecer parâmetros de autenticação para o devido uso do DNS, através de pares de chaves privadas/públicas. Contudo, mais uma vez, como não é encriptado nenhum dado, protegido nenhum servidor de ataques de DDoS ou protegido nenhum utilizador de ataques de *phishing* só alguns dos pontos detalhados em cima é que estão cobertos pelo DNSSEC.

O DNSSEC possui quatro novos tipos de registo. Um destes registos é o **DNSKEY** que contém a chave pública usada nas assinaturas das operações com zonas. Este registo pode tanto definir uma **ZSK** (*Zone Signing Key*, usada para assinar dados numa zona), como uma **KSK** (*Key Signing Key*, usada para assinar ZSK e assim criar um ponto de entrada seguro (SEP - *Secure Entry Point*)).

DNSKEY

ZSK

KSK

Na Figura 5.9 podemos ver a estrutura deste pacote que consiste num campo de *flags*, num campo de protocolo, num campo de algoritmo e no campo pertencente à chave pública.

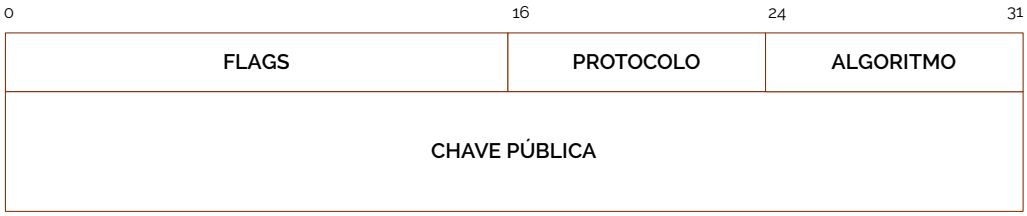


figura 5.9

O campo `FLAGS` (de 2 bytes) tem dois bits em particular que nos podem ser relevantes: o sétimo bit, sendo a *flag* de chave de zona, se tive valor '1', então significa que o registo possui uma chave de zona e que pode ser usada para verificar registos do tipo RRSIG; o décimo quinto bit corresponde à *flag* de SEP (os valores típicos para este campo são 256 ou 257, no primeiro caso tendo o bit 7 a '1' e o bit 15 a '0', e no segundo caso tendo o bit 7 a '1' e o bit 15 a '1' também). O campo `PROTOCOLO` deve ter valor 3 e o registo DNSKEY deverá ser tratado como inválido durante a verificação da assinatura, caso contrário. Já o campo `ALGORITMO` deverá ter uma desginação do algoritmo criptográfico em uso, entre Diffie-Hellman (valor 2), DSA/SHA-1 (valor 3), curva elítica (valor 4) ou RSA/SHA-1 (valor 5).

⊙ **Whitfield Diffie**

⊙ **Martin Hellman**

Outro registo importante é o **DS**, que é um tipo particular de DNSKEY, usado no processo de autenticação do pacote anterior. Este pacote relaciona-se com o anterior porque mantém uma etiqueta da chave pública, o seu algoritmo e uma versão já sumariada (*hashed*) da chave pública (*digest*). Sendo usado para criar uma **cadeia de confiança**, quem usar este registo irá comparar o seu *hash* com o *hash* do registo DNSKEY de onde este foi criado. Na Figura 5.10 podemos ver a estrutura de tal pacote, podendo reparar que é deveras semelhante à do DNSKEY.

DS

cadeia de confiança

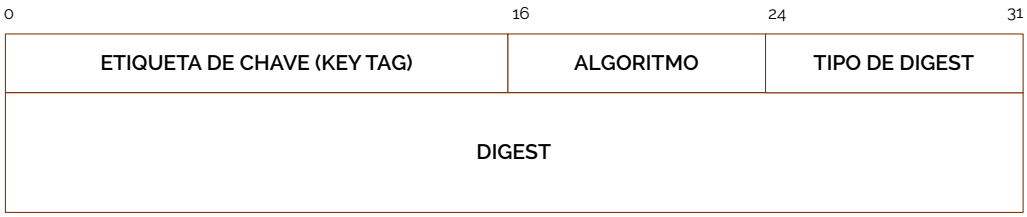


figura 5.10

Um registo de que já falámos foi o **RRSIG** que contém uma assinatura para um registo com um nome particular, classe e tipo, especificando o intervalo de validade para a assinatura. Para isso usa o algoritmo, o nome do assinante e a etiqueta de chave para identificar o registo DNSKEY que contém a chave pública, para que possa ser verificada a assinatura. A estrutura deste pacote está representada na Figura 5.11.

RRSIG

Finalmente, o último registo que foi acrescentado é o **NSEC**, que lista o nome seguinte (na ordem canónica da zona) que contém dados oficiais ou um ponto de delegação, juntamente com um conjunto de tipos de registos que o próprio tem no pacote

NSEC

NSEC. O conjunto completo de registos NSEC numa zona indica que registos oficiais existem numa zona, criando uma cadeia de proprietários de nomes oficiais na zona.

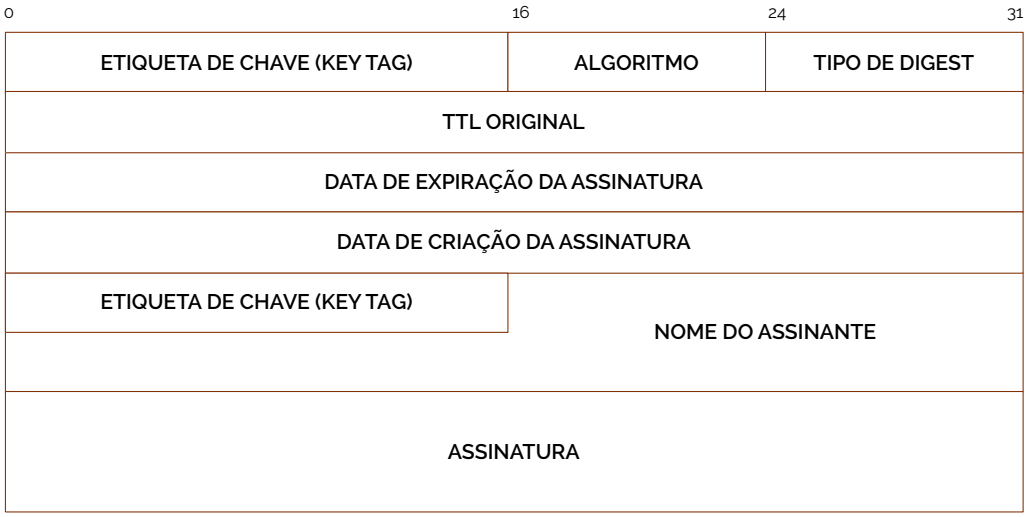


figura 5.11

Secret Key Transaction Authentication para DNS (TSIG)

Como vimos nem todos os problemas de segurança do DNS são resolvidos com o DNSSEC. Para tentar colmatar os outros problemas que ficaram por resolver, no RFC 2845 de maio de 2000 tentou-se criar uma forma de assegurar as comunicações que são feitas entre os servidores DNS.

RFC 2845

Isto consegue-se através de uma **chave TSIG**, que consiste numa *string* secreta e num algoritmo de *hashing*. Tendo a mesma chave em dois servidores DNS diferentes, as comunicações podem ser seguras, fazendo com que ambos os servidores confiem um no outro.

chave TSIG

Este mecanismo pode ser usado para autenticar pedidos provenientes de um cliente verificado ou respostas provenientes de um servidor de nomes recursivo. [9]

6. Tópicos de Segurança em Redes

Uma das maiores preocupações que existem hoje em dia no meio da computação e telecomunicações é, sem dúvida alguma, a **segurança**. Cada vez mais ataques se dão e, ao mesmo tempo, estamos mais dependentes do mesmo meio onde os primeiros acontecem.

segurança

Referimo-nos às comunicações entre duas ou mais máquinas ligadas em rede. Anteriormente já referimos alguns problemas e analisámos correspondentes propostas de solução - mais especificamente em termos de redes sem fios e de mecanismos de tradução de endereços IP em nomes. Mas a segurança em redes é muito mais do que isso, e neste capítulo iremos abordar algumas propostas de solução a problemas onde nem tudo corre como se pensou outrora que fosse correr.

Canais de comunicação seguros com IPSec

Uma forma de canalizarmos dados de forma segura em canais de comunicação entre duas ou mais máquinas é usando um *framework* de segurança de seu nome **IPSec**. Esta abreviatura, proveniente de *IP Secure*, corre sobre a camada de rede do modelo OSI e conjuga uma de três opções (um cabeçalho extra de autenticação, encriptação de dados - mas não dos cabeçalhos - ou um cabeçalho extra de autenticação e encriptação de dados), num de dois modos possíveis: sobre um túnel ou sob a forma de transporte.

IPSec

O **modo túnel**, de execução do IPSec, faz com que os terminais por onde o canal passa não se apercebam que está a ser usada esta *framework* para proteger o seu tráfego, fornecendo assim uma proteção totalmente transparente sobre redes não fiáveis.

modo túnel

Por outro lado, o IPSec pode correr em **modo de transporte**, isto é, cada terminal-extremo tem consciência de que existe o encapsulamento de dados sobre IPSec, havendo a necessidade de implementar o IPSec nos terminais.

modo de transporte

Na Figura 6.1 podemos ver as diferenças topológicas entre ambos os modos.

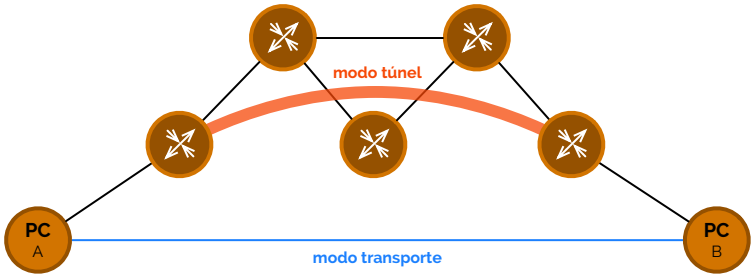


figura 6.1

Em termos de pacotes vejamos agora como são implementados os cabeçalho de autenticação e a cifra de dados. Começemos pelo **cabeçalho de autenticação**, AH, que no modo de túnel é colocado de forma análoga aos túneis que já abordámos anteriormente, como podemos ver na Figura 6.2.

cabeçalho de autenticação

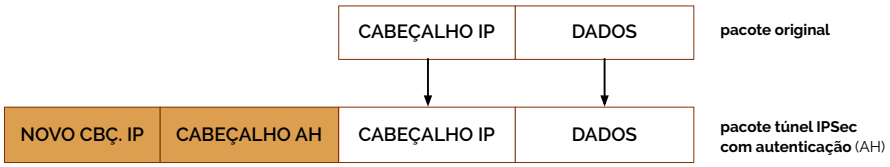


figura 6.2

Se quisermos implementar um novo cabeçalho de autenticação em modo transporte, então a única coisa que precisamos de fazer é mesmo só acrescentar esse cabeçalho antes do próprio cabeçalho IP, como podemos ver na Figura 6.3.

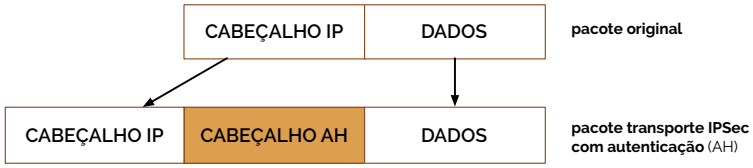


figura 6.3

A implementação da **cifra dos dados** (ESP) do protocolo IP é feito através, num modo de túnel, da inserção do protocolo IP total (neste caso o cabeçalho original também é cifrado) entre um cabeçalho ESP e um *trailer* ESP. Esta implementação é exibida na Figura 6.4.

cifra dos dados

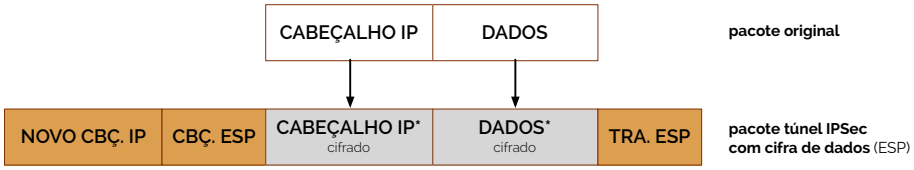


figura 6.4

O mesmo procedimento, mas agora em transporte é visível na Figura 6.5.

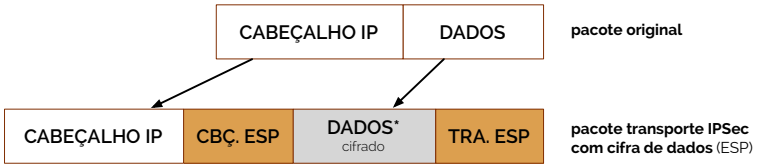


figura 6.5

Com a aplicação da lógica do IPSec surgiu a necessidade de dar um nome às relações seguras, isto é, aos contratos de políticas aplicadas entre dois parceiros (ou terminais). Criam-se assim as **associações de segurança** (vulgarmente abreviadas de SA) que descrevem como é que os *peers* irão usar os serviços de segurança do IPSec, de forma a prezar pela proteção do tráfego na rede. Para tal os SAs contêm a seguinte informação: algoritmos de autenticação e cifra, tamanhos de chaves e outros parâmetros de encriptação como tempo de vida de uma chave, entre outros...; chaves de início de sessão para autenticação (ou HMACs) e cifra, que poderão ser inseridos de forma manual ou automaticamente negociados; uma especificação do tráfego de rede onde o SA deverá ser aplicado; e a especificação do tipo e modo de aplicação do IPSec.

associações de segurança

Estabelecimento de associações de segurança e chaves criptográficas

De forma a podermos estabelecer os parâmetros de segurança vistos anteriormente é importante ter entidades que se responsabilizem por garantir que todos os critérios são devidamente correspondidos para o bom funcionamento do sistema.

