Administração de Redes 2023/24

Equipamento nos laboratórios

Breve introdução ao Cisco IOS

Hardware de rede

Ferramentas de configuração e teste

Ficheiros de configuração

Captura e análise de pacotes

Equipamentos no laboratório

Postos de trabalho

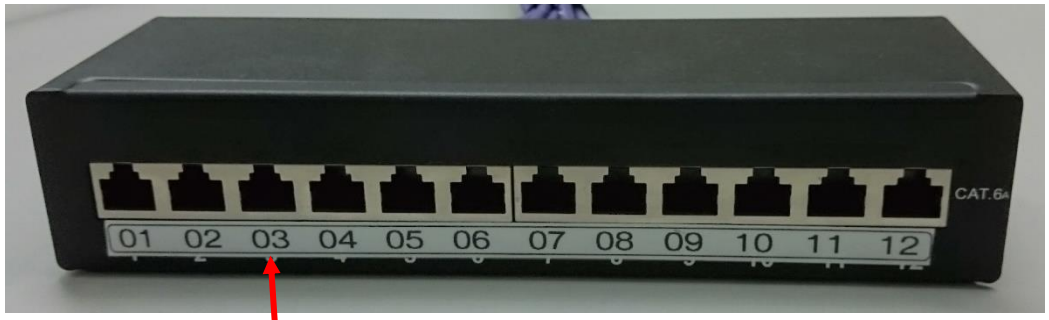
- 12 postos de trabalho (PCs) para 12 grupos em cada turma
  - 2 pessoas por grupo
  - Apenas para realizar os trabalhos (não há entregas)
- 2 partições de arranque (uma para cada turma)
  - Verificar que arrancam sempre com a partição da turma correcta
- Utilizador `ar` já criado, a password é `admredes23`
  - A password de `root` é a mesma
  - Mudar ambas na primeira utilização

Bastidores

Bastidor 2

Bastidor 3

Interligação ao Bastidor 2

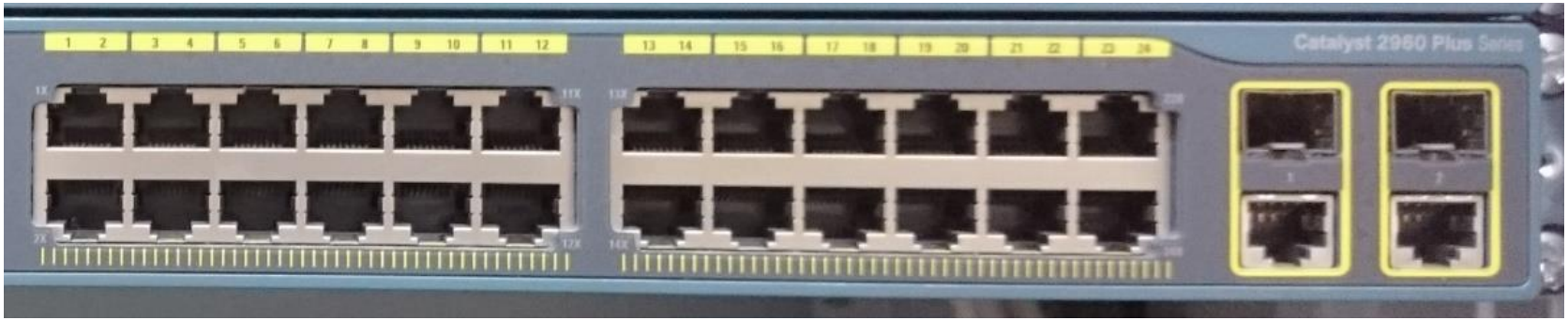


Desktop patch panels

da porta no bastidor indicado no desktop patch panel

Patch panel no bastidor 2

Switch Cisco 2960



Portas Fast Ethernet (RJ45)

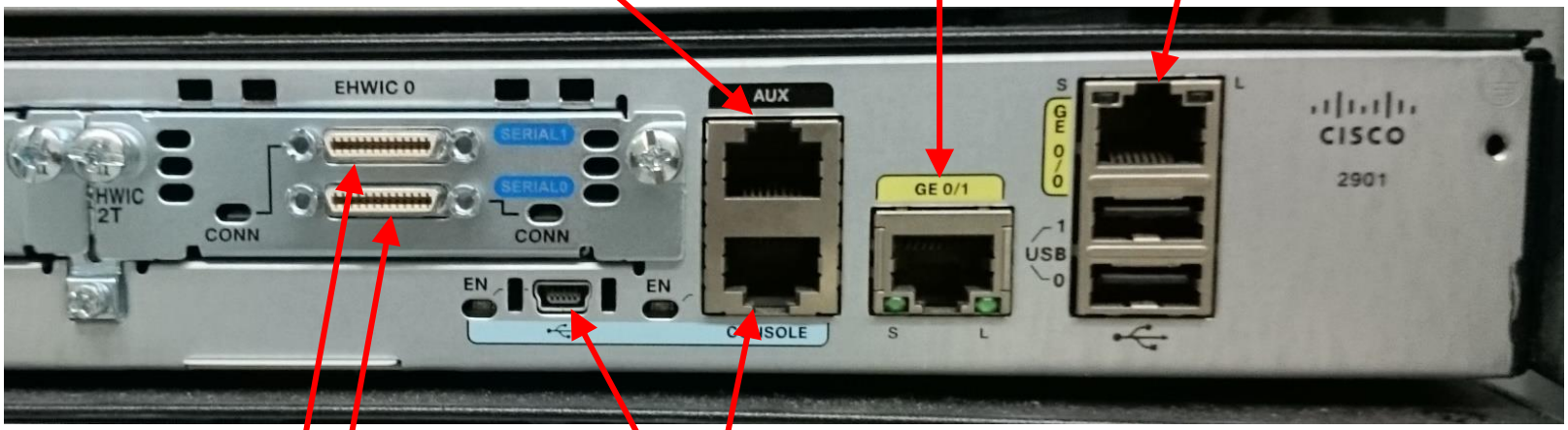
Portas Gigabit Ethernet (SFP ou RJ45) para *uplink*

Transceivers SFP para fibra óptica e cobre (indisponíveis no laboratório)

Router Cisco 2901

Porta AUX (RS232)

Portas Gigabit Ethernet



Portas série síncronas
(para interligação)

Portas de consola série, USB e RS232
(para configuração através do terminal)

Router Cisco 881W

Router Cisco 2691

Router Cisco 1841

Interfaces série síncronas

- *Os routers* podem interligar-se através de interfaces série síncronas
  - Diferentes das RS-232 usadas para Consola e AUX
- Os cabos têm uma ponta DCE (Data Communication Equipment) e outra DTE (Data Terminal Equipment)
  - A forma como é ligado determina que router funciona como DTE e como DCE
  - No router a funcionar como DCE (e apenas neste) é preciso ir à interface e configurar o *clock rate*
- Nos cabos usados para interligar portas *Smart Serial*
  - O DCE é a ponta com a etiqueta "72-1429-03"
  - O DTE é a ponta com a etiqueta "72-1428-03"

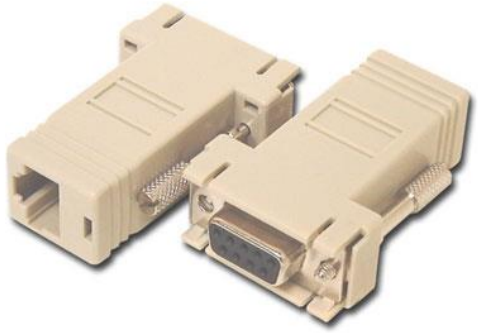
Cabos para interfaces série síncronas

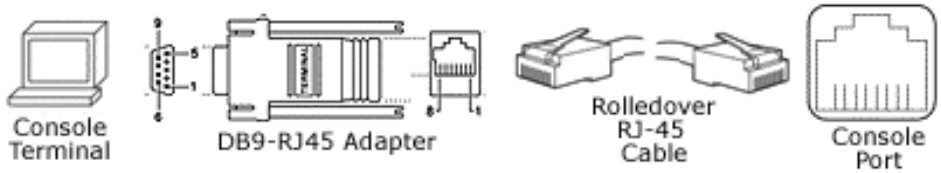
Cabo para interligar portas *smart serial* (2901, 2691)

Consola

- Para configurar e administrar os *routers* usa-se um emulador de terminal no PC (minicom), série (RS-232) ligada à porta de Consola do *router* usando este cabo: →

- Em alternativa, pode usar-se um cabo *rollover* e um adaptador:





Configuração por consola série dos equipamentos nos bastidores

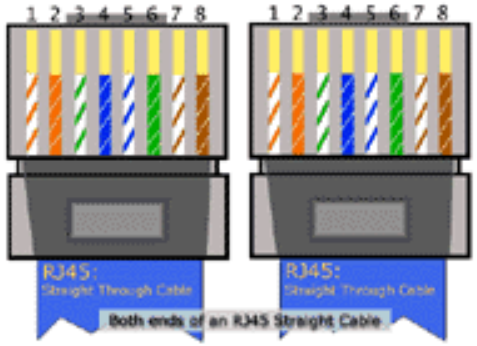
- Feita através de um servidor de consolas série
- Para aceder, usar `telnet consoles` *porta*

Mapeamento entre portas e equipamentos:

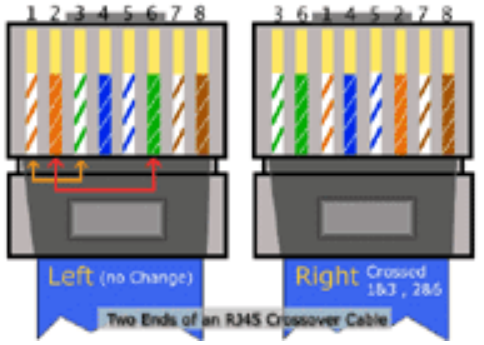
Porta 7001: RTSEC-LRB01-01 (Bastidor 2)
Porta 7002: RT-LRB01-01 (Bastidor 2)
Porta 7003: SW-LRB01-01 (Bastidor 2)
Porta 7004: SW-LRB01-02 (Bastidor 2)
Porta 7005: RT-LRB02-01 (Bastidor 2)
Porta 7006: RT-LRB02-02 (Bastidor 2)
Porta 7007: SW-LRB02-01 (Bastidor 2)
Porta 7008: SW-LRB02-02 (Bastidor 2)
Porta 7009: SW-LRB03-01 (Bastidor 3)
Porta 7010: SW-LRB03-02 (Bastidor 3)
Porta 7011: RT-LRB03-01 (Bastidor 3)
Porta 7012: RTSEC-LRB03-01 (Bastidor 3)

Nunca desligar os cabos das portas de consola série!

Diferentes tipos de cabo com conectores 8P8C (RJ45)

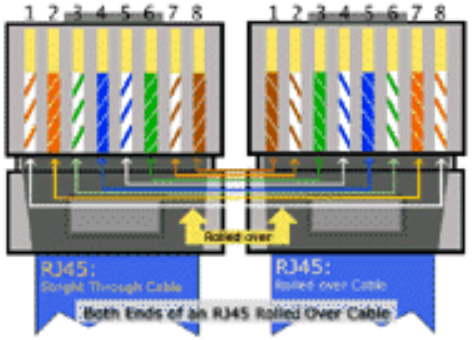


Directo (o mais comum). Para ligar PC ou *router* a concentrador ou comutador. Pode usar-se para interligar dois PCs se pelo menos um deles tiver interface Auto MDI-X.



Cruzado (pares TX e RX trocados numa das pontas). Para ligar directamente dois PCs (ou um PC e um *router*).