Um exemplo de entidade que garante o bom funcionamento destas práticas é o **ISAKMP** (*Internet Security Association and Key Management Protocol*), mas como este também existem o *Oakley key Determination Protocol* (que usa somente o algoritmo de Diffie-Hellman), o *SKEME* (que é um mero protocolo de troca de chaves) e o IKE (que usa parte do Oakley e do SKEME, em conjugação com o ISAKMP). Neste documento iremos retratar os casos de aplicação do IPSec coordenados com IKE/ISAKMP.

ISAKMP

A combinação do IPSec com IKE/ISAKMP melhora-o dando-lhe funcionalidades extra e flexibilidade, fornecendo autenticação a *peers* IPSec, negociando as chaves IPSec en associações de segurança.

Haver um túnel IKE nesta relação tem um forte impacto no que toca à proteção das negociações dos SAs, pelo que as transferências de dados passam a estar constantemente protegidas por este mecanismo.

Estas relações inter-protocolares permitem que se elimine a necessidade de especificar manualmente os parâmetros de segurança IPSec em ambos os parceiros, permitindo que as chaves de cifra sejam alteradas ao longo do tempo de sessão durante o funcionamento do IPSec e que haja autenticação dinâmica de *peers*. Estes procedimentos também permitem que se torne os nossos sistemas escaláveis, através do uso de autoridades de **certificados** (CA), e pelo facto dos administradores poderem especificar um tempo de vida para as SAs do IPSec.

certificados

O ISAKMP, em particular, funciona em duas fases. Numa primeira fase há um acordo que é celebrado entre ambas as partes no que toca à configuração de parâmetros a serem usados para a autenticação de *peers* e, depois, para encriptar parte das trocas feitas entretanto (tal como as elaboradas na fase 2). Aqui também se autenticam mesmo os *peers* e se geram as chaves a serem usadas como geradoras de novas chaves. Esta fase inicia, portanto, um primeiro modo denominado de **main mode**, que requer a troca de seis pacotes (ida e volta), fornecendo segurança total ao longo do processo de estabelecimento de uma ligação IPSec[11].

main mode

A segunda fase de processamento inicia um segundo modo denominado de **quick mode**. Neste modo são negociados e acordados os parâmetros necessários para estabelecer uma ligação perfeitamente funcional de IPSec.

quick mode

Protocolos de autenticação

O processo de **autenticação** é tal que valida a aclamada identidade de uma entidade que se apresenta a um órgão. Esta validação é feita através de uma ou mais caraterísticas já conhecidas dessas entidades.

autenticação

11 Em alternativa ao main mode do ISAKMP é possível ativar o Aggressive Mode, onde é usada apenas metade das trocas de mensagens, mas fornecendo menos segurança, porque alguma da informação enviada segue em aberto.

Os mecanismos que atestam esta premissa devem possuir pelo menos uma entidade que está para ser autenticada num determinado espaço e uma que necessita de uma autenticação (como o fornecedor de serviço a quem a primeira se pretende conectar). Estas duas entidades, quando ligadas formam uma ligação ponto-a-ponto.

Vários protocolos foram criados para tentar resolver este tipo de situação, entre os quais, logo um dos primeiros a surgir, o **PPP** (em inglês *Point-to-Point Protocol*). Este protocolo emergiu como um protocolo de encapsulamento para o transporte de tráfego IP sobre ligações ponto-a-ponto, que suporta a configuração da ligação, deteção de erros, atribuição de endereços IP, negociação de endereços na camada de rede, multiplexagem de protocolos de rede e configuração de autenticação. Na Figura 6.6 podemos ver o *frame* do PPP.

PPP

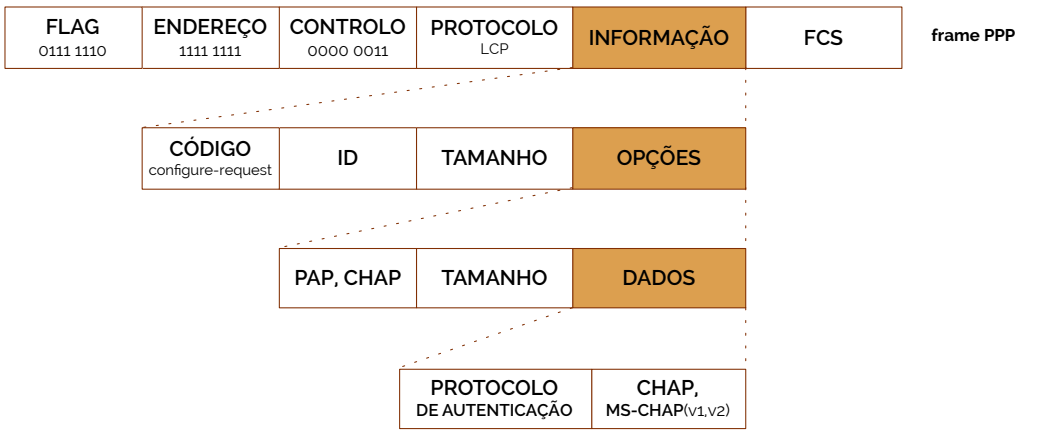


figura 6.6

Se verificarmos bem na Figura 6.6, escolhendo o protocolo LCP (*Link Control Protocol*) temos direito a escolher, mais à frente, nas opções, entre PAP e CHAP. Mas o que são o PAP e o CHAP?

O protocolo **PAP**, acrónimo inglês *Password Authentication Protocol*, é tal que fornece um método básico de autenticação com um parceiro através de um duplo *handshake*, fazendo passar uma palavra-passe, em aberto no *frame*, e aguardando pela aprovação ou rejeição. Este procedimento está exemplificado na Figura 6.7.

PAP

figura 6.7

Depois da ligação ter sido estabelecida, como vemos na Figura 6.7 o *login* e a palavra-passe são enviados repetidamente até que a autenticação seja feita e terminada.

O protocolo **CHAP**, acrónimo inglês para *Challenge-Handshake Authentication Protocol*, surgiu para tentar melhorar o mau trabalho de segurança que o protocolo anterior fazia, agora, usando um desafio cujo parceiro deveria responder com o valor *hashed* (em MD5), valor o qual seria confrontado com a solução. De tempos em tempos, o servidor envia um novo desafio para que o parceiro pudesse responder. Na Figura 6.8 podemos ver uma representação desta troca de mensagens num *three-way handshake*.

CHAP

Como o protocolo CHAP não teve muito sucesso dado que a segurança do mesmo não foi, ainda assim, suficiente, empresas como a Microsoft, embora sem muito sucesso, tentaram melhorar o mesmo algoritmo criando a sua própria versão a **MS-CHAPv1**. Esta versão, contudo, tinha problemas porque permitia a autenticação de forma unidirecional.

MS-CHAPv1

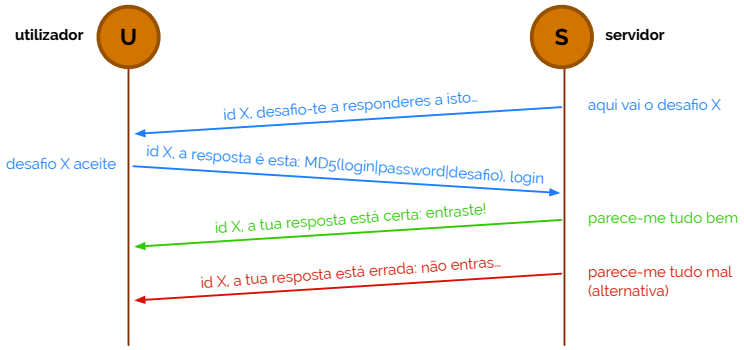


figura 6.8

Com o avançar dos tempos a Microsoft ainda lançou uma segunda versão mais apurada do MS-CHAP - a **MS-CHAPv2** - que ainda hoje se usa em conjugação com outros protocolos em algumas redes públicas com requisitos de autenticação, permitindo já a autenticação bi-direcional.

MS-CHAPv2

A Netscape - empresa dos anos '90, líder de criação de *software* para *browsing* da Internet, mais tarde passada para Mozilla - criara anos depois um protocolo chamado SSL ao qual lhe adicionaram capacidades de HMAC para normalizar um protocolo muito usado ainda hoje, que é o **TLS**, sigla de *Transport Layer Security Protocol*. Este protocolo garante privacidade e integridade de dados entre duas aplicações que comunicam entre si.

TLS

Por baixo, duas camadas de processamento são atuadas: um protocolo de *handshaking* que permite que o servidor e cliente se autentiquem entre si e negoceiem ambos algoritmos de encriptação e chaves criptográficas, e um protocolo de geração de registos, para especificar quando é que uma determinada ligação é privada ou pública.

Outros protocolos foram-se, depois, anexando ao TLS, como o EAP, já referido anteriormente, numa versão **EAP-TLS**, que cria uma ligação ao servidor de autenticação, segura. Para ser executado, no entanto, usa uma conexão sobre uma LAN específica, concordante com a norma **IEEE 802.1X**.

EAP-TLS

IEEE 802.1X

A norma IEEE 802.1X é a responsável por coordenar e detalhar o comportamento esperado dos equipamentos para controlo de acesso a uma rede (NAC), fornecendo mecanismos de autenticação para quem pretender conectar-se a uma rede de área local específica. Este tipo de aplicação segue protocolos arquiteturais de **AAA**, isto é, de *Authentication, Authorization and Accounting*.

AAA

Anteriormente, quando referimos a questão da segurança em WLANs referimos a implementação do RADIUS. Este é um exemplo de aplicação do protocolo AAA. Na Figura 6.9 podemos ver uma breve exemplifcação da topologia do protocolo AAA.

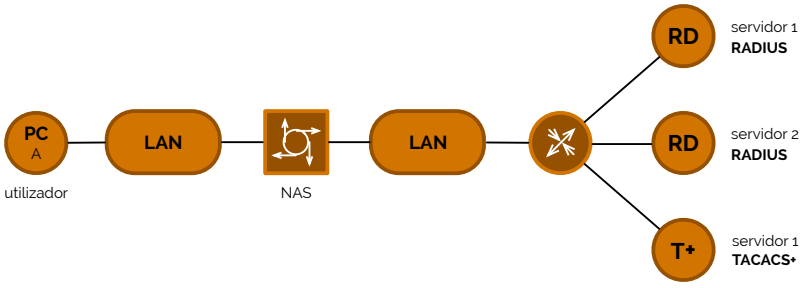


figura 6.9

Este protocolo ativa então a segurança no acesso sistematicamente através da autenticação de um utilizador por via de uma entidade denominada de **NAS** (*Network Access Server*), tal como da autorização sobre o que é que o utilizador pode fazer na rede e contabilidade do tempo de uso da rede para propósitos financeiros.

NAS

Na Figura 6.9, para além do protocolo RADIUS vemos também um denominado de TACACS+. O protocolo **TACACS+**, por extenso *Terminal Access Controller Access Control System Plus*, é tal que reencaminha um nome de utilizador e uma palavra-passe

TACACS+

para um servidor centralizado de segurança, que tanto poderá ser uma base de dados própria do TACACS como uma base de dados integrada com UNIX e com suporte TACACS. Esta prática separa as funcionalidades do protocolo-base AAA, usa TCP, com uma autenticação bi-direcional, todos os pacotes encriptados, mas tem uma personalização muito limitada em termos de contabilidade.

Por fim, a última *framework* lançada pela IETF para a próxima geração de servidores com protocolo AAA é a **DIAMETER**. Esta ferramenta é praticável com protocolos de mobilidade como o MIP (*Mobile-IP*) e é retro-compatível com o protocolo RADIUS.

DIAMETER

Redes privadas virtuais (VPN)

No contexto empresarial muitas vezes os trabalhadores necessitam de conseguir entrar no espaço de trabalho deles de forma remota, isto é, fora do escritório ou até mesmo fora do local de emprego. Para o poderem fazer necessitam de conseguir entrar na rede privada da empresa. Esta é uma das funções das **VPN**s - as redes privadas virtuais.

VPN

As VPNs podem ser de dois tipos genéricos - de acesso remoto ou *site-to-site*. Para estas últimas o IPSec é uma das formas de resolver este problema de contexto, tal como criar túneis IPSec conjugando-os com cabeçalhos GRE. Para resolver as VPNs do tipo de acesso remoto já o problema se torna mais complicado, ainda assim, havendo já um grande leque de possíveis soluções.

A primeira solução de todas são as VPNs **PPTP**. Tendo base no protocolo PPTP, que encapsula dados em tramas PPP dentro de pacotes IP, aqui é usada uma forma semelhante à do GRE para obter dados de e para o destino final. Suporta autenticação baseada em PAP, EAP, CHAP e MS-CHAP e usa duas chaves diferentes como elementos de cifra (através dos mecanismos de MPPE), que dirigem cada uma das chaves para cada uma das direções de comunicação, sendo derivadas da palavra-passe e dos desafios propostos pelo MS-CHAPv2. Este tipo de VPN só permite que seja criado um túnel por utilizador.

PPTP

O IPSec, para além da aplicação nas VPNs *site-to-site*, também permite a criação de VPNs de acesso remoto em conjugação com L2TP, onde a autenticação pode ser feita com Certificados Digitais (RSA) ou com os mesmos mecanismos autenticação PPP que em PPTP. Esta forma permite que se tenha integridade de dados, tal como autenticação da origem e proteção de repetições. Embora tenha um desempenho mais lento que no caso do PPTP, esta forma suporta vários túneis por utilizador.

Existem outras formas de criar VPNs de acesso remoto, entre os quais através de SSL e TLS, SSH ou OpenVPN (que usa uma variante de SSL/TLS).

Sistemas de segurança em redes: firewalls, IPS e outras aplicações

Um dos mecanismos de proteção de rede mais conhecido é a **firewall**. Este equipamento é capaz de fornecer um ponto único de defesa entre redes, protegendo uma rede das outras.

firewall

De uma forma mais generalizada, uma *firewall* é um sistema ou um grupo de sistemas que reforça políticas de controlo entre duas ou mais redes [10]. Os serviços de que dispomos ao usá-la são de NAT, autorização, redirecionamento de tráfego para máquinas específicas, *proxying*, análise de conteúdos, comunicações seguras e deteção e defesa de ataques de DoS (*Denial of Service*) e DDoS (*Distributed Denial of Service*). No fundo as *firewalls* podem trabalhar num largo espetro do modelo OSI, conseguindo cobrir as camadas de ligação à camada de aplicação.

As redes empresariais podem ser divididas em múltipas **zonas seguras** com diferentes níveis de segurança. Por exemplo, uma zona que podemos considerar como "semi-protegida" deverá ser a **DMZ** (zona desmilitarizada), que é um perímetro específico fora do regime interno da rede, onde se alojam serviços e servidores de domínio público. Por esta mesma razão é sempre conveniente pensar que qualquer máquina ligada na DMZ está em risco em termos de segurança.

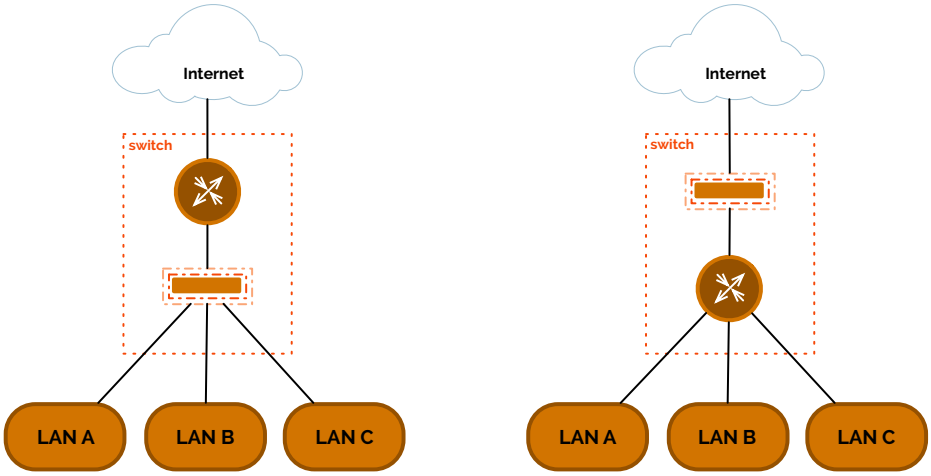
zonas seguras

DMZ

Pensando na divisão de uma rede por vários níveis de segurança somos automaticamente levados a pensar na seguinte questão: "então onde é que colocamos as nossas *firewalls*?". Ora, as *firewalls* deverão estar localizadas um pouco ao longo de toda a rede, mas essencialmente, mais que tudo, à entrada da nossa rede privada, como mostram as duas soluções, perfeitamente viáveis, da Figura 6.10, num **contexto individual**.

contexto individual

figura 6.10



Se tivermos várias VLANs abaixo da camada de núcleo também podemos colocar *firewalls* à entrada destas, num **contexto múltiplo**.

contexto múltiplo

Para diminuir a possibilidade de ataques de DoS podemos obrigar a que o tráfego se disperse por várias *firewalls*, num conceito de **balanceamento de cargas**, diminuindo também, por conseguinte, requisitos de memória e processamento de cada *firewall*.

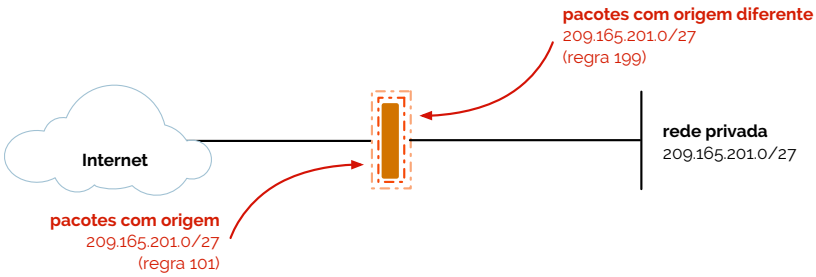
balanceamento de cargas

Um dos ataques mais bem sucedidos a redes é o envio de pacotes IP com origens falsas - dizemos que isto acontece com o nome **IP Spoofing**. Esta prática permite que a identidade do emissor da mensagem seja oculta ou possa passar por outra, localizada numa rede específica.

IP Spoofing

Este tipo de ataques, tendo *firewalls* instaladas pela rede (especialmente à entrada desta) não só deverá ter regras (políticas) para limitar e assinalar pacotes provenientes de fora com um determinado endereço (ou gama de endereços e *multicast*), como também deverá ter as suas regras para permitir o tráfego de gamas de endereço na direção contrária e bloquear as restantes. Através das *access-lists* da Cisco, por exemplo, seria possível, na rede da Figura 6.11, prevenir o IP *Spoofing* através da configuração do Código 6.1.

figura 6.11



código 6.1

```
R1# configure terminal
R1(conf)# interface <nome-da-interface>
R1(conf-int)# ip access group 101 in
R1(conf-int)# ip access group 199 out
R1(conf-int)# exit
R1(conf)# access-list 101 deny ip 209.165.201 0.0.0.31 any
R1(conf)# access-list 101 deny icmp any any redirect
R1(conf)# access-list 101 deny ip 224.0.0.0 31.255.255.255 any
R1(conf)# access-list 101 deny ip 240.0.0.0 15.255.255.255 any
R1(conf)# access-list 101 deny ip 127.0.0.0 0.255.255.255 any
R1(conf)# access-list 101 deny ip host 0.0.0.0 any
R1(conf)# access-list 101 deny ip 10.1.1.0 0.0.0.255 any
R1(conf)# access-list 101 deny ip 172.16.0.0 0.15.255.255 any
R1(conf)# access-list 101 deny ip 192.168.0.0 0.0.255.255 any
```

```
R1(conf)# access-list 101 permit ip any any
R1(conf)#
R1(conf)# access-list 199 permit ip 209.165.201.0 0.0.0.31 any
R1(conf)# access-list 199 deny ip any any
R1(conf)# end
```

O IP *Spoofing* também poderá ser controlado através do **IP Source Guard**, que é uma funcionalidade integrada na camada 2, prevenindo assim tráfego IP proveniente de portas não conhecidas a clientes com endereços IP atribuídos, através da filtragem de endereços com um endereço IP de origem (não o atribuído por DHCP ou outra configuração estática).

IP Source Guard

Os ataques de DoS geralmente usam, em camada 4 do modelo OSI, comunicações TCP mas que ficam retidas a meio do processo, isto é, permanecendo semi-abertas. Este problema, das **conexões TCP semi-abertas**[12] pode ser identificado pelas *firewalls* uma vez que estas poderão manter o estado das sessões TCP em memória, condicionando tempos para que estas permaneçam abertas, o que irá prevenir que estas eperdam o poder de processamento e cortem estes processos a tempo.

conexões TCP semi-abertas

Portanto, como podemos ver, uma das soluções possível para este tipo de situações passa muito por haver uma contagem ou estimativa da taxa de transmissão usada nos canais de comunicação ao longo de toda a rede, pelas *firewalls*. Mais um exemplo - o **CAR**, de *Committed Access Rate* - limita (uma classe de) tráfego a uma taxa de transferência específica, pelo que evita que uma só origem envie tráfego acima de um *threshold* pré-definido.

CAR

No meio envolvente das VPNs as *firewalls* também têm de estar presentes, sendo que precisam de conseguir filtrar todo o tráfego.

7. Mecanismos de Gestão de Redes

Até agora, desde o início deste documento, temos vindo a incrementar o nível de complexidade e a dimensão das redes que estamos a analisar, mas continuamos no nicho dos sistemas autónomos, isto é, das redes empresariais, tendo os pisos dos edifícios da empresa como unidade base do nosso trabalho. Em Arquitetura de Redes Avançada (a4s1) iremos subir ainda mais um nível, referindo como é que vários sistemas autónomos interagem entre si e que tipos de rede estabelecem entre eles.

No meio de toda esta complexidade como é que são mantidas a ordem e capacidade de reparar a rede ao longo do tempo? Consideremo-nos como administradores de redes: como é que podemos saber, no mínimo, o estado da nossa rede a qualquer momento? Vamos ter com cada um dos equipamentos de rede localizados nos edifícios e testamos o seu estado?

Ao longo de todo um trabalho de administração em redes o número de documentação aumenta drasticamente: desde dicionários de endereços, a dicionários de nomes, a mapas e diagramas para sinalizar mudanças de paradigmas ou outros, num ambiente de produção todo este procedimento poderá causar atrasos que serão prejudiciais para o adminstrador de redes e para a própria empresa.

A solução destes problemas passa por ter configurações automáticas para a **manutenção** do sistema, que deverá, sempre que necessário, exibir todas as informações que estão em curso, de forma atual e fácil de usar. Esta informação, combinada apenas com diagramas e outra documentação mais pequena consegue relevar mais informação útil no mais pequeno intervalo de tempo.

manutenção

Modelos de gestão de redes

Existem inúmeras formas de aplicar rotinas de manutenção a uma rede empresarial. Dados padrões que se foram criando com as ações de reparação e verificação de estados

12 Uma conexão TCP semi-aberta consiste num SYN enviado de um computador A para B, seguido da resposta SYN ACK, ficando-se por aqui, à espera que o último ACK chegue de A.

de funcionamento, criaram-se **modelos de gestão** de redes, como o TMN (*Telecommunications Management Network*) para redes de telecomunicações, o OAM&P (*Operation, Administration, Maintenance and Provisioning*), o TOM (*Telecoms Operations Map*) ou o FCAPS (*Fault, Configuration, Accounting, Performance and Security*) que serve para todos os tipos de redes, como iremos abordar de seguida.

modelos de gestão

Todas as especificações do **FCAPS** trabalham em uníssono, sendo que por cima de todas está a segurança, que é completamente transversal a todos os núcleos deste modelo de gestão. Vejamos assim cada uma das valências deste modelo.

FCAPS

A **gestão de falhas** é o conjunto de funções que permitem a deteção, o isolamento, e a correção de operações anormais em redes de telecomunicações. Aqui testam-se as garantias de fiabilidade, disponibilidade e resiliência dos sistemas, a capacidade de vigilância e alarmes do próprio sistema[13], a localização de falhas (tal como a sua correção), o seu teste e a adminstração de problemas, isto é, a capacidade de recolher dados diretamente de clientes que usam uma rede e reportam problemas ou passam por estados de verificação.

gestão de falhas

A **gestão de configuração** fornece funções para identificar, recolher dados de configuração, exercer controlo sobre e fornecer dados de configuração a elementos de uma rede. Aqui instalam-se as configurações dos equipamentos físicos e lógicos, planeiam-se e negoceiam-se os serviços a prestar, provisionam-se as capacidades necessárias para garantir serviços, recebem-se estados das máquinas e procede-se à engenharia por si, da rede.

gestão de configuração

A **gestão de contabilidade** permite que sejam feitas medidas sobre os serviços prestados na rede e, assim, determinar custos para o fornecedor de serviços, cobrando o cliente.

gestão de contabilidade

A **gestão de desempenho** permite avaliar e reportar o comportamento de equipamentos de rede instalados, recolhendo e analisando dados estatísticos para monitorização da mesma, corrigindo depois comportamentos que não fossem esperados.

gestão de desempenho

A **gestão de segurança**, enquanto membro transversal a todos os processos anteriores, consiste em fornecer serviços para autenticação, controlo de acessos, confidencialidade e integridade de dados e deteção de eventos inesperados ou que possam incorrer em situações perigosas para a rede e seus clientes. [11]

gestão de segurança

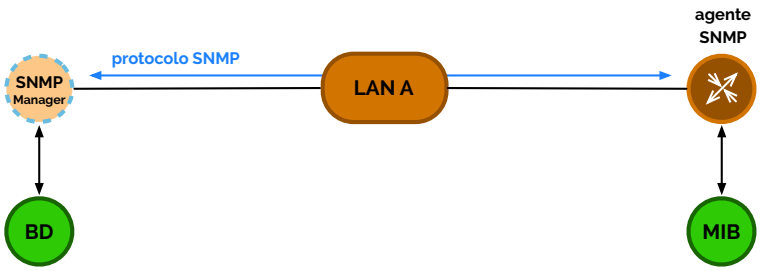
Simple Network Management Protocol (SNMP)

Uma forma fácil de obter informação sobre a rede e os seus equipamentos é usando o protocolo de gestão **SNMP**, sigla de *Simple Network Management Protocol*. Este protocolo consiste numa rede implementada de gestão consituída por um conjunto de **agentes** (módulo de *software*) que residem em **dispositivos geridos** (*managed devices*), que recolhem e guardam informação de gestão de redes. As várias informações guardadas podem ser acedidas através de **Network-Management Systems** (NMSs), que executam aplicaões capazes de monitorizar e controlar os dispositivos geridos.