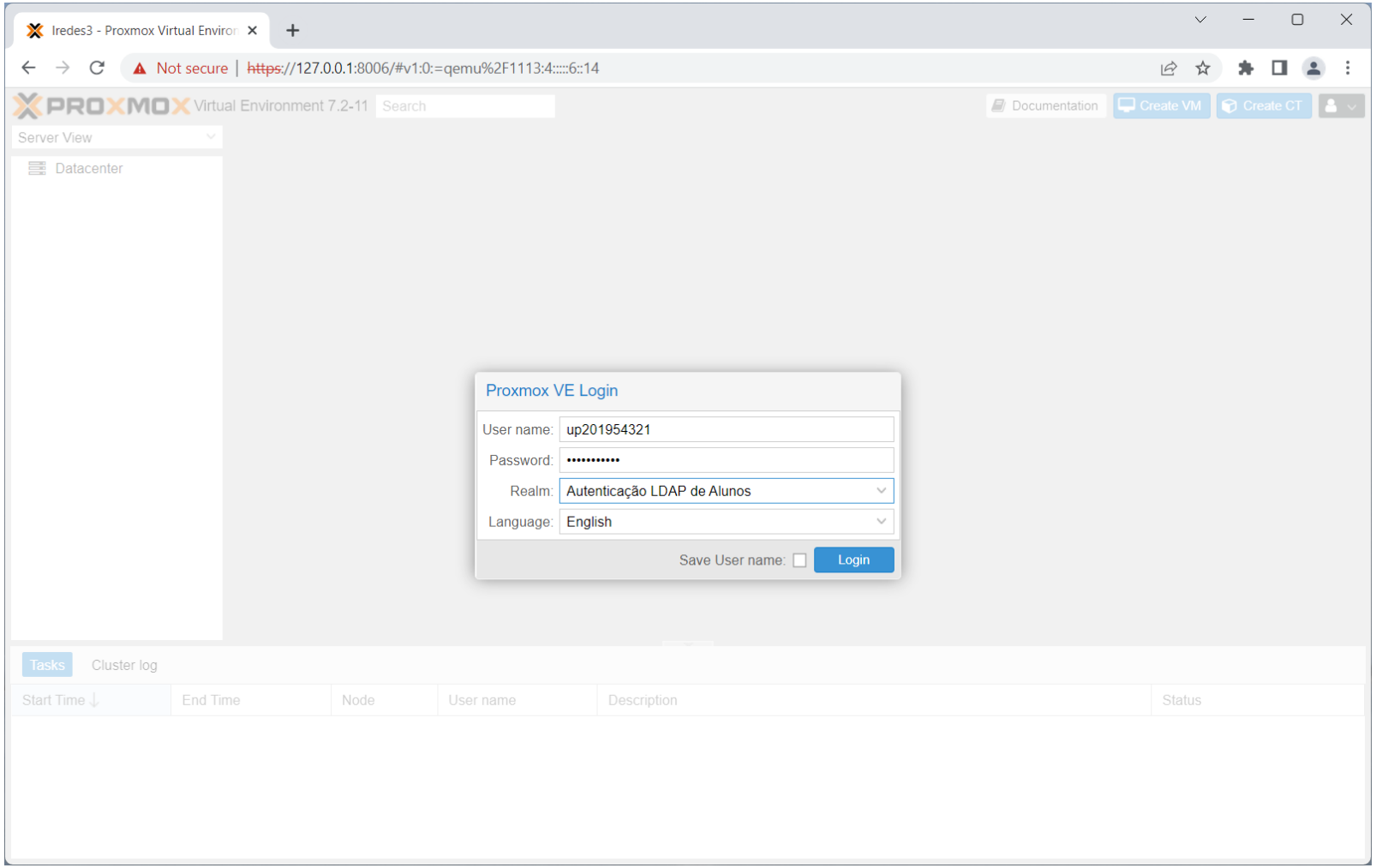
Rollover. Para ligar à Consola dos *routers*. Não são de rede!
NOTA: Os servidores de consolas série usam cabos directos.



Infraestrutura virtualizada

- Equipamentos físicos usados na 1ª aula
  - Nos trabalhos, os PCs servirão apenas como terminais gráficos para aceder a uma infraestrutura virtualizada
- 1 máquina virtual por grupo com nome tx-gy, onde *x* é a turma e *y* é o grupo nessa turma
  - Corre o GNS3 e virtualizações aninhadas
- Cluster com 3 servidores de máquinas virtuais
- Ligar e desligar a máquina usando o Proxmox
  - https://lredes1.alunos.dcc.fc.up.pt:8006
  - Seleccionar "Autenticação LDAP de Alunos"
  - Entrar com as credenciais UPorto

Autenticação no Proxmox



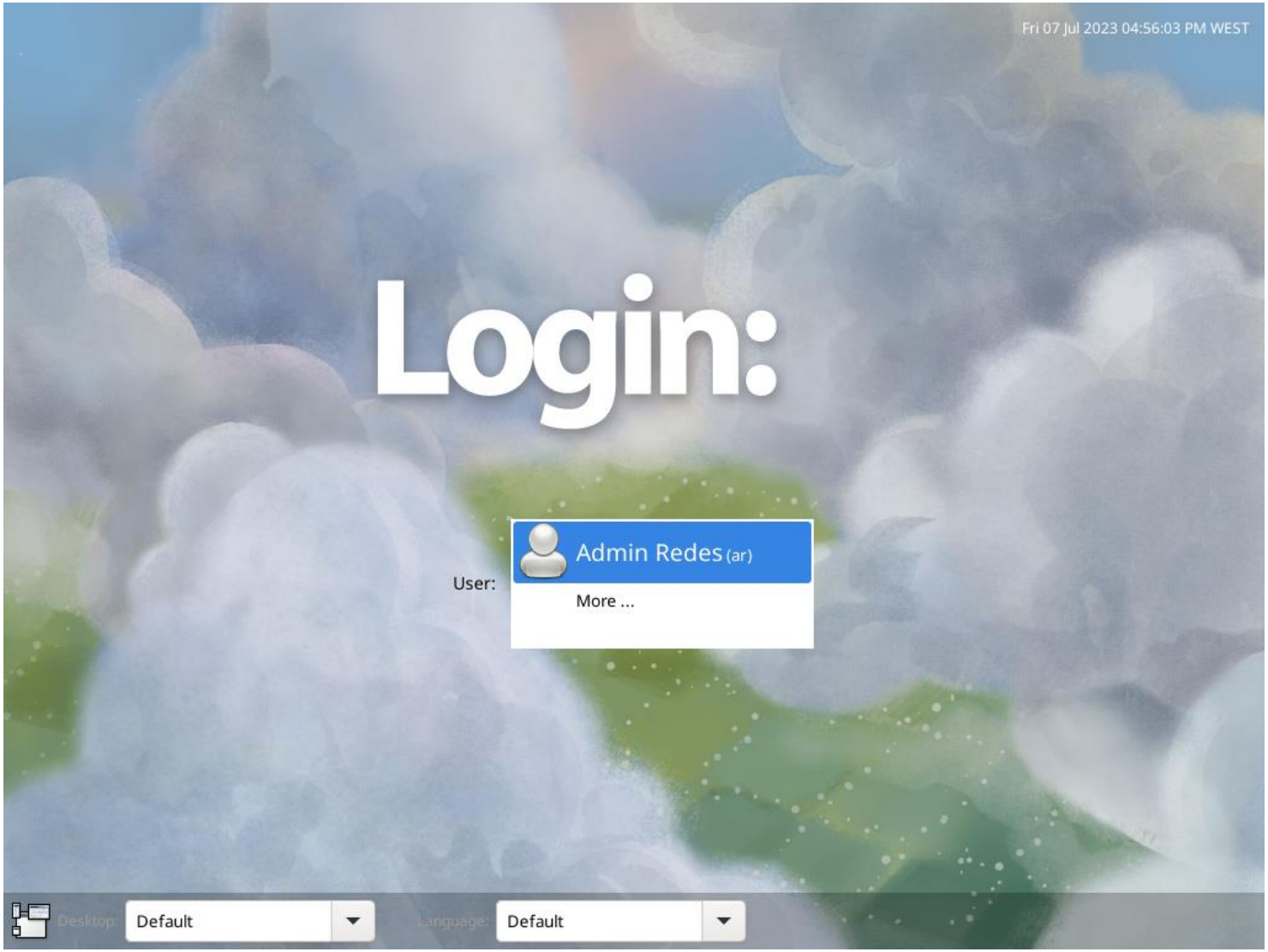
Utilização das máquinas virtuais

- Máquinas virtuais são acessíveis a partir de qualquer laboratório
  - E também do exterior usando ssh port forwarding
- Cada grupo é responsável pela gestão da sua VM
- Durante as aulas de laboratório, apenas os grupos da turma em aula podem ter a VM a correr
  - Outras que estejam a correr serão desligadas
- Fora do tempo de aula, ter a VM ligada apenas quando estão a fazer os trabalhos
  - Evitar sobrecarregar os servidores, prejudicando quem precisa de trabalhar
  - Podem desligar a VM ou suspendê-la para depois continuar

Utilização das máquinas virtuais

- Aceder à VM usando Consola SPICE
  - Abrir o ficheiro com extensão `.vv` no `remote-viewer`
- Credenciais
  - Username: `ar`
  - Password: `admredes23`
  - VM principal e outras que se criem dentro

Máquina virtual a correr

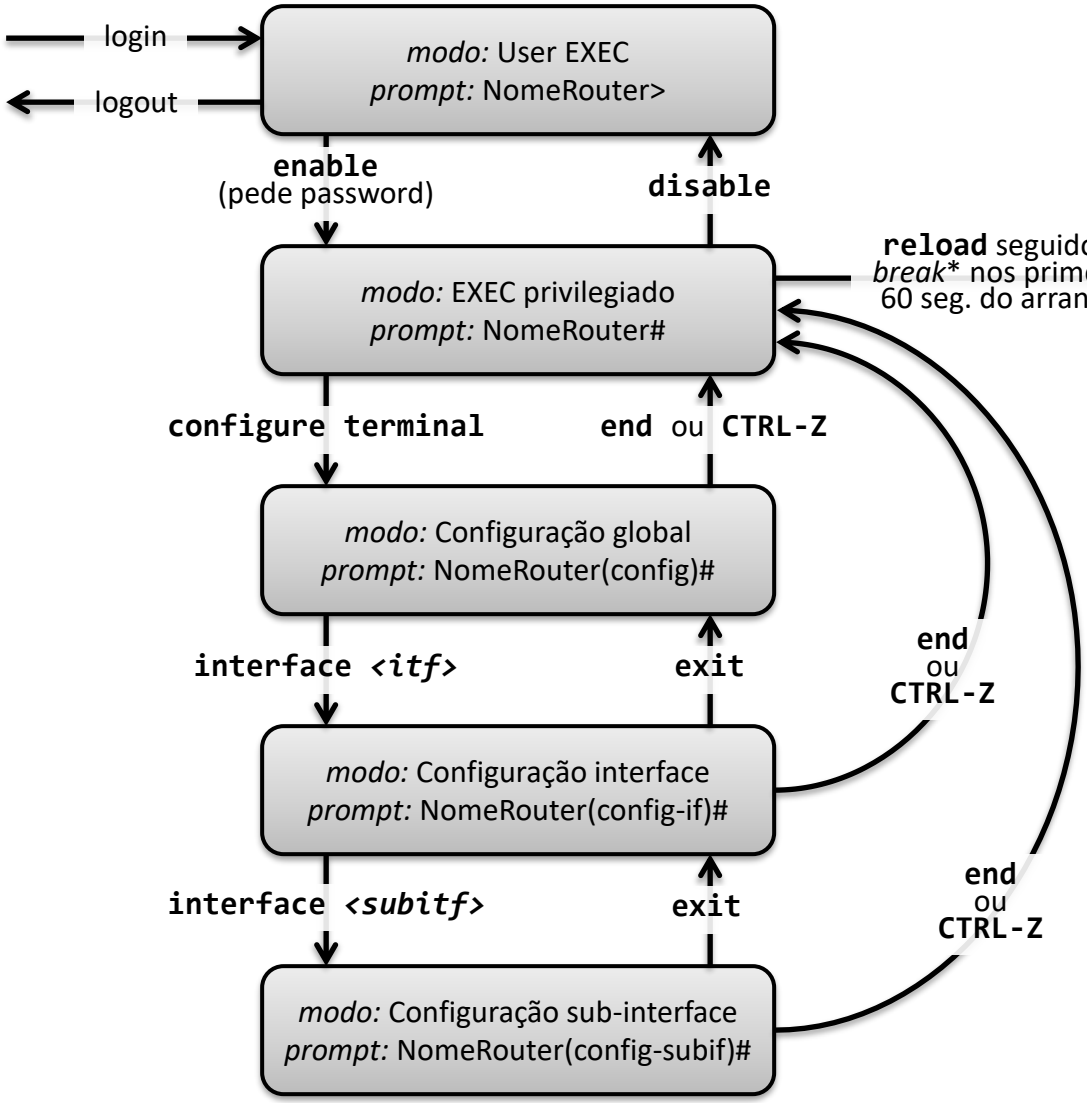


Breve introdução ao Cisco IOS

Cisco IOS

- O IOS (Internetworking Operating System) é um sistema operativo proprietário da Cisco Systems
  - Corre nos equipamentos deste fabricante
- Configurável por linha de comando
  - (Emulador de) terminal ligado à porta série (cabo *rollover*)
  - Telnet / SSH
- Interface de linha de comando
  - Sintaxe própria
  - Modal