SNMP
agentes
dispositivos geridos
Network-Management Systems

O SNMP é um protocolo gerido pela IETF, desenhado para facilitar a troca de informação de gestão entre equipamentos de rede, seguindo a topologia-base da Figura 7.1.

figura 7.1



Os agentes SNMP, localizados nos dispositivos geridos, são tais que guardam as suas informações em bases de dados denominadas **MIB**. O MIB, acrónimo para *Manage-*

MIB

13 A própria capacidade de se autoavaliar em tempo-real e alertar situações perigosas para o meio de rede.

ment Information Base, é uma coleção de objetos geridos no meio de rede organizado em vários módulos.

Para interagir com os módulos SNMP apenas se tem de escolher um de três comandos possíveis: ler, escrever ou preparar uma *trap*. O **comando de leitura** é usado por um NMS para monitorizar dispositivos geridos, examinando diferentes variáveis mantidas por estes dispositivos. O **comando de escrita** é usado por um NMS para controlar dispositivos geridos, examinando diferentes variáveis mantidas por estes dispositivos. Por fim, o comando para **preparar uma trap** é usado para assincronizadamente reportar eventos para o NMS.

comando de leitura

comando de escrita

preparar uma trap

O SNMP funciona essencialmente por **polling**, isto é, questionando periodicamente o agente SNMP por nova informação, o que é bom por um lado, porque assim o *manager* controla completamente o equipamento de rede, conhecendo bem os detalhes do envolvente, mas também é mau, porque cria um atraso entre o evento e a sua entrada no sistema, gerando inclusive *overhead* no canal de comunicação.

polling

Esta noção de *polling*, conjugada com a possibilidade de fazer *traps*, permite que quando uma *trap* é lançada, o *manager* possa efetuar pedidos de mais informação.

Existem várias versões do SNMP, entre as quais a versão 1, 2c e 3. Na sua versão inicial os processos de autenticação e de autorização eram baseados no conceito de **string de comunidade**, que identifica as permissões da máquina que acedia ao agente: num processo só-de-leitura ou de leitura e escrita. Por defeito, todos os sistemas são configurados com as *strings* `public` e `private`, sendo a primeira só-de-leitura e a segunda de leitura e escrita.

string de comunidade

Nas versões seguintes criou-se uma normalização da estrutura de informação de gestão (SMI), criando inclusive novas operações protocolares (versão 2c), e criou-se um novo formato de mensagem, com segurança inerente e controlo de acesso. Esta segurança passa por questões de controlo de acesso dependente do utilizador, com autenticação por chaves partilhadas e questões de privacidade melhoradas, através de fenómenos de cifra.

Mensagens SNMP por versão

Na primeira versão existiam três tipos de mensagem SNMP. A estrutura-base para as três mensagens é visível na Figura 7.2.

figura 7.2

No início de todas as mensagens existem dois campos denominados de versão e comunidade. O campo da comunidade pretende designar, então, o nome desta, usado para termos de autenticação entre um agente e o NMS.

O terceiro campo de uma mensagem SNMP, o **PDU** (*Protocol Data Unit*), pode, então, ser um de três tipos designados na Figura 7.3.

PDU

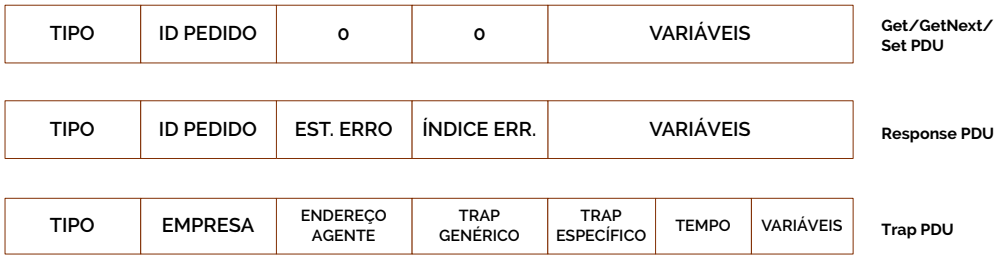


figura 7.3

O primeiro tipo de todos é constituído por um PDU de **GetRequest** que serve para obter uma entrada do MIB, identificada pelo campo de variável. O **GetNextRequest** pemite obter vários registos, sendo usado para percorrer a estrutura de dados armazenada no MIB, até obter o registo pretendido do campo variável. O **SetRequest** permite fazer

GetRequest

GetNextRequest

SetRequest

uma atribuição de um valor a uma variável, conform indicado no campo variável da trama do PDU [12].

Em baixo ainda temos o PDU **GetResponse**, resposta dos três primeiros PDUs e o PDU para as *traps*, onde é trazida uma mensagem inesperada (não solicitada) sobre o estado do sistema quando algum sinal despoletou um alerta.

GetResponse

Com a versão 2c do SNMP adicionou-se um novo pacote denominado **GetBulk**, que corresponde ao anterior GetNext, sendo que aqui juntam-se dois campos `NON-REPEATERS` e `MAX-REPETITIONS` que permitem que o NMS obtenha dados de muitos objetos de gestão de um agente SNMP. O PDU de *trap* também foi alterado, adotando a mesma forma que o Get/GetNext/Set PDU do SNMPv1. Ambos PDUs podem ser vistos na Figura 7.4.

GetBulk

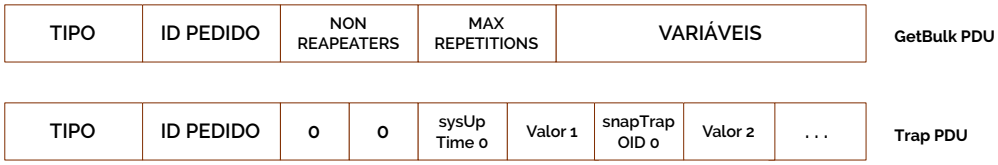


figura 7.4

Com a versão 3 a própria mensagem mudou por completo, de forma a poder comportar as questões de segurança que foram acrescentadas com esta atualização do protocolo. Contudo, apesar das alterações à mensagem, os PDUs mantêm-se iguais aos da versão 2c. A estrutura da mensagem SNMPv3 está visível na Figura 7.5.

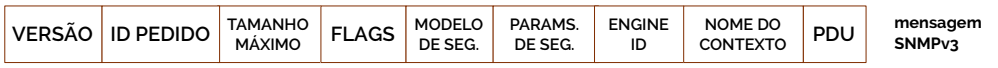


figura 7.5

Estrutura de organização do SNMP (SMI) e módulos MIB

Como já tivemos oportunidade de verificar, um **módulo MIB** é uma especificação de uma informação de gestão num dispositivo de rede. Estes módulos encontram-se todos estruturados numa forma de árvore por uma norma denominada de **SMI** (*Storage of Management Information*), onde cada recurso gerido é representado por um objeto que, por sua vez, é identificado de forma inequívoca por um **OID** (*Object Identifier*), que representa, também, o caminho a percorrer na árvore até chegar ao registo pretendido.

módulo MIB

SMI

OID

Estes objetos podem ser **nomeados**, pelo que na hierarquia formada pela árvore, cada folha assume-se que possui um nome e um número. Consideremos a seguinte árvore SMI da Figura 7.6.

nomeados

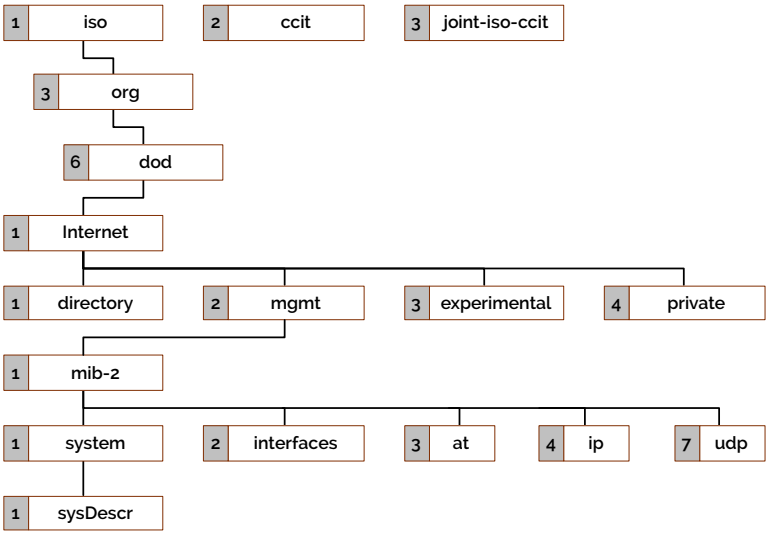


figura 7.6

Para representarmos cada objeto temos então um caminho a corresponder na árvore para o identificar, por exemplo, e como podemos verificar na Figura 7.7, para chegar ao objeto MIB-II basta escrever 1.3.6.1.2.1.

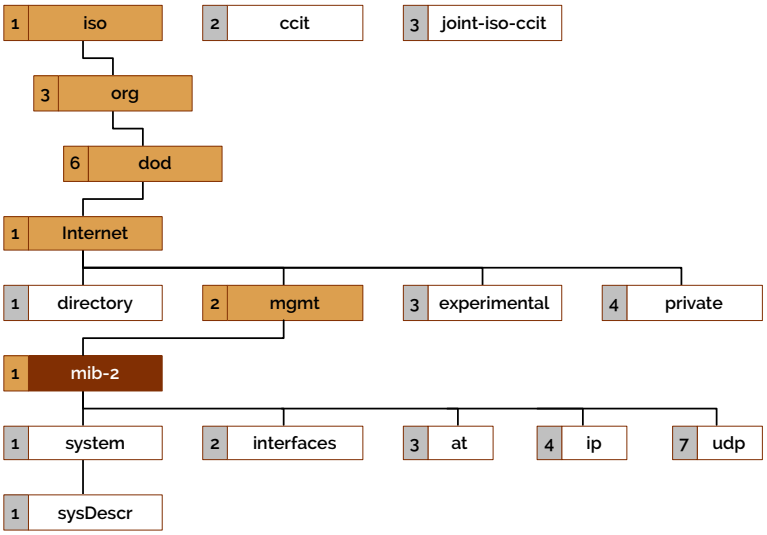


figura 7.7

Já para chegar ao objecto UDP temos que percorrer a árvore com a identificação 1.3.6.1.2.1.7.1, como podemos ver na Figura 7.8.

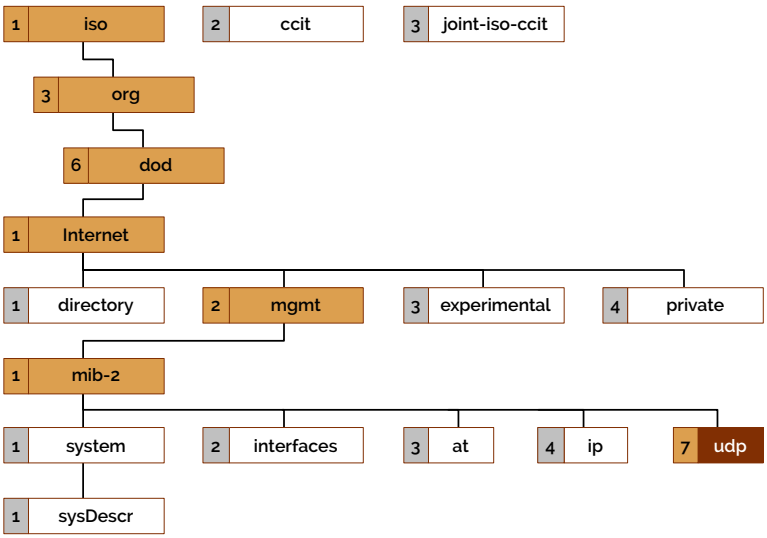


figura 7.8

Cada objeto da árvore SMI possui, tal como em qualquer outra estrutura de dados, um tipo de dados, um estado e uma semântica própria. Este **tipo de dados** pode ser um inteiro, um contador (que é basicamente um inteiro também), uma *string*, um objeto identificado, um endereço IP, entre outros...

tipo de dados

Consciencialização do estado de uma rede

Quando se tem uma rede já instalada, para além dos esforços que devem ser feitos para saber mapear e bem documentar a mesma, é também importante **recolher dados** de forma a saber auscultar a rede e determinar os locais por onde o tráfego de pacotes é feito. A Cisco criou um sistema que o permite fazer, denominado **NetFlow**, que se baseia num processo em que os vários equipamentos de rede recolhem dados enquanto há tráfegos e, depois de armazenados numa unidade própria chamada **NetFlow Collector**, permite que outros analisem os seus dados, de forma a poder fornecer melhores serviços aos utilizadores

recolher dados

NetFlow

NetFlow Collector

da rede, como melhores critérios de qualidade de serviço, segurança, combate aos ataques de refusa de serviço (DoS) ou até *data mining* para propósitos de *marketing*.

A IETF, por outro lado, tentou normalizar um processo denominado de **IPFIX** (acrónimo de *IP Flow Information eXport*) bastante semelhante ao NetFlow.

IPFIX

Na mesma altura em que o IPFIX estava a ser desenvolvido, outras equipas estavam a desenvolver o protocolo **PSAMP**, pelo que havia mesmo uma verdadeira necessidade de normalizar estes processos de captura de informação para melhorar os serviços fornecidos.

PSAMP

O foco principal do grupo que desenvolveu o PSAMP passou então a ser: a especificação de um conjunto de operações pelos quais os pacotes são "capturados" (somente a nível qualitativo); a especificação de informação que será disponibilizada para reportar mais tarde; a descrição de protocolos pelos quais a informação reportada dos pacotes é entregue a aplicações; a descrição de protocolos pelos quais a seleção de pacotes e os seus relatórios são configurados. Assim sendo, a equipa de desenvolvimento do PSAMP optou por trabalhar para usar o protocolo IPFIX e melhorá-lo com as exportações do protocolo PSAMP.

Ferramentas de monitorização e gestão em resumo

Em suma, das várias ferramentas que abordámos neste documento (e outras), para monitorização (e alerta) usamos avaliação do uso das ligações, dos fluxos, ... e protocolos como o SNMP.

Para fazermos controlo de gestão da rede podemos usar SNMP e, para executar vários comandos SNMP, podemos usar *scripting* em consola ou outra ferramenta.

Em ambiente de investigação, onde há a necessidade de testar capturas de pacotes em massa e num modo mais avançado, usamos preferencialmente ferramentas da família do **PCAP**, como o *Wireshark* ou o TCPdump.

PCAP

O uso integrado destas ferramentas tem o intuito de passarmos a poder analisar o estado de uma rede de forma local e sem grandes equipamentos intermediários, sendo que, em simultâneo, não estamos a invadir a privacidade e segurança de nenhum utilizador ou rede. Para isto prevê-se um futuro em que a ideia de monitorização é distribuído, heterogéneo (diferente de rede para rede, consoante a necessidade em cada uma) e inteligente.

8. Qualidade de Serviço

Ao longo deste documento já referimos inúmeras vezes o termo **qualidade de serviço** (QoS), mas afinal do que é que se trata?

qualidade de serviço

Muitas vezes, enquanto estamos a usar serviços nas nossas redes, em termos de aplicações somos confrontados com baixas taxas de transferência. Ou então, quando estamos a fazer uma video-chamada esta por vezes para, o que mostra, depois, haver um grande atraso nas ligações (cada vez maior ao longo do tempo). Entre outras situações, estes casos são geralmente consequência de limitações por parte do não aproveitamento ótimo (ou próximo de ótimo) das caraterísticas físicas da rede - o que se revela por haver falta de largura de banda (uma vez que muitos fluxos estão a concorrer para um mesmo recurso), demasiados atrasos (os pacotes terão de atravessar muitos equipamentos de rede e ligações que só ajudam a somar para uma latência total), atrasos variáveis (por vezes acontece haver outros tipos de tráfego que só ajudam a haver ainda mais atrasos de forma totalmente variável) e pacotes descartados (por vezes os pacotes têm de ser descartados, quando já não podem ser guardados em mais lado nenhum.

Aplicação de parâmetros de QoS nas tramas Ethernet

E tudo começa com a camada 2 do modelo OSI (a camada de ligação), que complementa as tramas Ethernet. Nestas, é necessário que os *switches* tirem partido da norma

IEEE 802.1p para que possam ser fornecidos critérios de qualidade de serviço. Sendo a norma IEEE 802.1p parte da IEEE 802.1Q - norma que define o conceito de *tagging* das VLANs -, num dos campos da trama 802.1Q temos um espaço denominado de `TAG CONTROL INFORMATION` onde se diferenciam **classes de serviço**.

classes de serviço

Os três bits mais significativos do campo `TAG CONTROL INFORMATION` são conhecidos como **Priority Code Point** (PCP) e são usado para definir a **prioridade do frame**. Este valor pode ser definido através do porto de chegada, através de um pacote de um terminal com 802.1Q ativo ou através de parâmetros de qualidade de serviço da camada de rede, como iremos falar de seguida, muitas vezes referidos como valores DSCP. [13]

Priority Code Point, prioridade do frame

Problemas no uso de rede sem regras de QoS

Como referimos no início deste capítulo, existem pelo menos quatro problemas que usualmente acontecem quando não se tem regras de qualidade de serviço instauradas numa rede: atrasos *end-to-end*, atrasos variáveis de processamento, perda de pacotes e pouca largura de banda. Analisemos cada um destes problemas e tentemos propor soluções para cada um destes.

Os **atrasos end-to-end** acontecem em cenários como o representado na Figura 8.1.

atrasos end-to-end

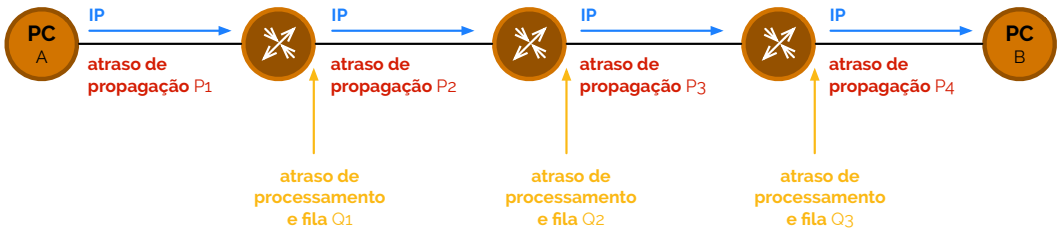


figura 8.1

Num cenário como o representado na Figura 8.1 podemos ter uma estimativa do atraso total se somarmos as parcelas todas de atraso de propagação *Px* com as parcelas do atraso de processamento e fila por parte dos *routers* *Qy*. Como referido anteriormente, o grande problema nestes casos são os atrasos *Qy*, dado que estes são inconstantes ao longo do tempo, uma vez que dependem estritamente do uso que os *routers* estão a ter com outros controlos de fluxos.

Para resolvermos este problema, uma possível solução está na atualização da ligação, permitindo que o atraso de propagação diminua. Contudo, isto apenas resolve o problema que é consideravelmente menor aos atrasos de processamento e de fila, pela instabilidade dos segundos. Para resolvermos este problema teremos de arranjar novas formas de fazer as nossas filas e de selecionar o que é que é tráfego priveligiado ou não - isto são critérios de qualidade de serviço, que iremos falar adiante.

Outras soluções poderiam passar por comprimir os dados (*payload*) das tramas de camada 2 (que se pode tornar num processo muito moroso) ou até mesmo do seu cabeçalho.

Este **atraso de processamento e fila** acontece porque os *routers* também têm um processador e memórias próprias, pelo que é natural que se tome algum tempo para que este resolva cálculos tendo base um conjunto de pacotes e que cause algum atraso a manipular a fila de pacotes que se encontra à saída deste. À soma entre os dois atrasos damos o nome de **atraso de serialização**.

atraso de processamento e fila

atraso de serialização

E com este último problema entramos na questão da **perda de pacotes**. Este evento acontece quando a fila está cheia, pelo que quando chega um pacote para a fila, então este terá de ser descartado, por **tail-drop**, uma vez que não há mais espaço na fila para ele. A qualidade de serviço poderia entrar em ação aqui, dado que há a possibilidade, com o cabeçalho 802.1p ativo, de sinalizar que um determinado pacote que chega deverá tomar o lugar de algum em prol da qualidade de um determinado serviço para o qual ele é essencial. Outros métodos de prevenção deste acontecimento, embora mais caros, passam por atualizar as ligações e a largura de banda para a transmissão de pacotes.

perda de pacotes

tail-drop

Por último, a **largura de banda** por vezes mostra-se curta para o tratamento de tanto pacote em simultâneo. Ao longo de uma rede, a largura banda máxima, medindo-se como a largura de banda da ligação mais fraca (é o mínimo de todas as larguras de banda ao longo de um caminho), então uma possível solução para este problema passasse por tirar largura de banda a aplicações que precisem menos dela.

largura de banda

Algoritmos de escalonamento

Antes de avançarmos para os algoritmos de escalonamento, relembremos apenas três conceitos que nos serão úteis para o futuro:

- **fluxo**: uma instância única de um fluxo de aplicação para aplicação, de pacotes identificados pelo seu endereço de origem, porto de origem, endereço de destino, porto de destino e identificação do protocolo que usam;
- **stream de tráfego**: um conjunto administrativamente significativo de um ou mais fluxos que atravessam um segmento de um caminho, podendo consistir num conjunto de fluxos ativos que são selecionador por um classificador em particular;
- **perfil de tráfego**: descrição de propriedades temporais de um *stream* de tráfego, tais como a média, taxa máxima e tamanho do *burst*.

Agora, voltando à questão dos algoritmos, temos que um **algoritmo de escalonamento**, tal como estudámos na disciplina de Sistemas de Operação (a3s1), é tal que decidem a ordem, neste caso, com que os pacotes de diferentes fluxos são servidos numa fila. Bons algoritmos garantem sempre que o servidor (*router*) não está ocupado se e só se não existe qualquer pacote à espera de ser servido.

algoritmo de escalonamento

Um primeiro exemplo de algoritmo - e também o mais simples e clássico de todos - é a **fila FIFO** (*First In, First Out*). O seu processo está representado na Figura 8.2.

fila FIFO

figura 8.2

Neste tipo de fila, como já a conhecemos de outras disciplinas de programação (principalmente), a política de que o primeiro elemento que entra também é o primeiro que sai não permite a aplicação de quaisquer critérios de qualidade de serviço, dado que não é possível diferenciar os pacotes e, assim, modificar a ordem da sua disposição em pleno processamento.

Evoluindo a partir desta lógica de fila entramos na criação de **filas de prioridade**, que já permite a diferenciação, por conseguinte, alguns critérios de qualidade de serviço. A sua representação está feita na Figura 8.3.

filas de prioridade

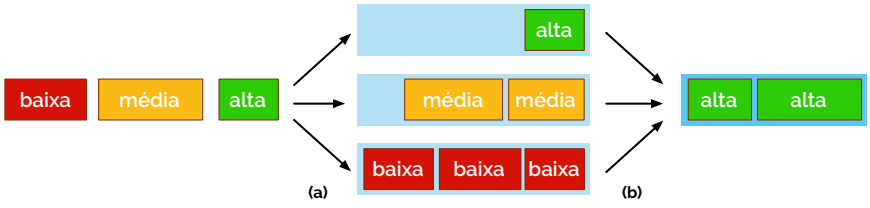


figura 8.3

Na Figura 8.3 podemos então reparar que os pacotes, à medida que vão chegando, sofrem um processo de classificação, ordenando-os entre prioridades altas, médias e baixas, todas, em filas diferentes. Depois começa a fase de prioridade estática, onde primeiro se esvazia a fila dos pacotes com prioridade alta, seguido dos de média e terminando nos de baixa. Este processo, no entanto, prefere em demasia os pacotes de elevada prioridade, deixando aos de baixa uma qualidade de serviço muito má.

Para colmatar as falhas do algoritmo de escalonamento por filas com prioridade, decidiu-se criar um novo método que distribuisse pacotes para a fila final, de forma justa.

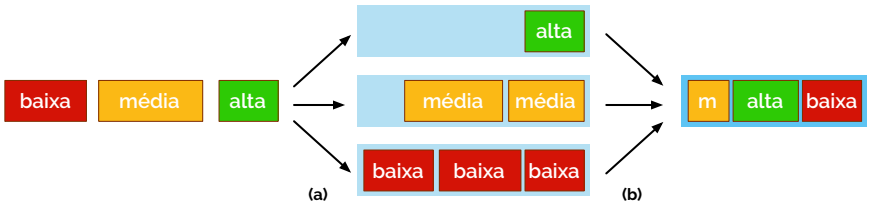


figura 8.4

Agora, conforme exibido na Figura 8.4 podemos reparar que, embora ainda haja um pacote de prioridade alta para ser distribuído, pacotes de média e baixa prioridade já se encontram na fila final, num algoritmo denominado de **filas de prioridade justa** (em inglês *fair queueing*, FQ).

filas de prioridade justa

Para melhorar ainda mais, em (b), da Figura 8.4, podemos distribuir os pacotes através do seu peso nos fluxos, estimando a necessidade que o público tem perante tais valores. Com este método (**weighted fair queueing**, WFQ) é permitido que cada fila tenha a percentagem da largura de banda da ligação que é, pelo menos, igual ao seu peso dividido pela soma de todos os pesos de todas as filas.

weigthed fair queueing

Caso estes algoritmos falhem é também importante saber criar/usar algoritmos que estejam preparados para evitar a perda de pacotes que sejam importantes, sabendo discriminá-los quanto antes. Assim sendo cria-se o **Random Early Detection**, RED, que é um mecanismo de tentativa de evitar congestionamento, através do controlo de congestionamento do TCP, perdendo pacotes aleatoriamente antes de períodos de largo congestionamento. Isto permite que depois, indiretamente, seja dito ao transmissor dos pacotes, que diminua a taxa de transferência.

Random Early Detection

Da mesma forma que foi feito com a criação de filas com prioridade e critério de justiça (FQ), aqui também podemos aplicar pesos no algoritmo RED, criando o algoritmo **WRED**, onde o descarte de pacotes é feito com base em prioridades de classe, onde tráfego de maior prioridade é enviado com uma maior probabilidade que o tráfego menor.