Linha de comando Cisco IOS — Modos



NOTAS:

1. No minicom usa-se **CTRL-a f** para enviar um *break*
2. Sempre que não sejam ambíguos, os comandos podem ser abreviados
3. Pode usar-se **TAB** para auto-completação
4. Pode usar-se **?** para obter ajuda

Cisco IOS — Configurar interface

```
R1>
R1>enable
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface FastEthernet 0/0
R1(config-if)#ip address 172.16.0.2 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#
*Feb 22 19:20:20.483: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
R1(config-if)#
*Feb 22 19:20:20.483: %ENTITY_ALARM-6-INFO: CLEAR INFO Fa0/0 Physical Port Administrative
State Down
*Feb 22 19:20:21.483: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0,
changed state to up
R1(config-if)#end
R1#
*Feb 22 19:20:29.715: %SYS-5-CONFIG_I: Configured from console by console
R1#ping 172.16.0.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.1, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 40/96/172 ms
R1#
```

Cisco IOS – Guardar configuração

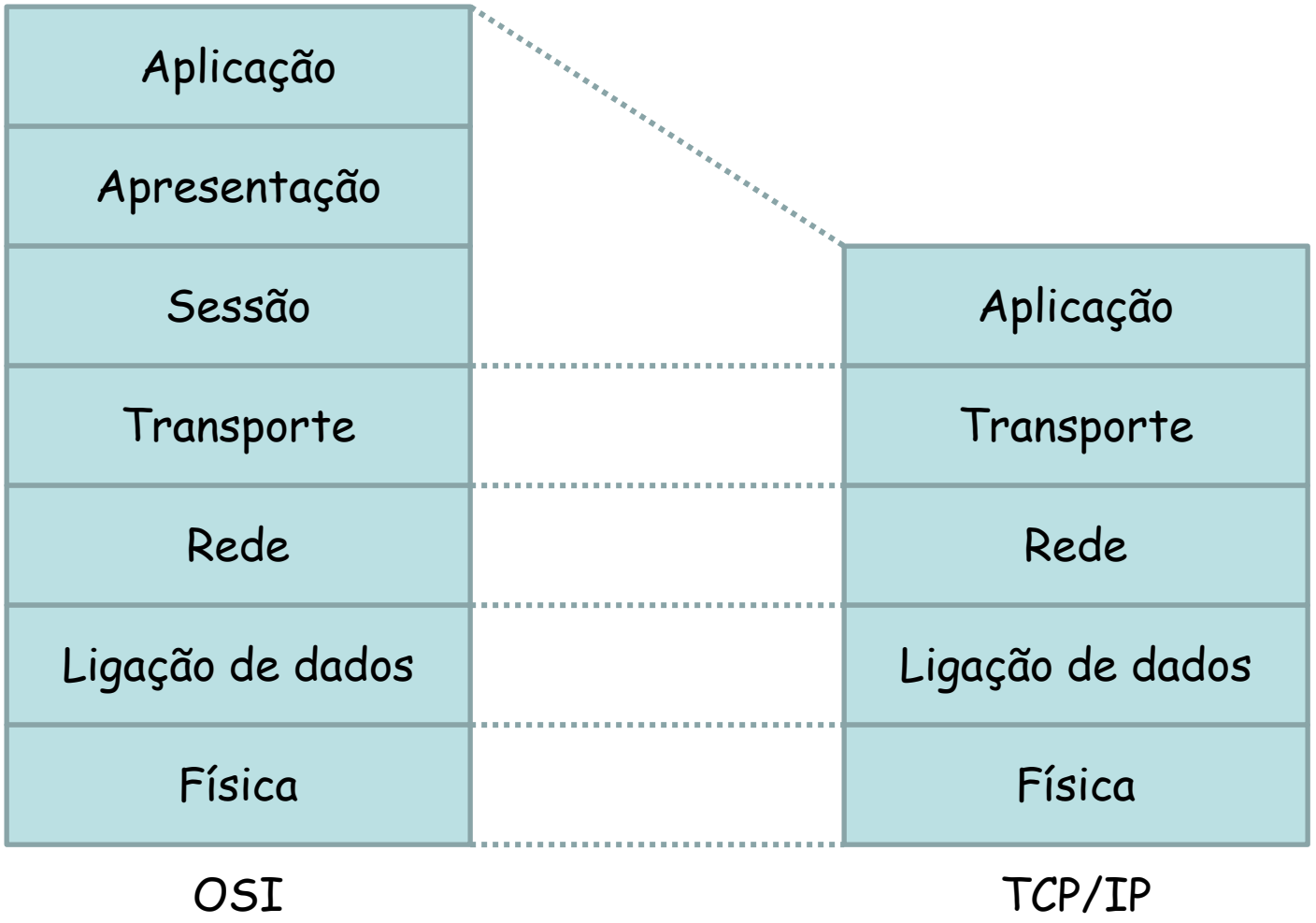
```
R1#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
R1#
```

Algumas dicas úteis

- O carácter '?' permite obter ajuda contextual
  - Lista de comandos
  - Completação possível
  - Parâmetros seguintes de um comando
- O carácter Tab faz auto-completação
- É possível abreviar comandos sempre que não haja ambiguidade
  - E.g., `copy running-config startup-config` pode ser abreviado para `copy run st`
- Para desfazer o que um comando faz, usar o prefixo "no"
  - E.g., `no ip route 10.0.0.0 255.0.0.0`
- Alguns comandos também suportam o prefixo "default" para usar um valor-padrão
  - E.g., `default clock rate`

And now for something completely different…

Pilha protocolar

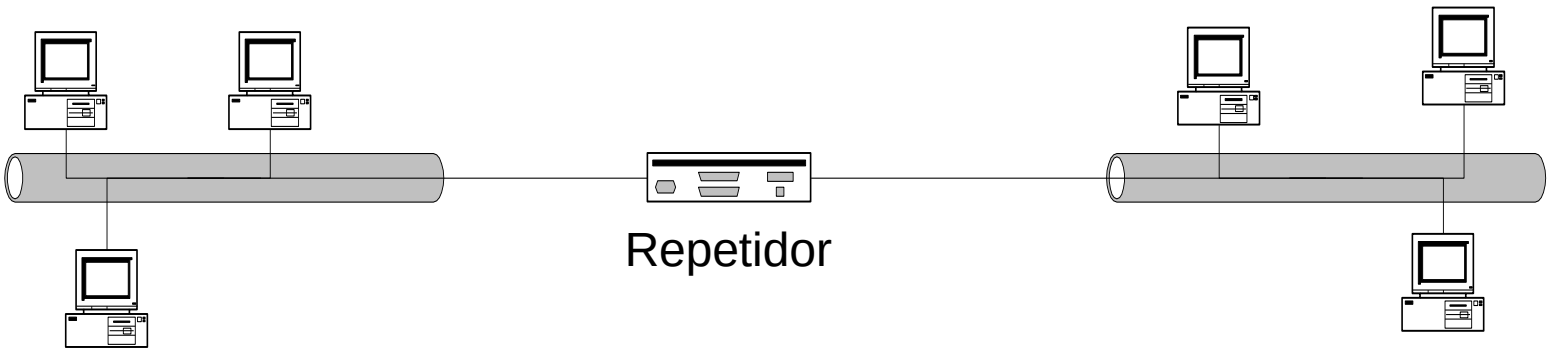


Dispositivos numa rede

- *Router*
  - Opera na camada de rede
  - Trabalha com endereços IP
- Comutador *(switch), bridge*
  - Opera na camada de ligação de dados
  - Faz filtragem de tramas com base em endereços MAC
- Concentrador *(hub)* — obsoleto
  - Opera na camada física
  - Repetidor estúpido (não faz filtragem)

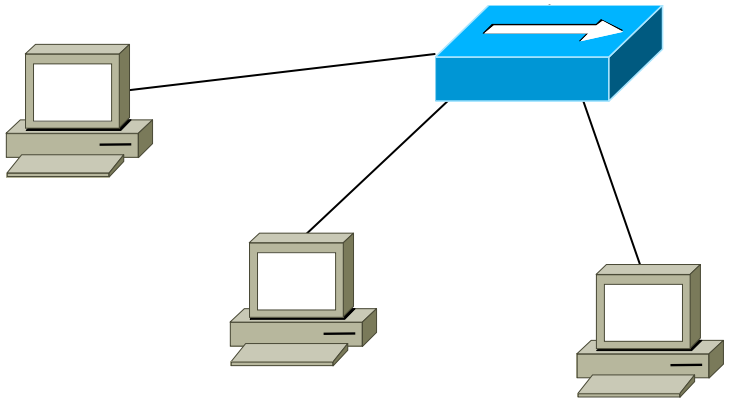
Repetidor

- Camada Física
- Interliga dois segmentos de rede
  - Amplifica o sinal para alcançar maiores distâncias
  - Pode fazer regeneração de sinal digital



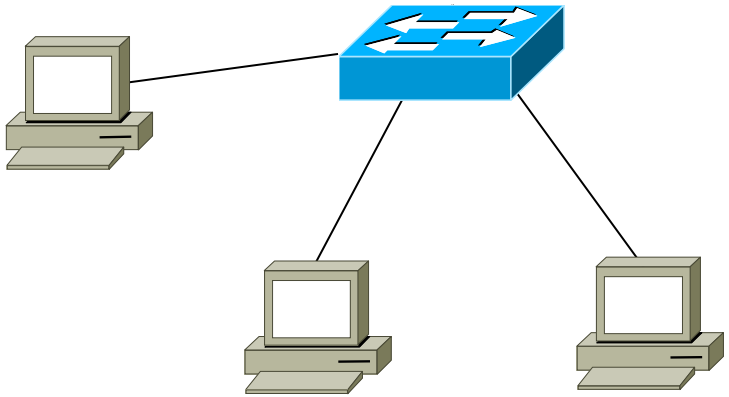
Concentrador (hub)

- Camada Física
- Repetidor com múltiplas portas
  - Sinal recebido numa porta repetido para todas as outras
  - Todas com o mesmo *bit-rate*
  - Ligações *half-duplex*
  - Um único domínio de colisão
  - Topologia física em estrela
  - Topologia lógica em barramento
  - Podem interligar-se (sem ciclos; máx. 4 atravessados)



- *Alguns são dual-speed*
  - Dois segmentos com taxas de transmissão diferentes interligados por uma *bridge*
- Obsoletos

Comutador (switch)



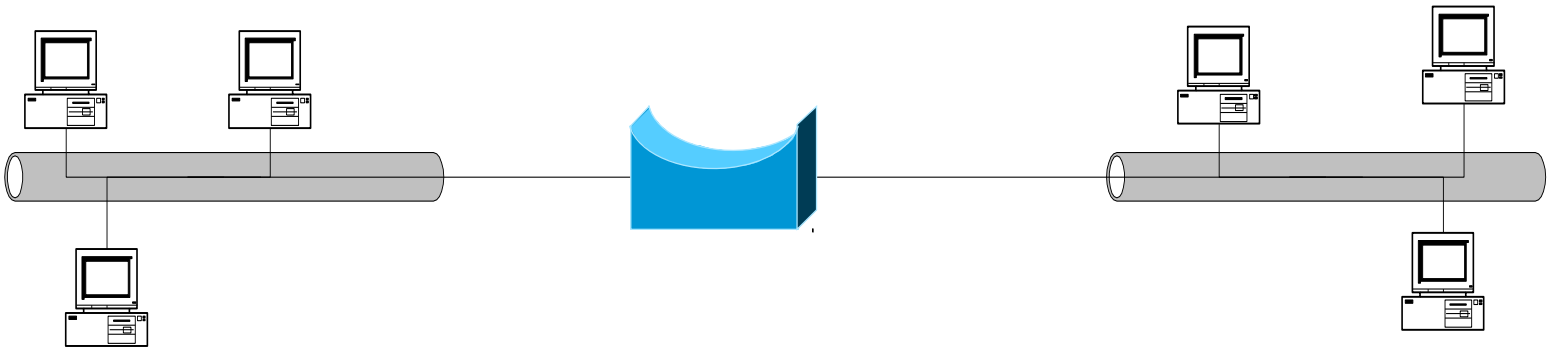
- Camada de Ligação de Dados
- Topologia lógica em estrela
  - Um domínio de colisão por porta física
- Possível ter ligações *full-duplex*
  - Sem colisões
- *Store-and-forward* ou *cut-through*
- Filtra tramas com base nos endereços MAC
  - Reenvia só para a porta onde está o MAC de destino
  - Auto-aprendizagem de associações MAC ↔ porta física
- Podem interligar-se vários
  - Uso de STP (Spannig Tree Protocol) para evitar ciclos
- Alguns podem dividir as portas em múltiplos domínios de difusão
  - VLAN (Virtual LAN)