WRED

Em suma, estas foram as bases para a aplicação de qualidade de serviço no seu **melhor esforço** com os recursos já presentes na rede. Mas estas políticas não poderão ser aplicadas apenas desta forma. De seguida iremos abordar uma segunda forma de abordar este problema, através de **serviços integrados** (quando as aplicações sinalizam que precisam de estados de qualidade de serviço específicos), mas ainda teremos **serviços diferenciados** (onde a rede reconhece classes que requerem especiais atenções ao nível de QoS).

melhor esforço

serviços integrados

serviços diferenciados

Arquitetura de serviços integrados (IntServ)

O **modelo de serviços integrados** (normalizado no RFC 1633) foi introduzido como forma de garantir comportamentos previsíveis de uma rede perante uma ou mais aplicações. No fundo, para fluxos que tenham qualidade de serviço como requisito, será necessário, antes, haver uma reserva específica de caminhos para fluxo entre a origem e o destino das comunicações, por uma aplicação. Contrariamente ao que vimos até agora, aqui a rede implementa um mecanismo que controla a admissão de reservas, denominado de **Call Admission Control**, CAC, onde os fluxos que não recebem este tratamento são considerados como "melhor esforço" (em inglês *best effort*).

modelo de serviços integrados

↬ RFC 1633

Call Admission Control

Dentro da lógica de serviços integrados temos duas **classes**, entre as quais, a **Controlled Load** (apresentada no RFC 2211) que fornece um serviço muito semelhante ao "*best effort*", mas numa rede sem-congestionamentos e onde as estações terminais deverão sentir que uma maior percentagem dos seus pacotes está, de facto, a ser enviado com atrasos muito próximos de zero.

classes

Controlled Load

↬ RFC 2211

A segunda classe dentro desta lógica é a de **Guaranteed Service** (apresentada no RFC 2212), que fornece a capacidade de se marcar um valor máximo para os atrasos de todos os pacotes IP. Ambas as classes necessitam que o emissor prepare o processo de envio do pacote de acordo com um modelo de "*token bucket*".

Guaranteed Service

↬ RFC 2212

Comunicação de necessidade de recursos entre equipamentos

Dentro do âmbito da integração de serviços para a aplicação de regras de qualidade de serviço, uma das ações que se poderiam tomar de forma automática seria a partilha de informações em termos de necessidades de recursos de terminais e outros equipamentos de redes. Surge assim o **RSVP**, sigla de *Reservation Protocol* (definido em ambos RFC 2205 e 2215), que permite que uma determinada origem descreva as caraterísticas de um fluxo de pacotes IP, um determinado destino descreva a reserva de recursos que pretende e que os *routers* saibam como processar tais fluxos de pacotes, de forma a satisfazer os pedidos de reserva.

RSVP

↬ RFC 2205

↬ RFC 2215

Este protocolo, estando já encapsulado sobre o pacote IP e identificado com ID 46, tem uma sinalização baseada em apenas dois tipos de mensagens: a **mensagem PATH** (tipo 1), que possui especificações para o fluxo de dados - contém os parâmetros que descrevem a fonte do tráfego com base no modelo de *token bucket*, sendo que inclui três objetos RSVP obrigatórios (um identificador de uma sessão com um endereço de destino IP, um porto e um protocolo, um *hop* que indica qual é o *router* seguinte através do seu endereço IP e porto, e um período de tempo entre envios com sucesso de mensagens PATH); a **mensagem RESV** (tipo 2), que enviará ou uma especificação de filtro ou uma especificação de reserva de fluxo. No primeiro caso a mensagem trará um descritor de fluxo que permite que *routers* identifiquem que pertençam a uma dada reserva, através de um endereço de origem, destino, protocolo, porto de origem e destino ou qualquer outra combinação destes valores anteriores. Já no segundo caso a mensagem trará parâmetros que descrevem a reserva que quem recebe quer que seja suportada: envia-se uma especificação onde é especificado se o recetor quer um serviço do tipo "serviço garantido" ou, quando tal não está especificado, significa que o recetor quer um serviço do tipo "*controlled load*".

mensagem PATH

mensagem RESV

Na Figura 8.5 podemos ver um exemplo de aplicação da sinalização RSVP onde, como descrevemos, as mensagens PATH servem para reconhecer os caminhos e as mensagens RESV servem para criar agregação de reservas.

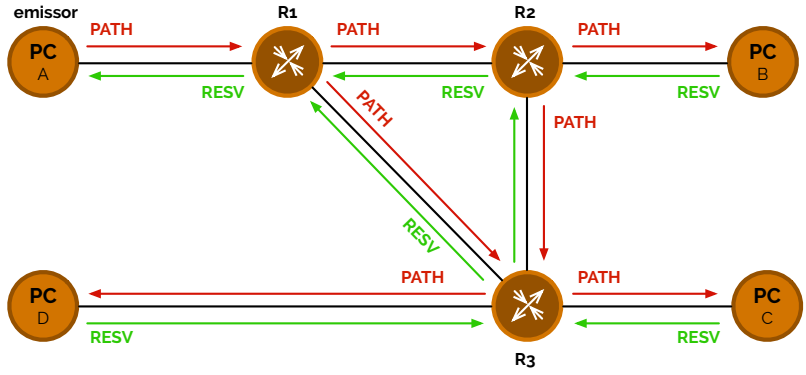


figura 8.5

Nas mensagens RESV é possível fazer reservas específicas indicando **filtros**. Estes podem ser **fixos**, se se pretender fazer uma reserva específica por cada emissor, **wildcards** se se pretender especificar uma reserva única para receber tráfego de um conjunto de emissores ou **explícitos**, se o recetor especificar uma lista de emissores dos quais pretende receber informação e um único valor para receber tráfego de emissores específicos.

filtros

fixos, wildcard

explícitos

Na verdade não existem apenas dois tipos de mensagens, mas as outras mensagens que existem (tipos ERR, TEAR e CONFIRMATION para cada RESV e PATH) servem para confirmar, indicar erros ou terminar reservas.

As vantagens do uso de RSVP estão essencialmente no facto de se usar um modelo simplex (de multiponto para multiponto), com reservas iniciadas pelos próprios recetores e temporizadas. Este protocolo também permite uma separação entre as reservas de recursos, encaminhamentos e filtragem de pacotes, com diferentes estilos adjacentes e com agregação de reservas.

Arquitetura de serviços diferenciados (DiffServ)

A arquitetura de serviços integrados, como vimos anteriormente, para além de muitas vantagens, possui algumas falhas como os *routers* determinarem a ordem de pacotes de acordo com uma lista muito grande de parâmetros (o que dificulta o desempenho do equipamento), os *routers* terem de manter a informação sobre o estado das reservas que estão feitas entre equipamentos (o que não permite a escalabilidade) com sinalizações constantes entre ambos os extremos da reserva, e suporta somente duas classes: "*controlled load*" e serviço garantido.

Para tentar contornar estes problemas criou-se um novo modelo de arquitetura à que se designou de arquitetura de **diferenciação de serviços** onde são celebrados contratos para que os fluxos de tráfego decada cliente é classificado como pertencendo a uma classe em particular, tratando a forma como algumas classes pedem a mesma qualidade de serviço. À entrada da rede, os pacotes, para isso, deverão ser marcados como pertencendo a uma classe em contrato e o escalonamento de pacotes é feito, depois, consoante essas marcas.

diferenciação de serviços

Em suma, o DiffServ (abreviatura vulgar de serviços diferenciados) permite que se implementem operações de encaminhamento no núcleo de uma rede e que se deixem operações mais complexas nos *routers* da fronteira da rede, definindo elementos funcionais que apenas terão de ativar o suporte de qualquer classe de serviço.

Este modelo terá de possuir, portanto, uma forma bastante precisa de classificar e condicionar na fronteira (*edge*) da rede, o que resulta na troca de pacotes com valores próprios de *Differentiated Services Code Point* (**DSCP**). Isto permite que seja possível efetuar escalabilidade.

DSCP

Portanto, o DiffServ terá de conseguir definir critérios de qualidade de serviço para responder a requisitos e garantias dadas a um agregado de tráfego, sendo que a aplicação de serviços provém de funções de condicionamento e de comportamentos realizados por cada salto dos pacotes entre *routers* (**PHB** - *Per-Hop Behavior*).

PHB

Antes de avançarmos mais convém, primeiro, identificarmos algumas terminologias próprias deste modelo arquitetural, que nos servirão muito no futuro:

- **Behevior Aggregate** (BA) - em português agregado de comportamento - é uma coleção de pacotes com o mesmo DSCP que cruza uma ligação numa determinada direção;
- **Per-Hop Behavior** (PHB) - em português comportamentos realizados por cada salto - é a propriedade que define a criação de filas em nós da rede que permite a aplicação de comportamentos de reencaminhamento de acordo com BAs especificados;
- **Mecanismo PHB** - é um algoritmo específico ou operação que é implementada num nó da rede para realizar um conjunto de um ou mais PHB.

O DiffServ permite um vasto **conjunto de ações**, descritas sumariamente abaixo:

conjunto de ações

- **classificação** - cada mecanismo de qualidade de serviço orientado à classe terá de suportar algum tipo de classificação (através de listas de acesso, *route maps*, ...);
- **medida** - alguns mecanismos de medida verificam a taxa de tráfego para reforçar uma certa política;
- **dropping** - alguns mecanismos são usados para descartar pacotes (como RED ou WRED);
- **policiamento** - alguns mecanismos são usado para reforçar limites de taxas às medidas efetuadas (tráfego em excesso é descartado);
- **shaping** - alguns mecanismos são usado para reforçar limites de taxas às medidas efetuadas (tráfego em excesso é atrasado);

- **marcas** - alguns mecanismos têm a capacidade de marcar pacotes com base na classificação e/ou medidas que foram feitas;
- **fila** - cada interface deverá ter um mecanismo de manipulação de filas;
- **reencaminhamento** - existem vários mecanismos de reencaminhamento suportados.

Estas ações podem (e são) organizadas numa máquina de estados, que poderá ser vista na Figura 8.6.

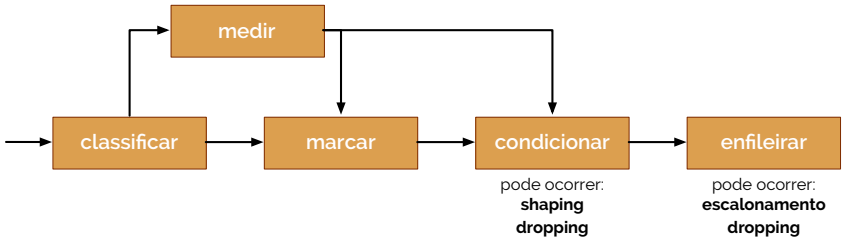


figura 8.6

O DiffServ possui um conjunto de **elementos funcionais**, como já tivemos a oportunidade de verificar: *routers* do núcleo e de fronteira. Os **routers fronteira** (*edge*) são os que permitem classificar pacotes e condicionar tráfegos (usando o modelo de *token bucket*). Os **routers do núcleo** permitem identificar tratamentos que deverão existir com os pacotes, dependendo da marca que transportam e de acordo com o PHB. Note-se que diferentes PHBs resultam em diferentes desempenhos da rede, pelo que os PHBs não especificam que mecanismos de manipulação de filas é que devem ser usados. O que acontece é que são definidos parâmetros que permitem estabelecer premissas como "uma determinada percentagem da largura de banda deverá ser atribuída aos pacotes da classe A durante um determinado intervalo de tempo, de um tamanho específico" ou "pacotes da classe A são servidos com o dobro da largura de banda de serviço que é atribuída à classe B".

elementos funcionais

routers fronteira

routers do núcleo

Tradicionalmente, o valor de DSCP por defeito é "000000", o que codifica o serviço de melhor esforço. A seleção de classes com o DSCP é feita através da alteração dos primeiros três bits (os mais significativos), de forma a podermos ter a retro-compatibilidade com o esquema de precedência do IP.

Existem entretanto dois (ou mais) esquemas de encaminhamento do PHB. O primeiro é o **Expedited Forwarding** (EF) que garante uma taxa de partida mínima, mas com certezas de largura de banda com encaminhamento prioritizado e que não permite as classes excedam a quantia da mesma largura de banda definida (caso contrário o tráfego será descartado)[14].

Expedited Forwarding

O segundo (ou restantes) esquema de encaminhamento dá pelo nome de **Assured Forwarding** (AF) e permite fornecer diferentes garantias de encaminhamento. Garantindo uma largura de banda definida, também permite extra largura de banda caso seja necessário, através de quatro classes AF1, AF2, AF3 e AF4[15]. Note-se que AFi é servido com maior largura de banda que AFj, onde assumimos que $i < j$.

Assured Forwarding

Integração de IntServ e DiffServ

Tendo em conta as especificações de ambas integração de serviços e diferenciação de serviços é possível **integrar** ambas entre um emissor e um recetor com requisitos de qualidade de serviço.

integrar

Enquanto que a arquitetura de IntServ é mais apropriada para redes pequenas, a arquitetura DiffServ é mais aconselhada para redes maiores. Assim sendo, será importante aplicar a primeira em redes de acesso (que usualmente são pequenas) e a segunda em redes

14 O valor recomendado para o DSCP com EF é "101110", o que permitirá que os IP's sem suporte de DSCP tenham precedência 5.

15 Os valores recomendados para o DSCP com AFa é "aaadd0" onde "aaa" é o valor binário da classe e "dd" é a probabilidade de dropping.

de trânsito. Para separar os vários tipos de redes podemos usar RSVP com pedidos em classes específicas de DiffServ, sendo que são rejeitados em caso de insuficiência de recursos.