Tipos de comutador (*switch*)

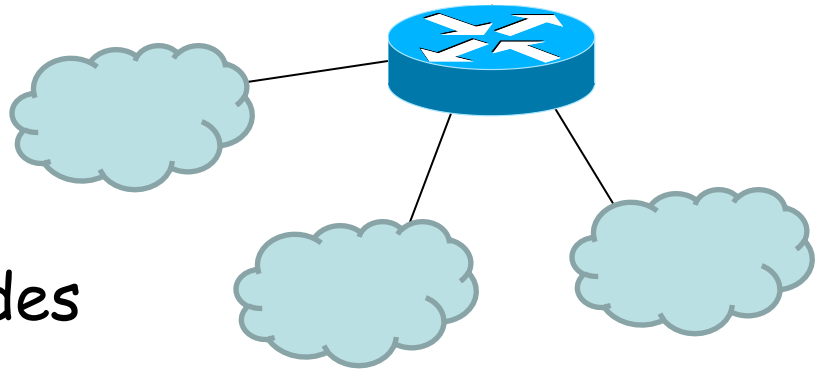
- *Unmanaged ("dumb") switch*
  - Não corre STP e não suporta VLAN, agregação, gestão por SNMP ou outras funcionalidades avançadas; sem *stacking*
  - Sem qualquer tipo de configuração
- *Fully managed switch*
  - Corre STP e suporta VLAN, agregação, gestão SNMP, etc.; pode ser *stackable*
  - Configurável por linha de comando (porta série, telnet, ssh) e normalmente também por interface *web* e/ou *cloud*
- *Smart managed switch*
  - Corre STP e suporta VLAN e agregação, mas não suporta as funcionalidades mais avançadas; alguns são *stackable*
  - Configurável individualmente por interface *web* e/ou de forma integrada com outros equipamentos de rede através de um serviço *cloud*
  - Normalmente não configurável por linha de comando
  - Pode suportar ou não gestão SNMP

Ponte (*bridge*)

- Camada de Ligação de Dados
- Interliga 2 (normalmente) ou mais segmentos de rede
  - Segmentos podem ser de *bit-rate* ou tecnologia diferente
- Faz filtragem por endereço MAC
  - Semelhante a comutador
- Uso de STP para evitar ciclos



Router



- Camada de Rede
- Interliga diferentes sub-redes
  - Múltiplos domínios de difusão
- Reenviam pacotes usando a tabela de encaminhamento
  - Preenchida por protocolos de encaminhamento e/ou rotas estáticas configuradas manualmente
  - Cada router determina de forma independente o próximo salto para cada pacote (datagramas / sem conexões)
- Podem interligar-se em topologias arbitrárias

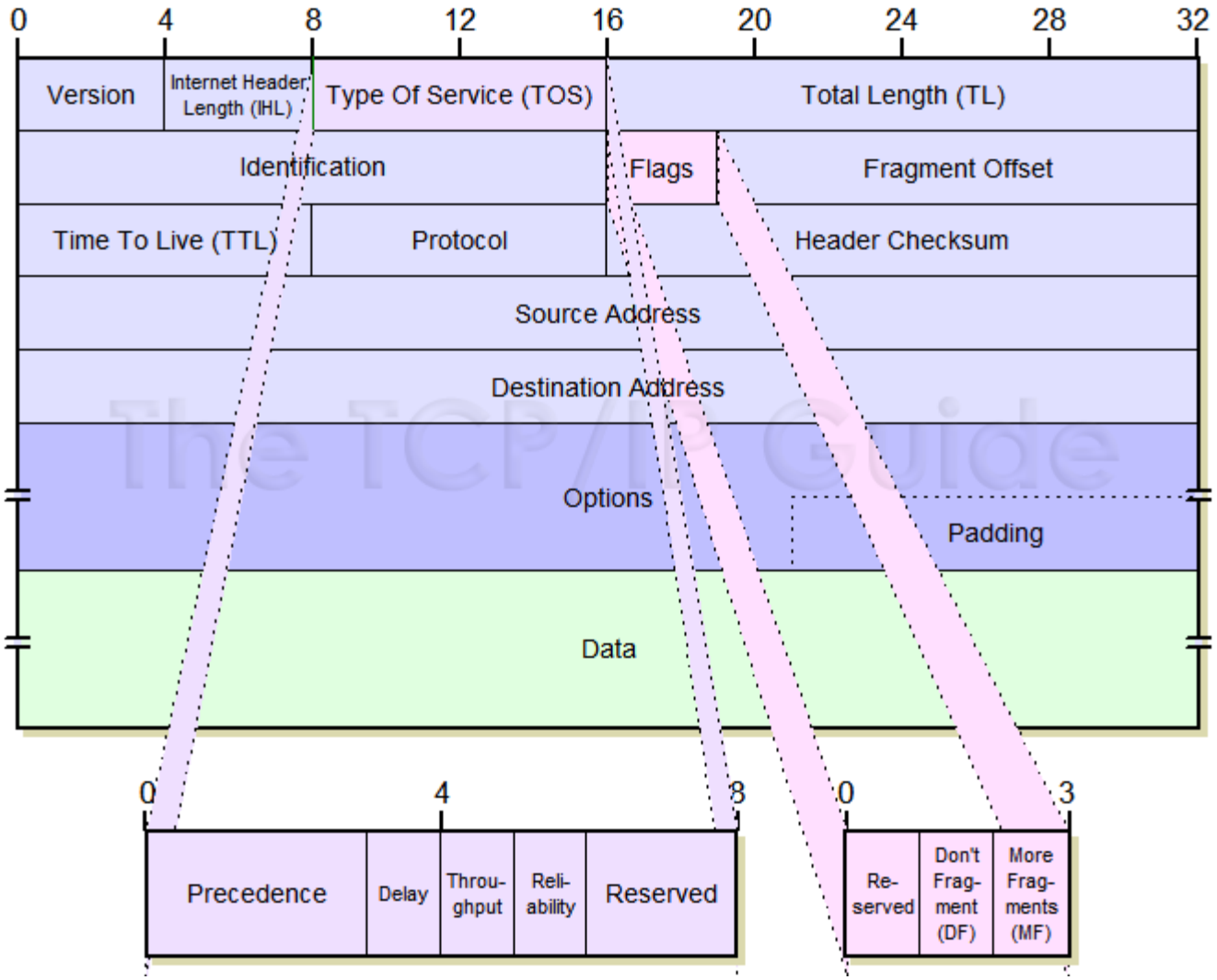
Layer 3 switch, multilayer switch

- Alguns comutadores suportam funções de camadas acima da de ligação de dados (camada 2)
- Um *Layer 3 switch* é um híbrido entre *switch* e *router*
  - Pode fazer reenvio de pacotes entre diferentes portas físicas ou VLAN sem necessidade de um *router* externo
  - Alguns suportam protocolos de encaminhamento, para além de rotas estáticas
- Alguns *multilayer switches* podem ainda suportar funções de camadas superiores à 3
  - E.g., separação por tipo de tráfego (porta TCP/UDP)
- Principais diferenças entre um Layer 3 switch e um router:
  - Um *L3 switch* só tem portas *ethernet* (normalmente muitas); um *router* pode ter outros tipos de portas (portas WAN)
  - Um *L3 switch* faz reenvio de pacotes usando um ASIC (*hardware* dedicado); um *router* pode fazer em *software* a correr no CPU principal

Routers vs terminais

- Terminais são sempre origem e/ou destino de pacotes
  - Descartam pacotes recebidos que não se destinam a nenhuma das suas interfaces
- Routers são intermediários
  - Reenviam pacotes recebidos cujo destino não seja o endereço de uma das interfaces locais
- Configurar PC Linux como router
  - Temporarimente (perde-se ao reiniciar a máquina)
    `sysctl -w net.ipv4.ip_forward=1`
  - Persistentemente (mantém-se após reiniciar a máquina)
    Acrescentar `net.ipv4.ip_forward=1` a `/etc/sysctl.conf` (ou a ficheiro em `/etc/sysctl.d/`)

Cabeçalho IP



(Re)envio de pacotes

- Baseado na tabela de encaminhamento
  - Encaixe de prefixo mais longo
  - Próximo salto é
    - a gateway, se diferente de 0.0.0.0
    - o IP de destino, se directamente ligado (gw=0.0.0.0)
  - Rota default: dest=0.0.0.0, mask=0.0.0.0
    - Prefixo mais curto possível (/0), usada apenas se não encaixar noutra

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

NOTA: Cisco IOS permite gateways indirectas, mas Linux não.

Encaixe de prefixo mais longo

Conceptualmente*

- Para cada entrada na tabela de encaminhamento
  - Fazer um AND bit a bit do endereço IP de destino do pacote com a máscara de rede; no resultado
    - os bits que estão a 1 na máscara ficam iguais aos correspondentes do endereço IP de destino do pacote
    - os bits que estão a 0 na máscara ficam a 0
  - Comparar o resultado com o prefixo
    - Se forem iguais, encaixam ➔ adicionar à lista de encaixes
- Entre todas as entradas que encaixam, seleccionar a que tem maior comprimento de prefixo
  - Máscara com mais bits a 1

*Na prática são usados algoritmos mais eficientes

Encaixe de prefixo mais longo

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

- Exemplo: encontrar próximo salto para pacote destinado a 192.168.1.234

Endereço de destino:

Decimal: 192.168.1.234

Binário: 11000000.10101000.00000001.11101010

Encaixe de prefixo mais longo

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

Prefixo: Decimal: 192.168.5.20

Binário: 11000000.10101000.00000101.00010100

Máscara: Decimal: 255.255.255.255

Binário: 11111111.11111111.11111111.11111111

Encaixe de prefixo mais longo

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

```
     11000000.10101000.00000001.11101010      192.168.  1.234
AND  11111111.11111111.11111111.11111111      255.255.255.255
     11000000.10101000.00000001.11101010      192.168.  1.234
```

192.168.1.234 ≠ 192.168.5.20 ➔ Não encaixa

Encaixe de prefixo mais longo

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

```
     11000000.10101000.00000001.11101010      192.168.  1.234
AND  11111111.11111111.11111111.10000000      255.255.255.128
     ---------------------------------------------------------
     11000000.10101000.00000001.10000000      192.168.  1.128
```

192.168.1.128 ≠ 192.168.1.0 ➔ Não encaixa

Encaixe de prefixo mais longo

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

```
     11000000.10101000.00000001.11101010      192.168.  1.234
AND  11111111.11111111.11111111.00000000      255.255.255.  0
     11000000.10101000.00000001.00000000      192.168.  1.  0
```

192.168.1.0 ≠ 192.168.10.0 ➔ Não encaixa

Encaixe de prefixo mais longo

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

```
     11000000.10101000.00000001.11101010      192.168.  1.234
AND  11111111.11111111.11111110.00000000      255.255.254.  0
     11000000.10101000.00000000.00000000      192.168.  0.  0
```

192.168.0.0 = 192.168.0.0 ➔ Encaixa

Como as entradas estão ordenadas por comprimento de prefixo decrescente, o primeiro encaixe é o de prefixo mais longo

Encaixe de prefixo mais longo

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.5.20	192.168.10.7	255.255.255.255	UGH	1	0	180	eth1
192.168.1.0	192.168.10.5	255.255.255.128	UG	1	0	243	eth1
192.168.10.0	0.0.0.0	255.255.255.0	U	0	0	63311	eth1
192.168.0.0	192.168.10.7	255.255.254.0	UG	1	0	2132	eth1
192.168.18.0	0.0.0.0	255.255.254.0	U	0	0	753430	eth0
192.168.64.0	192.168.10.5	255.255.192.0	UG	1	0	47543	eth1
10.0.0.0	0.0.0.0	255.0.0.0	U	0	0	3123	ppp0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	564	lo
0.0.0.0	192.168.10.20	0.0.0.0	UG	1	0	183436	eth1

- O próximo salto para um pacote destinado a 192.168.1.234 é o 192.168.10.7 (coluna "gateway")
- Se a coluna "gateway" contivesse 0.0.0.0, significaria uma ligação directa à rede de destino
  - Nesse caso, o próximo salto seria o próprio endereço de destino, 192.168.1.234
- Qualquer endereço de destino encaixa na rota default
  - Usada se não encaixar em nenhuma mais específica

Address Resolution Protocol (ARP)

- Dois tipos de ligações de dados
  - Ponto-a-ponto
    - Interligam exactamente duas interfaces
  - Acesso múltiplo
    - Permitem interligar mais de duas interfaces
- Tabela de encaminhamento dá endereço IP do próximo salto
- Interfaces de acesso múltiplo têm endereço MAC
  - Necessário traduzir IP para MAC para enviar a trama para o nó correcto ➔ função do ARP
  - Traduções guardadas na tabela (cache) de ARP, associadas a um prazo de validade

Address Resolution Protocol (ARP)

- Quando a tradução de um endereço IP não está na cache de ARP é necessário "resolver" esse endereço
  - Pedido ARP enviado para o endereço MAC de difusão:
    "Quem tem o endereço IP `X.Y.Z.W`?"
    - Interface onde é enviado determinada pela tabela de encaminhamento
  - Todos os nós recebem e processam a trama, mas apenas o que tem o endereço IP `X.Y.Z.W` responde:
    "Estou aqui, e o meu endereço MAC é o `A:B:C:D:E:F`"
    - Informação é adicionada à cache de ARP
  - Se não houver resposta no espaço de 1 segundo, volta a tentar
  - Se à terceira tentativa não receber resposta, desiste e envia ICMP Host Unreachable (`!H`) ao IP de origem do pacote

Address Resolution Protocol (ARP)

- Resolução bem sucedida:

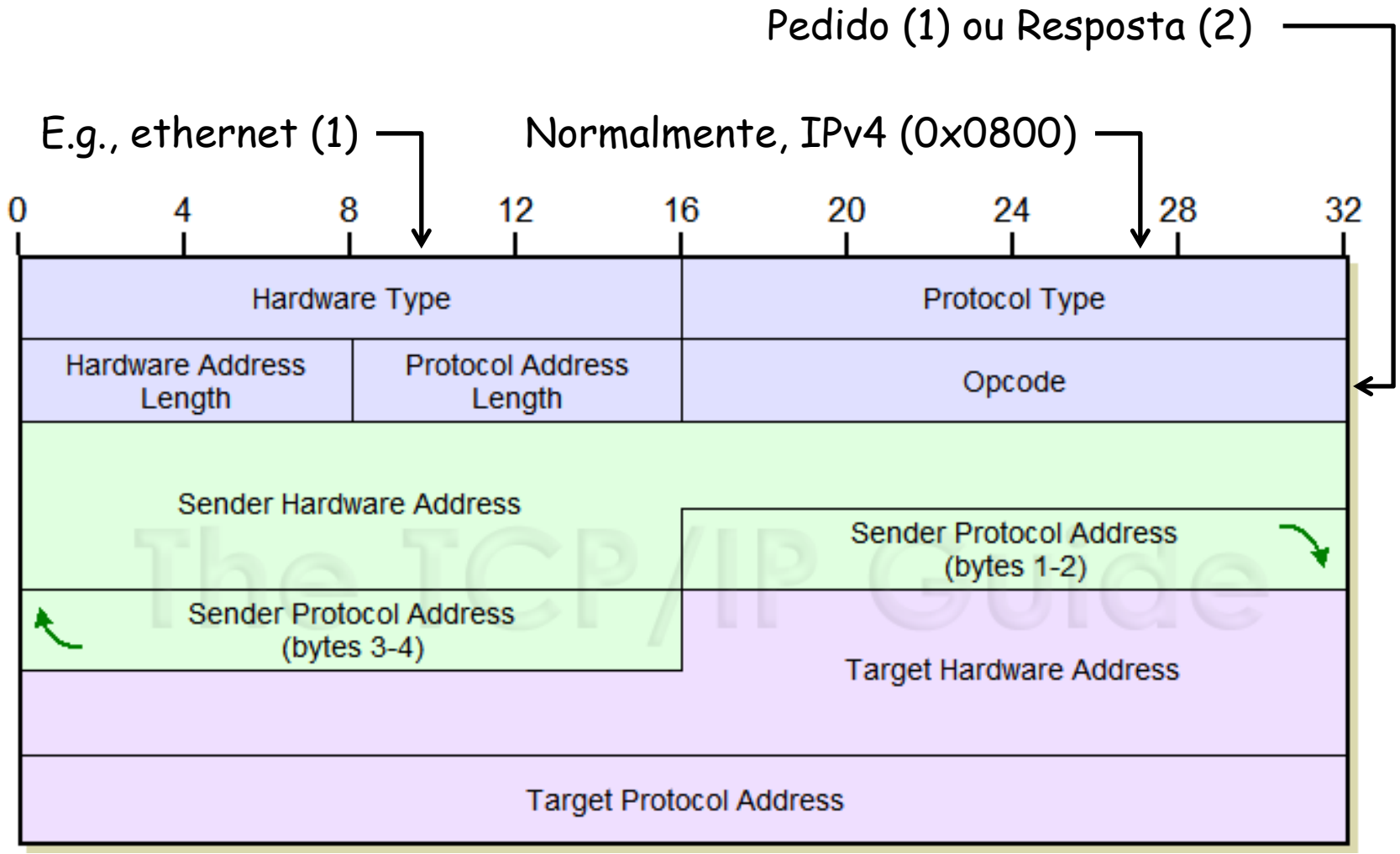
No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.000000000	52:54:00:17:8c:8d	ff:ff:ff:ff:ff:ff	ARP	42	Who has 192.168.122.164? Tell 192.168.122.171
2	0.000362000	52:54:00:60:41:3e	52:54:00:17:8c:8d	ARP	42	192.168.122.164 is at 52:54:00:60:41:3e
3	0.000371000	192.168.122.171	192.168.122.164	ICMP	98	Echo (ping) request id=0x04bf, seq=1/256, ttl=64
4	0.000769000	192.168.122.164	192.168.122.171	ICMP	98	Echo (ping) reply id=0x04bf, seq=1/256, ttl=64

```
⊞ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
⊞ Ethernet II, Src: 52:54:00:17:8c:8d (52:54:00:17:8c:8d), Dst: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
⊟ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 52:54:00:17:8c:8d (52:54:00:17:8c:8d)
    Sender IP address: 192.168.122.171 (192.168.122.171)
    Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.122.164 (192.168.122.164)
```

- Resolução falhada:

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.000000000	52:54:00:17:8c:8d	ff:ff:ff:ff:ff:ff	ARP	42	Who has 192.168.122.10? Tell 192.168.122.171
2	1.001808000	52:54:00:17:8c:8d	ff:ff:ff:ff:ff:ff	ARP	42	Who has 192.168.122.10? Tell 192.168.122.171
3	2.003796000	52:54:00:17:8c:8d	ff:ff:ff:ff:ff:ff	ARP	42	Who has 192.168.122.10? Tell 192.168.122.171
4	3.005808000	192.168.122.171	192.168.122.171	ICMP	126	Destination unreachable (Host unreachable)

Address Resolution Protocol (ARP)



ARP Gratuito

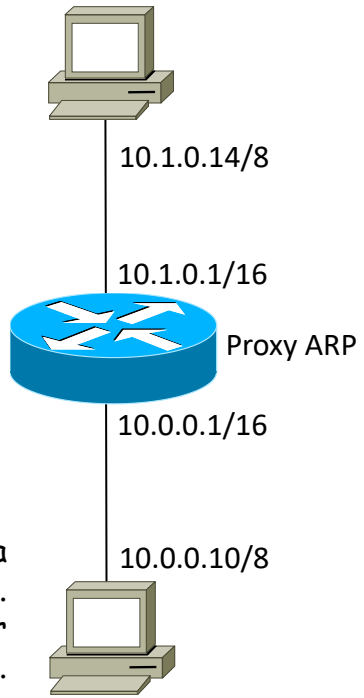
- Pedido ARP em que o *Sender Protocol Address* e o *Target Protocol Address* são o endereço IP de quem envia
- Endereço MAC de destino `ff:ff:ff:ff:ff:ff` (broadcast)
- Utilidade:
  - Actualizar tabelas de ARP de vizinhos quando se activa uma interface, ou se mudar o endereço MAC para um dado IP
    - E.g., substituição de placa de rede
  - Actualizar a tabela de comutação ao mudar de uma porta (física) do comutador para outra

Detecção de endereços duplicados (DAD)

- Pedido ARP em que
  - *O Sender Protocol Address* é 0.0.0.0
  - *O Target Protocol Address* é o endereço IP que se pretende usar
- Usado antes de activar na interface um endereço obtido por DHCP
- Se houver resposta é porque esse IP já está atribuído a outra máquina
  - E.g., por configuração manual
  - O endereço não pode ser usado

Proxy ARP

- Máquina responde a pedido de ARP sobre um endereço IP que não lhe pertence
- Utilidade
  - *Subnetting* transparente
  - Isolar máquinas potencialmente perigosas fazendo parecer que estão na mesma sub-rede



Neste exemplo de subnetting transparemte, os terminais acham que estão na mesma sub-rede (10.0.0.0/8), mas estão em sub-redes diferentes (10.1.0.0/16 e 10.0.0.0/16). Para conseguiem comunicar, o Router tem que fazer Proxy ARP: se o 10.0.0.10 perguntar quem tem o endereço 10.1.0.14, o *router* responde com o seu próprio endereço MAC.

Internet Control Message Protocol (ICMP)

- Protocolo associado ao IP para
  - Relatar erros
  - Controlo
  - Diagnóstico
- Mensagens transportadas directamente sobre IP
- Mensagens contêm
  - Tipo
  - Código
  - Primeiros 8 bytes do pacote que desencadeou o seu envio

Alguns exemplos:

Tipo	Cód.	Descrição
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (not used)
5	0	redirect for network
5	1	redirect for host
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL exceeded
12	0	bad IP header

netstat

- Obter informação sobre
  - Sockets
  - Interfaces de rede
  - Tabela de encaminhamento
  - Etc.

netstat — interfaces

```
$ netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
lo       65536        8      0      0 0             8      0      0      0 LRU
eth0      1500   997478      0      0 0         53425      0      0      0 BMRU


$ netstat -iea
Kernel Interface table
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 8  bytes 1104 (1.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 1104 (1.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.51  netmask 255.255.255.0  broadcast 192.168.50.255
        inet6 fe80::f66d:4ff:fed8:759b  prefixlen 64  scopeid 0x20<link>
        ether f4:6d:04:d8:75:9b  txqueuelen 1000  (Ethernet)
        RX packets 997510  bytes 364988550 (348.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 53436  bytes 4133903 (3.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

netstat — tabela de encaminhamento

```
$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.50.1    0.0.0.0         UG        0 0          0 p5p1
192.168.50.0    0.0.0.0         255.255.255.0   U         0 0          0 p5p1
```

Opção -n ➔ não traduzir endereços IP para nomes DNS ➔ aconselhável

netstat — sockets

```
$ netstat -uln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:67              0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 192.168.50.51:39231     0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
udp        0      0 0.0.0.0:856             0.0.0.0:*


$ netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0    224 192.168.50.51:22        192.168.50.75:59938     ESTABLISHED