Na Figura 8.7 podemos ver uma representação gráfica do pretendido.

figura 8.7

9. Encaminhamento Multicast

No capítulo de Endereçamento IPv4 e IPv6 referimos que existem várias **abstrações** na organização do espaço de endereçamento IP. Uma dessas abstrações foi a **multicast**, no qual conseguimos endereçar um conjunto de máquinas numa rede que possuem um ou mais artefactos em comum.

abstrações
multicast

Contrariamente ao que se possa pensar, em comparação às redes *unicast*, estas não são *connectionless*, isto é, requerem que se estabeleçam múltiplos caminhos entre *routers* para que estes possam saber como fazer o encaminhamento entre eles. Tendo tais caminhos realizados, o último passo é mesmo só a sinalização destas vias e a respetiva aplicação de rotas de encaminhamento - por via de protocolos.

Identificação de estações de destino

Em IPv6 já verificámos que para identificar uma máquina temos um conjunto de endereços já definidos para o fazer, sobre os quais, inclusive, podemos fazer os respetivos endereços MAC. Isto acontece através dos endereçamentos sobre MLD - mecanismo de controlo e contrato de endereços em *multicast* para IPv6.

Em termos de IPv4 também existe o suporte de IPv4. Enquanto que em IPv6 temos os endereços FF00::/16, em IPv4 temos os endereços que começam com "1110", denominados endereços de **classe D**. A estrutura de um endereço IPv4 está representada na Figura 9.1.

classe D

figura 9.1

Tal como em IPv6, aqui também nos é possível passar de um endereço IP para um endereço MAC. Tendo como prefixo para o endereço MAC "01:00:5E" e, no quarto byte o bit mais significativo a '0', os 23 bits menos significativos do endereço *multicast* passam diretamente para o endereço MAC, conforme podemos ver na Figura 9.2.

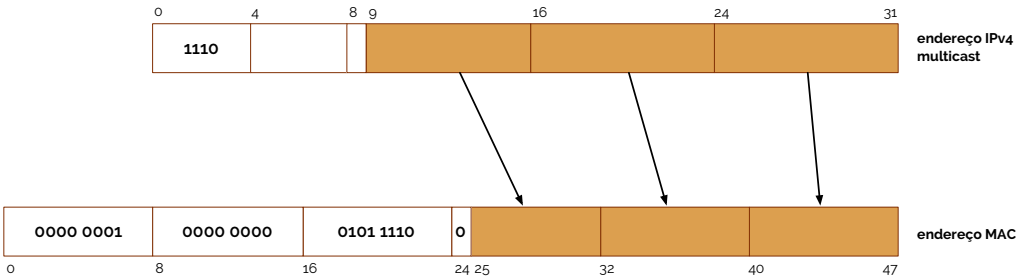


figura 9.2

Internet Group Membership Protocol (IGMP)

Da mesma forma que existe o MLD para IPv6, em IPv4 existe um protocolo para o contrato de endereços *multicast* denominado **IGMP** (sigla de *Internet Group Membership*

IGMP

Protocol). Atualmente na versão 2 definida no RFC 2236, este opera sobre a estação e os *routers* que lhe estão diretamente ligados, servindo para que a estação possa anunciar ao *router* que quer participar numa sessão *multicast*. Este pacote tem a estrutura da Figura 9.3, onde MRT significa *Maximum Response Time*, e que corre sobre IP, com número de protocolo 2, e enviando estes pacotes para endereços *multicast* em 224.0.0.1, referente a todos os *equipamentos*, com TTL de 1.

RFC 2236

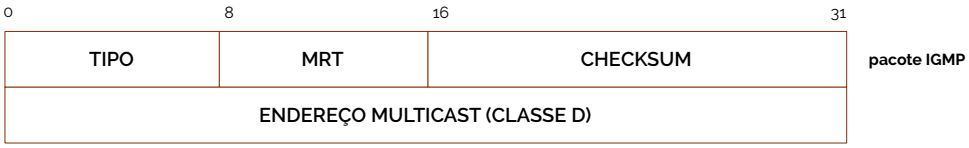


figura 9.3

O protocolo IGMP possui quatro **tipos de mensagens**, entre as quais um **General Membership Query** (GMQ) que é enviado pelos *routers* para as estações a questionar se pretendem participar numa sessão *multicast*, um **Specific Membership Query** (SMQ) enviado pelos *routers* a questionar se alguma estação pretende iniciar uma sessão *multicast*, um **Membership Report** (MR) enviado pelas estações para sinalizar que participam numa sessão *mulicast* e um **Leave Group Report** (LGR) que, embora seja opcional, é enviada pelas estações para sinalizar que pretendem terminar a sessão (é opcional porque pode ser usado o MR). Em cada rede o *router* que recebe estes pedidos, denominado de **Querier Router** (QR) é o que tem IP mais baixo em todas as interfaces ligadas a uma rede, sendo também o que mantém as mensagens IGMP com os terminais.

tipos de mensagens
General Membership Query
Specific Membership Query
Memberhip Report
Leave Group Report
Querier Router

A **troca de mensagens** efetua-se da seguinte forma. Primeiro os *routers* enviam periodicamente GMQs onde especificam um MRT, como podemos ver na Figura 9.4.

troca de mensagens

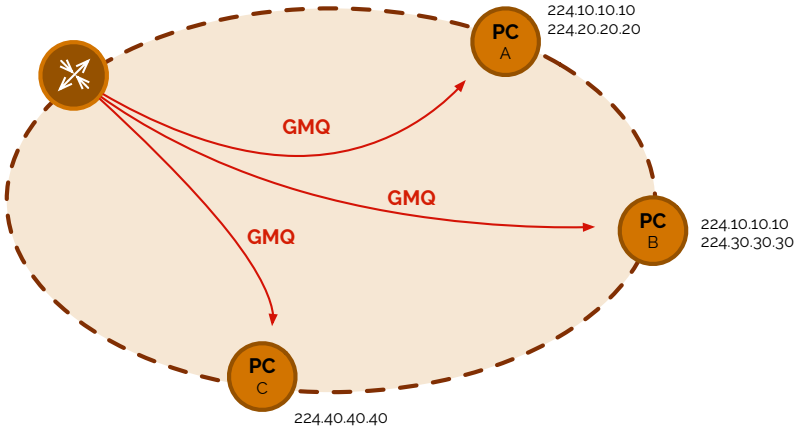


figura 9.4

Passado o ponto da Figura 9.4, então os terminais enviam MRs com endereço *multicast*, depois de esperarem um tempo aleatório entre 0 e MRT segundos (Figura 9.5).

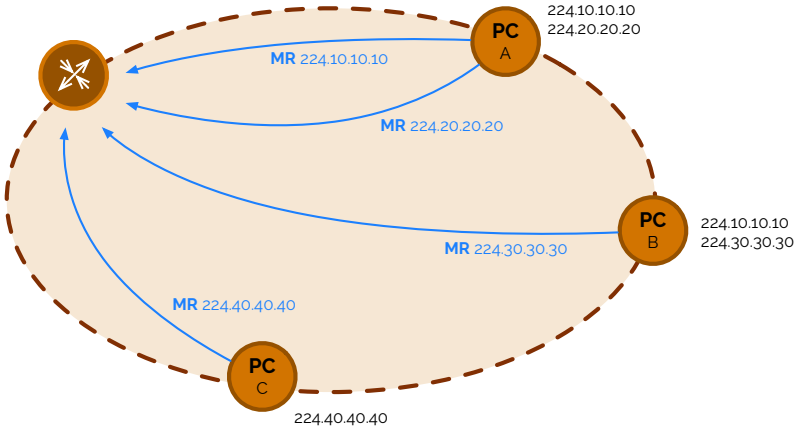
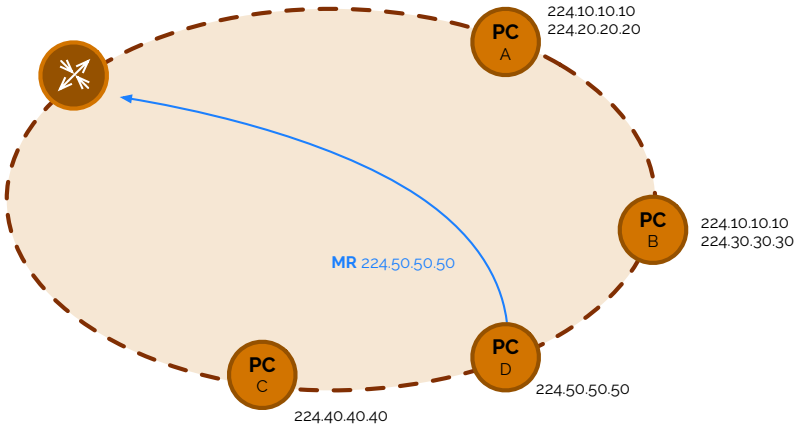


figura 9.5

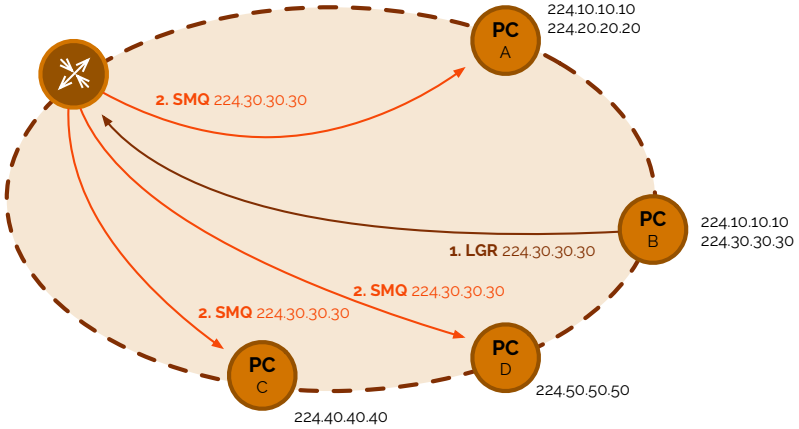
Por outro lado, também pode ser uma estação que entra na rede que pode enviar um MR diretamente para entrar numa sessão *multicast*, como podemos ver na Figura 9.6.

figura 9.6



Quando um terminal assim pretender, poderá enviar um pedido de saída da sessão com um LGR, ao qual se responde com um SMQ para verificar se ainda há estações que pretendem manter a sessão, como podemos ver na Figura 9.7.

figura 9.7



nota

nota!! nas figuras acima a comunicação está representada sempre de terminais para *routers*, mas, no entanto, estas são feitas para todos os equipamentos que estão no mesmo espaço *multicast*. Foi uma mera escolha de representação para limitar a entropia das figuras.

Encaminhamento em multicast

encaminhamento

O **encaminhamento** em *multicast* pode ser feito de uma de duas formas: tendo em conta uma *group-shared tree* ou uma *source-based tree*.

group-shared tree

Numa **group-shared tree** tenta-se determinar uma árvore de encaminhamento por sessão de *multicast* que conecta todos os *routers* com estações pertencentes a esta. Usando abrangência mínima por Steiner (árvores de Steiner) acaba por não ser muito usado na prática porque requer grande poder computacional, dada a sua complexidade, e informação sobre toda a rede. Pior, esta técnica tem de ser executada por cada vez que um *router* necessita de entrar ou sair de uma sessão.

⊙ Jakob Steiner

source-based tree

Por estas razões é que se prefere a solução de **source-based tree**. Aqui tentamos determinar a árvore de encaminhamento, por sessão *multicast* e por emissor, sendo que existe um *router* que permanece identificado como **ponto central**.

ponto central

Consideremos assim o seguinte cenário da Figura 9.8 [14], onde o emissor é co nhecido e está conectado a A.

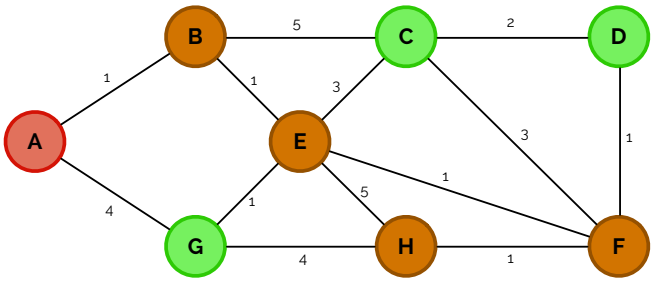


figura 9.8

Na Figura 9.8, cada nó que se encontra a verde tem recetores interessados numa sessão S de um emissor com endereço E envia um **join** para o endereço de E ligado a uma sessão S, como podemos ver na Figura 9.9, pela ligação de menor custo.

join

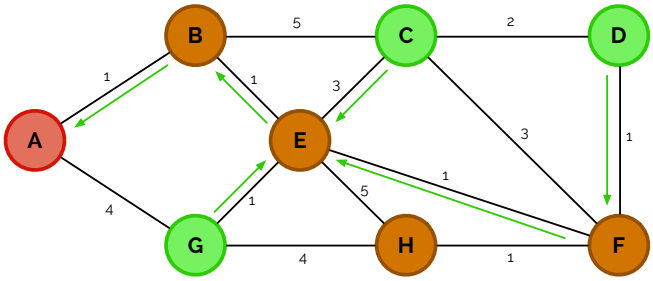


figura 9.9

No caminho da mensagem *join*, cada nó recebe a mensagem numa dada interface e reenvia-a através de outra criando uma entrada na tabela de encaminhamento com essa informação, sendo que o encaminhamento *multicast* não só é baseado no endereço de destino, como também é no endereço de origem dos seus pacotes.