# netstat -tpn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 192.168.50.41:43559     10.0.0.251:80           ESTABLISHED
20176/wget
```

ifconfig

- Configurar interfaces de rede
  - Necessita de permissões de *root*
  - Acção imediata e efémera (perde-se num *reboot*)

- Ver informação sobre as interfaces
  - Só as activas: `ifconfig` (equivalente a `netstat -ie`)
  - Todas: `ifconfig -a` (equivalente a `netstat -iea`)

ifconfig

- Activar interface `eth0`: `ifconfig eth0 up`
  - Não altera a configuração (endereço IP, máscara, etc.)

- Desactivar interface `eth0`: `ifconfig eth0 down`
  - Idem

- Configurar interface:
  `ifconfig eth0 192.168.1.234 netmask 255.255.255.0 up`

route

- Manipular tabela de encaminhamento
  - Necessita de permissões de *root*
  - Acção imediata e efémera (perde-se num *reboot*)
- Ver tabela: `route` (equivalente a `netstat -r`)
- Adicionar rota:

```
route add -net 172.16.1.0 netmask 255.255.255.0 gw 192.168.1.1
```

- Remover rota:

```
route del -net 172.16.1.0 netmask 255.255.255.0 gw 192.168.1.1
```

arp

- Listar cache de ARP

```
$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.0.5                         (incomplete)                              eth1
192.168.50.1             ether   00:04:23:ce:b8:df   C                     eth0
192.168.50.99            ether   00:23:18:c1:d0:95   C                     eth0
```

- Permite filtragem por endereço…

```
$ arp -n 192.168.50.1
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.50.1             ether   00:04:23:ce:b8:df   C                     eth0
```

- … ou por interface

```
$ arp -n -i eth0
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.50.1             ether   00:04:23:ce:b8:df   C                     eth0
192.168.50.99            ether   00:23:18:c1:d0:95   C                     eth0
```

arp

- Apagar uma entrada

```
# arp -d 192.168.50.1
```

- Criar manualmente uma entrada

```
# arp -s 192.168.50.1 00:04:23:ce:b8:df temp
```

- Proxy ARP — flag "publish" activa
  - No Linux só funciona se se cumprirem três requisitos
    - Reenvio de pacotes estiver activo
    - Existir uma rota para o destino
    - Essa rota é através duma interface diferente

```
# arp -i eth0 -Ds 192.168.60.1 eth0 pub
```

arping

- Envio de mensagens ARP controlado pelo utilizador
- ARP Gratuito

```
# arping -c 1 -I eth0 -U 192.168.122.10
ARPING 192.168.122.10 from 192.168.122.10 eth0
Sent 1 probes (1 broadcast(s))
Received 0 response(s)
```

- Detecção de endereço duplicado (DAD)

```
# arping -c 3 -I eth0 -D 192.168.122.10
ARPING 192.168.122.10 from 0.0.0.0 eth0
Sent 3 probes (3 broadcast(s))
Received 0 response(s)
```

arping

- Testar conectividade para outra máquina
  - Só na mesma LAN porque o ARP não é encaminhável

```
# arping 172.30.48.1
ARPING 172.30.48.1 from 172.30.60.16 eth0
Unicast reply from 172.30.48.1 [00:15:5D:18:28:20]  0.989ms
Unicast reply from 172.30.48.1 [00:15:5D:18:28:20]  0.821ms
Unicast reply from 172.30.48.1 [00:15:5D:18:28:20]  1.156ms
^CSent 3 probes (1 broadcast(s))
Received 3 response(s)
```

iproute2 (comandos ip e ss)

- Comando `ip` é "canivete suíço" da configuração de rede
  - Substitui a maior parte dos comandos tradicionais de configuração de rede
  - Possibilita configurações anteriormente inexistentes
- Comando `ss` para obter informação sobre sockets
- Ferramentas exclusivas do Linux…
  - Sintaxe e formatos de saída próprios

iproute2 (comandos ip e ss)

Alguns comandos tradicionais e equivalentes iproute2:

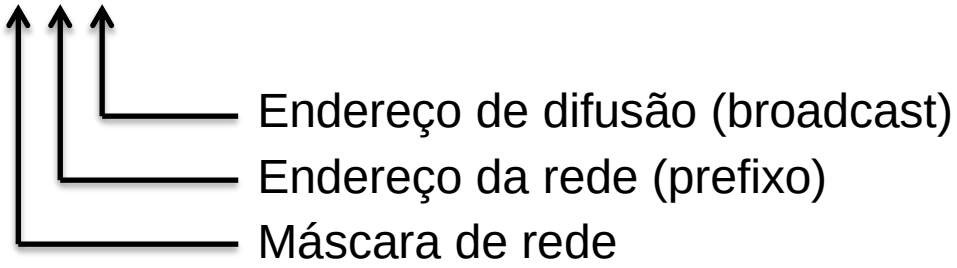
arp	ip neigh
ifconfig	ip link / ip addr
ifconfig eth0 up	ip link set eth0 up
netstat	ss
netstat -i	ip -s link
netstat -r	ip route
route add	ip route add
route del	ip route del

Ajuda: ip <obj> help ou man ip-<obj>
Exemplos: ip link help
man ip-route

ipcalc

- Simplifica o cálculo com endereços e máscaras
- Exemplo:

```
ipcalc -mnb 192.168.27.175/25
```




```
NETMASK=255.255.255.192
BROADCAST=192.168.27.191
NETWORK=192.168.27.128
```

- Opção `--class-prefix` para usar o comprimento de prefixo da rede *classful*

Ficheiros de configuração

- Os comandos anteriormente descritos têm acção imediata e efémera (perdem-se num *reboot*)
- Configurações permanentes fazem-se através de ficheiros de configuração
- Muitas distribuições Linux (incluindo Fedora) usam o NetworkManager e guardam as configurações em *keyfiles*
  - Ficheiros em `/etc/NetworkManager/system-connections`
  - Podem alterar-se com um editor de texto, a GUI, `nmtui`, `nmcli` ou outras ferramentas
- Cada *keyfile* descreve uma *conexão* (configuração específica para uma interface de rede)
- Alterações aos *keyfiles* não são aplicadas imediatamente
  - É preciso recarregá-las e activá-las (e.g., usando o `nmcli`)
    - Examplo: `nmcli con reload; nmcli con up eth0`

Exemplo de *keyfile*

- Ficheiro `/etc/NetworkManager/system-connections/eth0.nmconnection`

```
[connection]
id=eth0                                     # Identificador da conexão
uuid=d43b7a46-0dff-9d53-1068-ccc58c977db3
type=ethernet
interface-name=eth0                         # Nome da interface

[ipv4]
method=manual                               # Usar "auto" para DHCP
address1=192.168.56.173/24,192.168.56.1     # End.IP, preflen & defaultgw
dns=192.168.56.2;192.168.56.3               # Servidor(es) DNS recursivo(s)
route1=10.0.0.0/8,192.168.56.254            # Rota estática (prefixo e gw)

[ipv6]
method=ignore
```

Servidores de nomes

- Tradicionalmente, a configuração dos servidores DNS era feita no `/etc/resolv.conf`

```
search example.com local.lan
nameserver 172.16.1.254
nameserver 172.16.2.254
```

- *Agora é feita nos keyfiles*

- Fedora e outras usam o serviço `systemd-resolved`
  - Pode verificar-se os servidores de nomes em `/run/systemd/resolve/resolv.conf`
  - Ou usando o comando `systemd-resolve --status`

Indentificar interfaces de rede

- No Linux, a maneira mais bem suportada é
  - Usar o comando `ip monitor link`
  - Ligar/desligar o cabo
  - Só funciona em interfaces activas (*up*)
    - Mas podem não estar completamente configuradas (e.g., sem endereço IP)

```
# ip monitor link
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast
state DOWN group default
    link/ether f4:6d:04:d8:75:9b brd ff:ff:ff:ff:ff:ff
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default
    link/ether f4:6d:04:d8:75:9b brd ff:ff:ff:ff:ff:ff
^C
```

ping

- Permite verificar se
  - Máquina com um dado IP existe e está ligada
    - e -
  - Há comunicação bidireccional com ela
- Baseia-se em mensagens ICMP Echo Request e Reply
- Opções:
  - Broadcast: `-b` (necessário para endereços broadcast)
  - Flood: `-f` sem espera entre Echo Requests sucessivos (apenas acessível a `root`)
  - Sem resolução de endereços: `-n`
  - Configurar tamanho do pacote: `-s`
  - Mais: `man ping`

ping

- Exemplo 1: Sucesso

```
$ ping 193.137.55.13
PING 193.137.55.13 (193.137.55.13) 56(84) bytes of data.
64 bytes from 193.137.55.13: icmp_seq=1 ttl=251 time=0.975 ms
64 bytes from 193.137.55.13: icmp_seq=2 ttl=251 time=1.77 ms
64 bytes from 193.137.55.13: icmp_seq=3 ttl=251 time=1.64 ms
64 bytes from 193.137.55.13: icmp_seq=4 ttl=251 time=1.58 ms
64 bytes from 193.137.55.13: icmp_seq=5 ttl=251 time=1.63 ms
64 bytes from 193.137.55.13: icmp_seq=6 ttl=251 time=1.36 ms

--- 193.137.55.13 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5004ms
rtt min/avg/max/mdev = 0.975/1.496/1.779/0.263 ms
```

ping

- Exemplo 2: Insucesso

```
$ ping 193.137.55.1
PING 193.137.55.1 (193.137.55.1) 56(84) bytes of data.
```

(passado algum tempo de espera, CTRL-C)

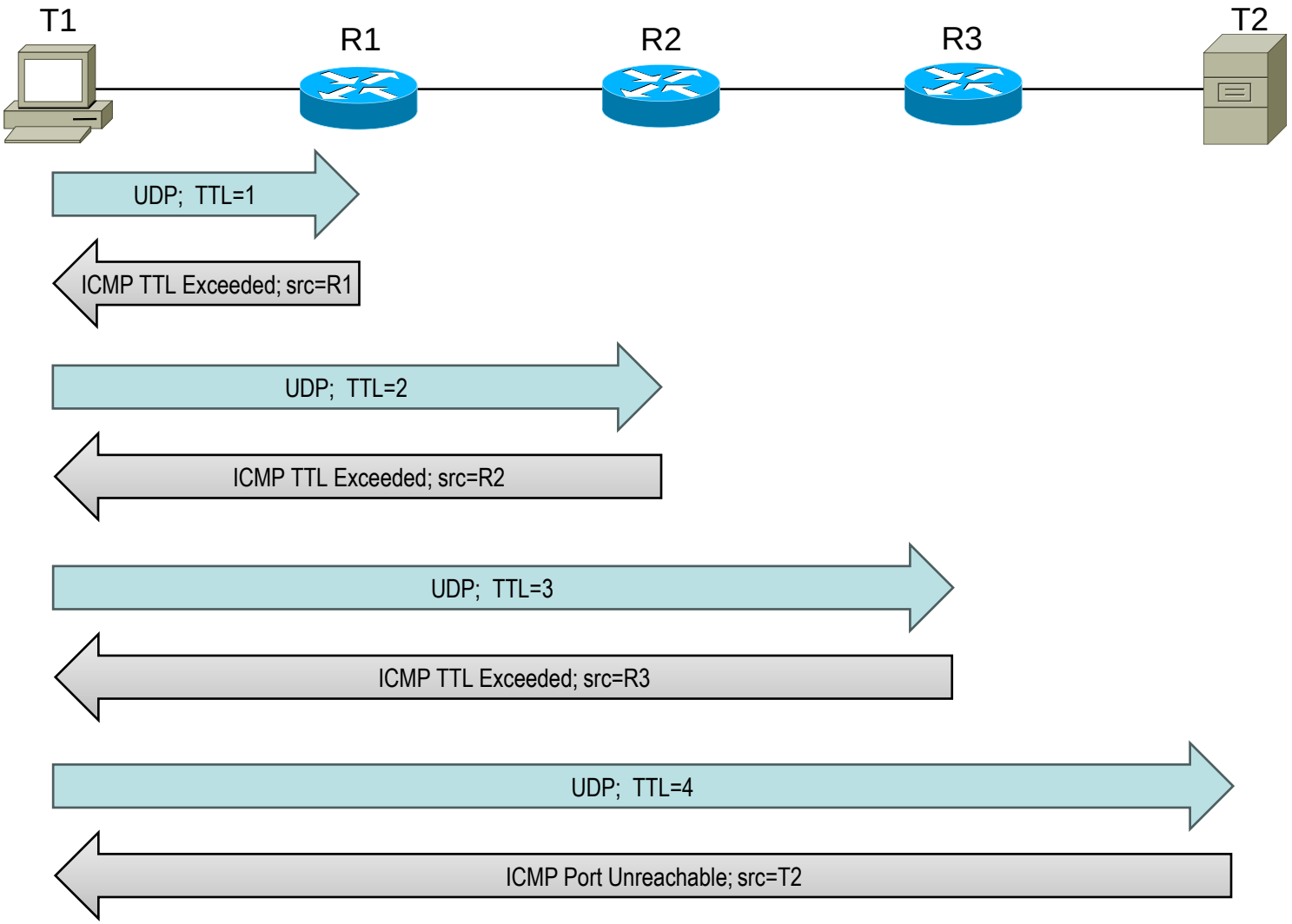
```
--- 193.137.55.1 ping statistics ---
20 packets transmitted, 0 received, 100% packet loss, time 18997ms
```

- Múltiplas causas possíveis:
  - Máquina não existe / está desligada / crashou / …
  - Falha de comunicação
    - Falta rota em sentido directo ou inverso
    - Ciclo de encaminhamento
    - Cabo desligado / mal ligado / defeituoso
    - …

traceroute

- Traçar o percurso até ao nó de destino (routers atravessados)
- Envio de segmentos UDP para nó de destino
  - Porta alta escolhida aleatoriamente, quase de certeza sem nada à escuta
  - Três pacotes com TTL=1, três pacotes com TTL=2, …
- TTL decrementado em cada salto, chega a 0 no n-ésimo salto
  - Pacote é descartado
  - Router envia mensagem ICMP TTL Exceeded ao nó de origem
  - Endereço IP de origem da mensagem ICMP identifica esse router
  - Ao receber o ICMP TTL Exceeded, o nó de origem calcula o RTT e imprime
  - Se não receber nada, ao fim de um certo tempo imprime um asterisco *
- Critério de paragem
  - Segmento UDP chega ao nó de destino, porta não tem nenhum processo à escuta
  - Nó de destino envia ICMP Port Unreachable ao nó de origem
  - Ao receber esta mensagem, o nó de origem sabe que chegou ao fim

traceroute


T1
R1
R2
R3
T2
UDP; TTL=1
ICMP TTL Exceeded; src=R1
UDP; TTL=2
ICMP TTL Exceeded; src=R2
UDP; TTL=3
ICMP TTL Exceeded; src=R3
UDP; TTL=4
ICMP Port Unreachable; src=T2

traceroute

- Opções:
  - Sem resolução de nomes: `-n` (aconselhável quando não há DNS)
  - Um segmento UDP de cada vez: `-N 1` (aconselhável para análise)
  - Mais informação: `man traceroute`

- Exemplo 1: Rota traçada até ao destino

```
$ traceroute -n -N 1 193.137.55.9
traceroute to 193.137.55.9 (193.137.55.9), 30 hops max, 38 byte packets
 1  10.0.0.1  0.269 ms  0.123 ms  0.100 ms
 2  193.136.39.1  2.038 ms  1.961 ms  1.958 ms
 3  193.137.26.1  0.928 ms  0.732 ms  1.116 ms
 4  193.136.54.25  1.155 ms  1.839 ms  *
 5  193.137.55.9  1.337 ms  1.472 ms  4.253 ms
```

traceroute

- Exemplo 2: A partir de um certo nó não recebe nada

```
$ traceroute -n -N 1 193.137.55.1
traceroute to 193.137.55.1 (193.137.55.1), 30 hops max, 38 byte packets
 1  10.0.0.1  0.266 ms  0.108 ms  0.099 ms
 2  193.136.39.1  2.038 ms  1.966 ms  1.974 ms
 3  193.137.26.1  0.914 ms  0.668 ms  0.782 ms
 4  193.136.54.25  1.000 ms  1.255 ms  1.192 ms
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
```

Diversas causas possíveis

- Falta de rota para o destino
  - Normalmente recebe-se também ICMP Host Unreachable, mas pode falhar
- Falta de rota em sentido inverso
  - Nós enviam pacotes ICMP, mas estes não chegam ao nó de origem
- Ciclo de encaminhamento em sentido inverso
  - Idem

traceroute

- Exemplo 3: Ciclo de encaminhamento em sentido directo

```
$ traceroute -n 193.137.55.111
traceroute to 193.137.55.111 (193.137.55.111), 30 hops max, 38 byte packets
 1  10.0.0.1  0.191 ms  0.104 ms  0.093 ms
 2  193.136.39.1  2.026 ms  1.980 ms  1.964 ms
 3  193.137.26.1  1.102 ms  1.036 ms  0.922 ms
 4  193.136.54.25  2.500 ms  1.634 ms  1.566 ms
 5  193.136.4.37  1.100 ms  1.175 ms  1.447 ms
 6  193.136.4.38  1.379 ms  4.179 ms  1.170 ms
 7  193.136.4.37  1.636 ms  1.754 ms  2.141 ms
 8  193.136.4.38  2.041 ms  3.945 ms  2.107 ms
 9  193.136.4.37  1.063 ms  1.730 ms  1.460 ms
10  193.136.4.38  1.647 ms  1.796 ms  2.130 ms
11  193.136.4.37  1.263 ms  1.562 ms  2.164 ms
12  193.136.4.38  2.316 ms  2.279 ms  2.025 ms
13  193.136.4.37  1.769 ms  1.972 ms  1.323 ms
14  193.136.4.38  1.736 ms  5.455 ms  1.818 ms
15  193.136.4.37  2.625 ms  2.387 ms  2.651 ms
```

} Sequência de routers repetida

"Não tenho acesso à Internet…"

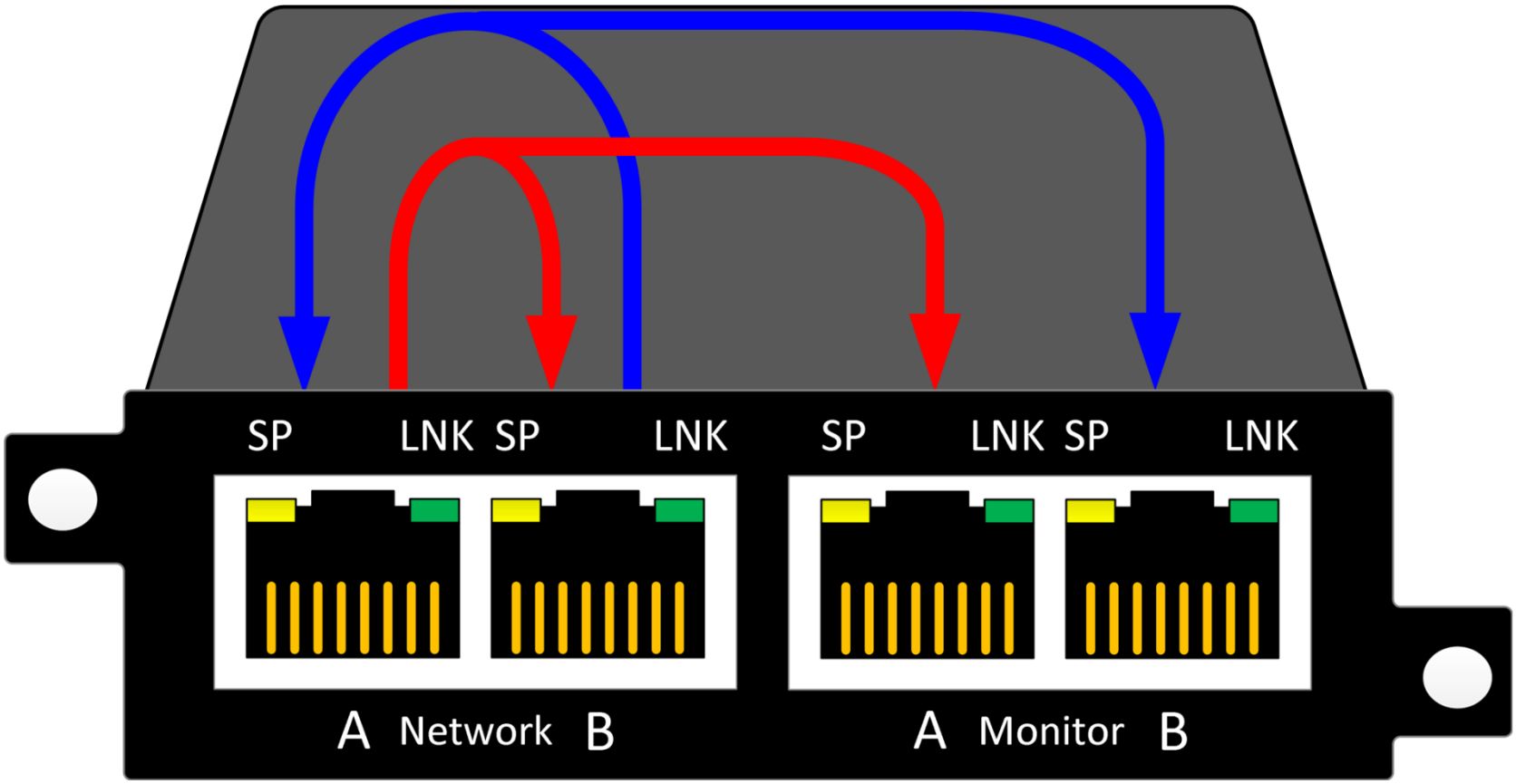
- Verificar
  - Configuração da interface (end. IP, netmask, estado)
  - Existência de rota-padrão
  - Comunicação bidireccional com *gateway* (`ping`)*
  - Configuração de servidores DNS
  - Comunicação bidireccional com servidor DNS (`ping`)*
  - Resolução de nomes (e.g., `host www.dcc.fc.up.pt`)
  - Encaminhamento para lá da *gateway* (`traceroute`)

* Poderão estar configurados para não responder a `ping`. Se estiverem na mesma sub-rede (a *gateway* tem que estar), pode também usar-se o `arping`.

Captura e análise de pacotes

- Análise de pacotes é útil para
  - Identificar problemas na rede
  - Monitorizar a rede
  - Detectar intrusões
  - Aprender sobre os protocolos
- Feita por Analisadores de Pacotes / Analisadores de Protocolos
  - Grau de sofistiação bastante variável
- Fácil em tecnologias de acesso múltiplo com meio partilhado (e.g., ethernet com concentradores)
- Mais difícil em tecnologias sem essas características (e.g., ethernet com comutadores)
  - Possível com *port mirroring* (SPAN em equipamentos Cisco)
  - Ou com *network taps*

Network tap



tcpdump

- Analisador de pacotes
- Baseado em linha de comando
- Usa biblioteca pcap para efectuar captura
  - Por interface ou geral
  - Suporta modo promíscuo (sem filtragem por endereço MAC)
- Permite
  - Utilizar filtros para limitar a captura ao que interessa
    - Sintaxe dos filtros em formato BPF (Berkeley Packet Filtering)
  - Gravar captura em disco para análise posterior

tcpdump — exemplo