Quando existem múltiplos emissores, então várias árvores de encaminhamento são estabelecidas, uma por cada emissor. se um recetor não estiver interessado numa determinada sessão *multicast*, então o seu nó conectado deverá enviar uma mensagem de **prune** (em português poda) do endereço de E para a sessão S para o endereço do emissor, como podemos ver na Figura 9.10, onde D desistiu da sessão.

prune

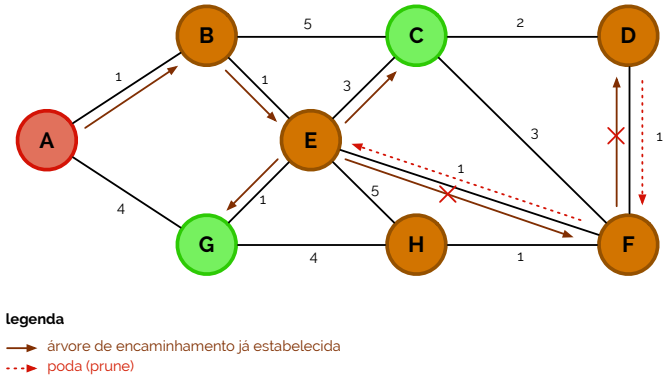


figura 9.10

A mensgem de *prune* é reenviada pelos nós que já não pertencem à árvore de encaminhamento e a rota é eliminada das suas tabelas de encaminhamento.

Agora consideremos um novo cenário, semelhante ao anterior, mas em que não se sabe quem é o nó emissor. Para resolvermos este problema teremos que criar uma árvore de abrangência virtual através do mecanismo de **Reverse Path Forwarding**, onde cada nó N encaminha os pacotes de uma origem O, que são recebidos dos vizinhos V, para todos os seus vizinhos somente se V for o último nó com custo mínimo de O para N. Vejamos a Figura 9.11, onde no nó E, por exemplo, os pacotes originados do nó A são encaminhados para todos os outros portos somente e eles vierem do nó B, isto porque este é o nó anterior de E no caminho de menor custo de A até E.

Reverse Path Forwarding

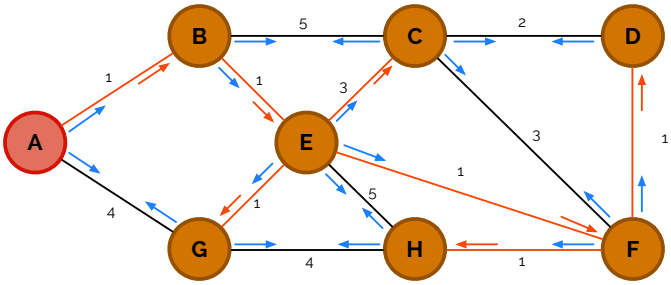


figura 9.11

Se ao RPF aplicarmos o *pruning* também temos que para cada nó de origem O, o nó N sabe para que vizinhos V é o último nó no caminho de custo mínimo de O para V, pelo que o encaminhamento será feito através destes vizinhos. Veja-se assim na Figura 9.12, por exemplo, que o nó E só encaminha pacotes da origem A para os vizinhos C, F e G, isto porque o nó E é o último nó no caminho de custo mínimo de A para qualquer um dos seus vizinhos.

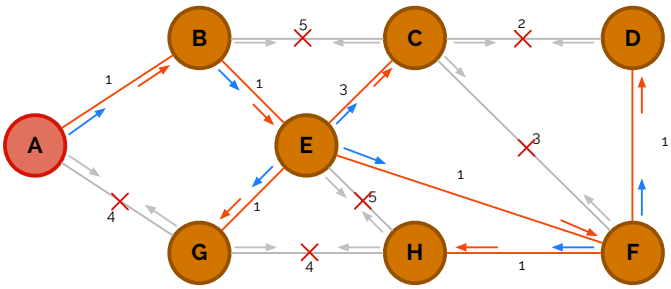


figura 9.12

Em rigor, as figuras acima necessitam de identificar que terminais é que estão interessados nas sessões *multicast*, uma vez que se não estiverem, então não vale a pena terem sequer ligação, sendo realizado o *pruning*. Este caso pode ser visto na Figura 9.13, onde os nós interessados continuam a extender a árvore e o nó H, não interessado sofre uma poda na sua ligação com F.

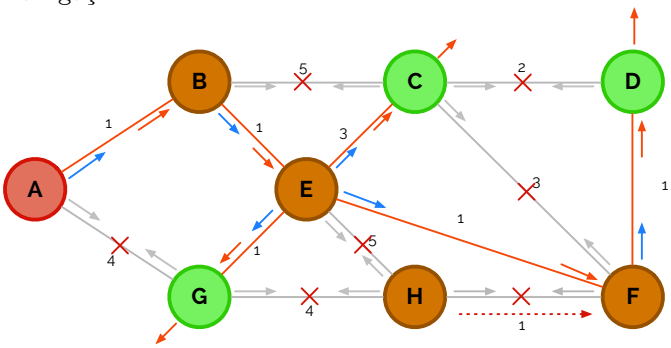


figura 9.13

Consideremos agora que D já não pretende receber mais pacotes *multicast*. Então este terá de enviar um *prune* para F que, por sua vez, por não estar interessado também, envia um *prune* para E, como podemos ver na Figura 9.14.

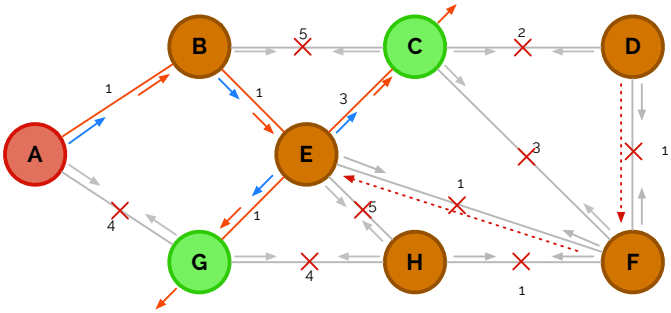


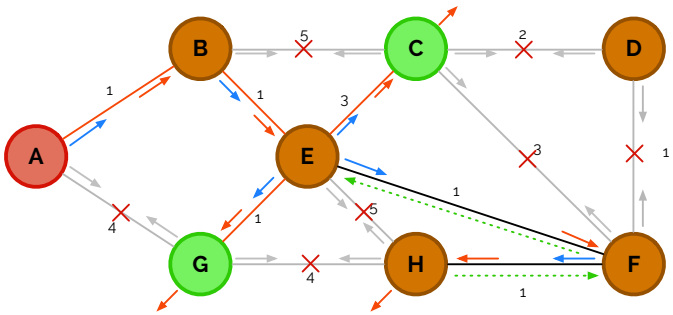
figura 9.14

O evento da Figura 9.14 acontece porque um nó sem nenhum terminal interessado na sessão *multicast* (e sem nenhum vizinho igualmente interessado) envia um *prune* para o vizinho de onde recebe pacotes *multicast*.

Se porventura agora H pretender juntar-se novamente à sessão, então existem duas opções possíveis para o fazer: ou se associa um tempo de vida às mensagens *prune* sobre a qual a ligação permanece cortada, mas no fim volta a encaminhar pacotes; ou se manda uma mensagem de **graft** (enxerto), para anular o *prune*, como podemos ver na Figura 9.15.

graft

figura 9.15



Protocolos de encaminhamento em multicast

Da mesma forma que o encaminhamento *unicast* é garantido por um conjunto muito vasto de protocolos, o encaminhamento *multicast* também o é, tanto por protocolos *distance-vector*, como protocolos *link-state*.

Abordando primeiramente protocolos *distance-vector* temos que falar do *Distance Vector Multicast Routing Protocol* (**DVMRP**) cujo comportamento é muito semelhante ao RIP. Estando designado no RFC 1075, usa uma estratégia de RPF com *pruning*, sendo que a distância é dada pelo número de saltos.

DVMRP

↪ **RFC 1075**

Já do lado dos protocolos *link-state* podemos referir o *Multicast Open Shortest Path First* (**MOSPF**), que é uma extensão do OSPF defendida no RFC 1584. Implementando uma estratégia igual de RPF com *pruning* tira usufruto do conhecimento topológico de cada *router*, sendo que cada um destes possui uma base de dados local de grupos *multicast* que contêm uma lista de membros destes diretamente conectados.

MOSPF

↪ **RFC 1584**

Independente do protocolo, existe um protocolo que permite o encaminhamento em *multicast* denominado **PIM** (acrónimo de *Protocol-Independent Multicast*). Este mecanismo possui duas formas de execução - uma esparsa e uma densa - que são escolhidas consoante o número de máquinas ligadas que pretendem entrar numa ou mais sessões *multicast*.

PIM

↪ **RFC 2362**

No **modo denso** do PIM é implementada uma estratégia de RPF com *pruning* que requer que todos os *routers* tenham o protocolo PIM ativo. Este modo de execução é mais simples que o esparso porque não necessita de recalcular as tabelas de encaminhamento, antes usa os mesmos mecanismos dos protocolos *unicast* para as criar, assumindo que todas as ligações ponto-a-ponto (e os seus encaminhamentos) são simétricas.

modo denso

Aqui, para iniciar o processo é feito um **flooding** inicial em que, quando um *router* recebe tráfego *multicast* numa interface que forneça custo mínimo para a origem (interface RPF), reencaminha o tráfego para todas as outras interfaces e, quando um *router* recebe tráfego *multicast* na interface que não fornece custo mínimo para a origem (não a interface RPF), descarta os pacotes.

flooding

Depois do *flooding*, inicia-se uma fase em que são feitas algumas podas através das mensagens de *pruning*, de forma a gerar uma árvore abrangente. Caso algum terminal/estação pretenda iniciar uma sessão *multicast*, então envia uma mensagem de *graft* para fazer um pequeno enxerto na rede e nos devidos encaminhamentos de pacotes *multicast*.

Quando existe mais do que um *router* a enviar tráfego de uma sessão *multicast* específica para um meio partilhado como uma LAN, então há que ser feita uma decisão sobre que *router* é que deve permanecer a enviar pacotes. Para resolver esta situação é enviada uma mensagem de **assert** para verficar a situação e resolver o problema.

assert

modo esparso

O **modo esparso** do PIM é usado preferencialmente quando as estações que pretendem usar *multicast* estão muito concentradas num pequeno número de redes. Enquanto que o modo denso é um protocolo *data-driven* (dado que requer que os *routers* que não tenham clientes ativos enviem mensagens *prune* recorrentemente), o modo esparso é um protocolo *receiver-driven*, isto porque cada *router* anuncia explicitamente (com mensagens *join*) que pretende juntar-se a sessões *multicast* específicas.

Este protocolo usado desta forma é preferido ser aplicado em redes mais pequenas porque usa, inicialmente, uma estratégia de *group-shared tree*. No entanto, passado um tempo após a ativação do protocolo, este muda de estratégia para *source-based tree*. Isto acontece quando é feito o primeiro registo PIM para o ponto de *rendezvous*, onde é enviado uma mensagem *join* e iniciada a criação da *source-based tree*.

E assim terminam os apontamentos para a disciplina de Arquitetura de Redes (a3s2) cuja informação é complemento para a disciplina de Fundamentos de Redes (a3s1) e que é continuada em cadeiras como Arquitetura de Redes Avançada (a4s1) ou Segurança (a4s1).

71 ARQUITETURA DE REDES

As referências abaixo correspondem às várias citações (quer diretas, indiretas ou de citação) presentes ao longo destes apontamentos. Tais referências encontram-se dispostas segundo a norma IEEE (as páginas Web estão dispostas de forma análoga à de referências para livros segundo a mesma norma).

[1] P. Salvador and A. Nogueira, "Enterprise Network Design Topics," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[2] "Gigabit Campus Network Design— Principles and Architecture," San Jose, CA, 1999.

[3] R. Lopes, Apontamentos de Fundamentos de Redes, 2 ed. Aveiro, 2017.

[4] P. Salvador and A. Nogueira, "IPv4 & IPv6 Addressing," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[5] P. Salvador and A. Nogueira, "Internal Routing," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[6] LabN, Adva, Alcatel-Lucent, and A. Productions, "RFC 5250: The OSPF Opaque LSA Option," ed, 2008.

[7] P. Salvador and A. Nogueira, "Wireless Networks," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[8] L. Edwards and C. Waelde, *Law and the Internet*: Bloomsbury Publishing, 2009.

[9] P. Salvador and A. Nogueira, "DHCP and NAT in complex networks," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[10] P. Salvador and A. Nogueira, "Security Topics," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[11] P. Salvador and A. Nogueira, "Network Management," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[12] IBM. (June 2017). *Protocol data units (PDUs)*. Available: https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.12/gtpc1/pdus.html

[13] P. Salvador and A. Nogueira, "QoS: Layer 2, Layer 3 IntServ and DiffServ," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[14] P. Salvador and A. Nogueira, "Multicast Routing," in *Arquitetura de Redes*, ed. Aveiro: Universidade de Aveiro, 2017.

[15] J. Kurose and K. Ross, Computer Networking: A Top-Down Approach, Global Edition: Pearson Education Limited, 2017.

Apontamentos de Arquitetura de Redes
1ª edição - junho de 2017

ar

Autor: Rui Lopes
Agradecimentos: professores Paulo Salvador e Susana Sargento

Todas as ilustrações gráficas são obra de Rui Lopes e as imagens são provenientes das fontes bibliográficas divulgadas.

apontamentos