```
# tcpdump -nn -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:56:30.302317 IP 192.168.50.41.53266 > 192.168.50.2.53: 30097+ A? www.dcc.fc.up.pt. (34)
16:56:30.302772 IP 192.168.50.2.53 > 192.168.50.41.53266: 30097* 1/1/1 A 10.0.0.251 (84)
16:56:30.558758 IP 192.168.50.41.37625 > 10.0.0.251.80: Flags [S], seq 2588667828, win 29200,
options [mss 1460,sackOK,TS val 98616760 ecr 0,nop,wscale 7], length 0
16:56:30.559007 IP 10.0.0.251.80 > 192.168.50.41.37625: Flags [S.], seq 1588416895, ack
2588667829, win 5792, options [mss 1460,sackOK,TS val 374135764 ecr 98616760,nop,wscale 6],
length 0
16:56:30.559052 IP 192.168.50.41.37625 > 10.0.0.251.80: Flags [.], ack 1, win 229, options
[nop,nop,TS val 98616760 ecr 374135764], length 0
16:56:30.559157 IP 192.168.50.41.37625 > 10.0.0.251.80: Flags [P.], seq 1:151, ack 1, win 229,
options [nop,nop,TS val 98616760 ecr 374135764], length 150
16:56:30.559354 IP 10.0.0.251.80 > 192.168.50.41.37625: Flags [.], ack 151, win 108, options
[nop,nop,TS val 374135765 ecr 98616760], length 0
16:56:30.562779 IP 10.0.0.251.80 > 192.168.50.41.37625: Flags [P.], seq 1:650, ack 151, win 108,
options [nop,nop,TS val 374135765 ecr 98616760], length 649
16:56:30.562793 IP 192.168.50.41.37625 > 10.0.0.251.80: Flags [.], ack 650, win 239, options
[nop,nop,TS val 98616764 ecr 374135765], length 0
16:56:30.563112 IP 192.168.50.41.37625 > 10.0.0.251.80: Flags [P.], seq 151:302, ack 650, win
239, options [nop,nop,TS val 98616764 ecr 374135765], length 151
16:56:30.572363 IP 10.0.0.251.80 > 192.168.50.41.37625: Flags [P.], seq 650:3114, ack 302, win
124, options [nop,nop,TS val 374135766 ecr 98616764], length 2464
```

Wireshark

- Analisador de protocolos
  - Interface gráfica (GUI)
  - Captura baseada na biblioteca pcap
    - Compatível com capturas feitas pelo tcpdump
    - Filtros de captura com a mesma sintaxe do tcpdump (BPF)
  - Filtros de visualização
    - Sintaxe própria
    - Dinâmicos
  - Capaz de relacionar sequências de pacotes

Wireshark

Menus de comando

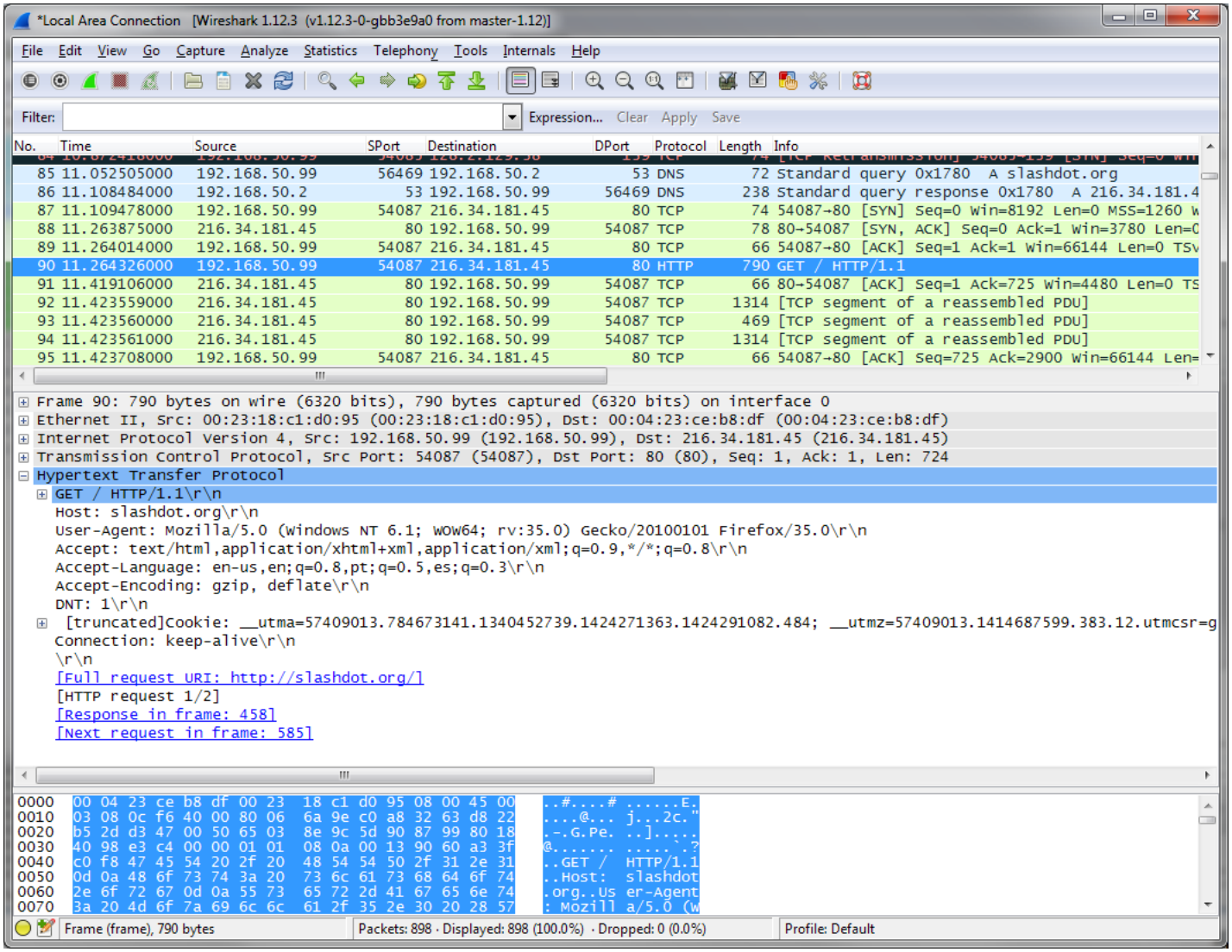
Filtro de visualização

Lista de pacotes capturados

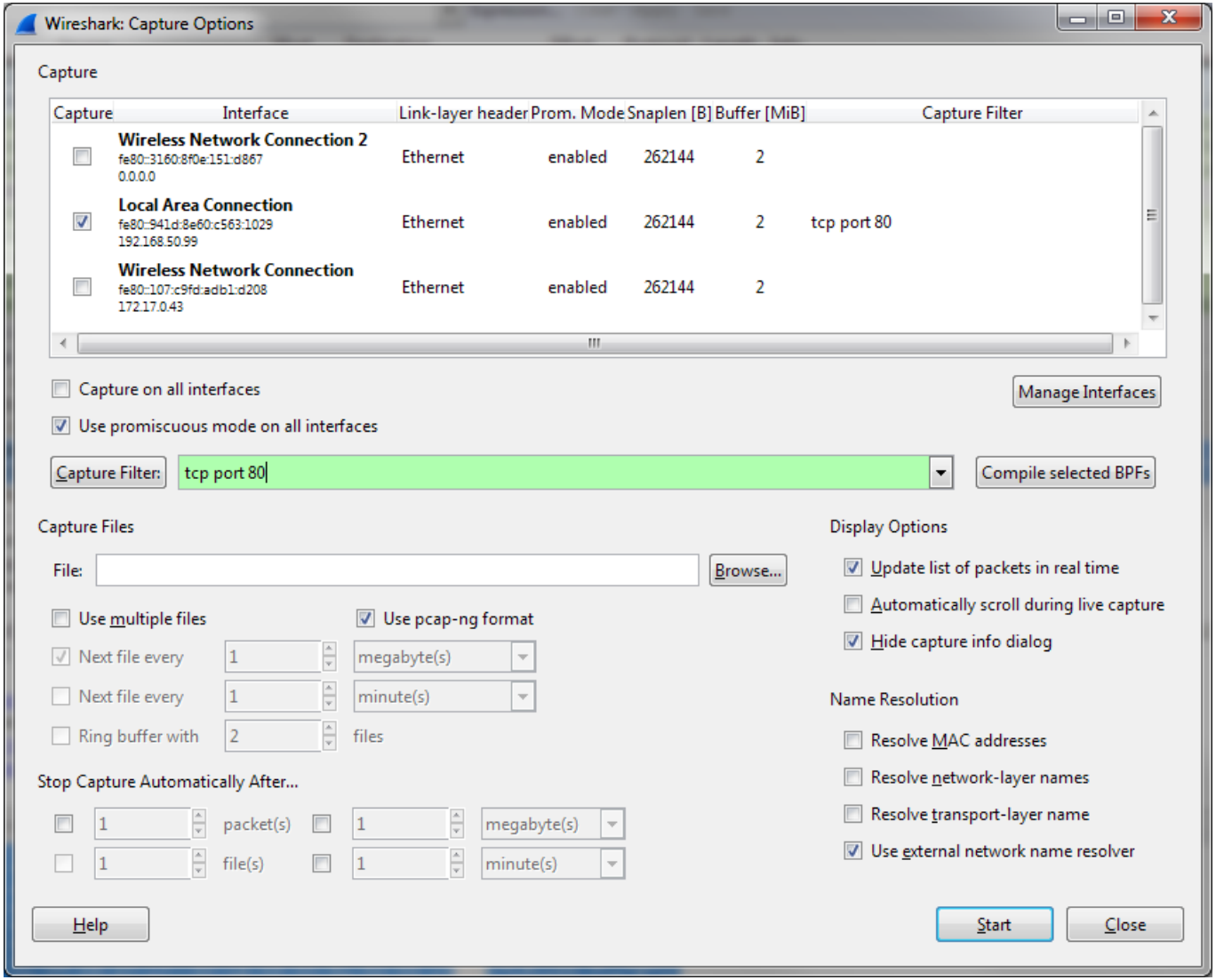
Detalhes do pacote seleccionado (por protocolo)

Conteúdo do pacote em hexadecimal e ASCII

Informação adicional



Wireshark


Wireshark: Capture Options
Capture
Capture
Interface
Link-layer header
Prom. Mode
Snaplen [B]
Buffer [MiB]
Capture Filter
Wireless Network Connection 2
fe80::3160:8f0e:151:d867
0.0.0.0
Ethernet
enabled
262144
2
Local Area Connection
fe80::941d:8e60:c563:1029
192.168.50.99
Ethernet
enabled
262144
2
tcp port 80
Wireless Network Connection
fe80::107:c9fd:adb1:d208
172.17.0.43
Ethernet
enabled
262144
2
Capture on all interfaces
Manage Interfaces
Use promiscuous mode on all interfaces
Capture Filter:
tcp port 80
Compile selected BPFs
Capture Files
File:
Browse...
Use multiple files
Use pcap-ng format
Next file every
1
megabyte(s)
Next file every
1
minute(s)
Ring buffer with
2
files
Stop Capture Automatically After...
1
packet(s)
1
megabyte(s)
1
file(s)
1
minute(s)
Display Options
Update list of packets in real time
Automatically scroll during live capture
Hide capture info dialog
Name Resolution
Resolve MAC addresses
Resolve network-layer names
Resolve transport-layer name
Use external network name resolver
Help
Start
Close

Wireshark — Filtros

- Filtros de captura
  - Limitam a quantidade de pacotes capturados / a analisar
  - Configurados antes da captura
  - Sintaxe definida pela biblioteca pcap (compatíveis com o tcpdump)
  - Úteis quando sabemos em que pacotes estamos interessados / que pacotes não nos interessam
- Filtros de visualização
  - Definem uma vista sobre a captura
  - Mais poderosos, com sintaxe própria
  - Configurados em qualquer momento / aplicáveis dinamicamente
  - Úteis para limitar o número de pacotes durante a análise
  - Também podem ser usados para captura no tshark* com a opção -R

* O tshark é uma ferramenta de linha de comando incluída no pacote Wireshark, mas menos eficiente que o tcpdump para capturas a débitos elevados

Filtros de captura

- Um filtro de captura consiste numa ou mais *primitivas*
  - Ligadas por *operadores lógicos* (`!`, `not`, `&&`, `and`, `||`, `or`)
  - Possível agrupar com parêntesis
- Uma *primitiva* consiste
  - Num *id* (nome ou número) precedido por zero ou mais *qualificadores*
    - Os *qualificadores* podem ser de
      - *Tipo* (`host`, `port`, …)
      - *Direcção* (`src`, `dst`, …)
      - *Protocolo* (`arp`, `tcp`, …)
  - Numa sequência *expressão comparador expressão* em que
    - As comparações são sempre sem sinal
    - As *expressões* são compostas por
      - *Constantes* (`tcpflags`, `tcp-ack`, …)
      - *Operadores* aritméticos e/ou de bit
      - Operador de tamanho (`len`)
      - *Campos extraídos de pacotes* usando a sintaxe `protocolo[expressão:tamanho]`

Ver `man pcap-filter`

Posição no cabeçalho (*offset*)

Exemplos de filtros de captura

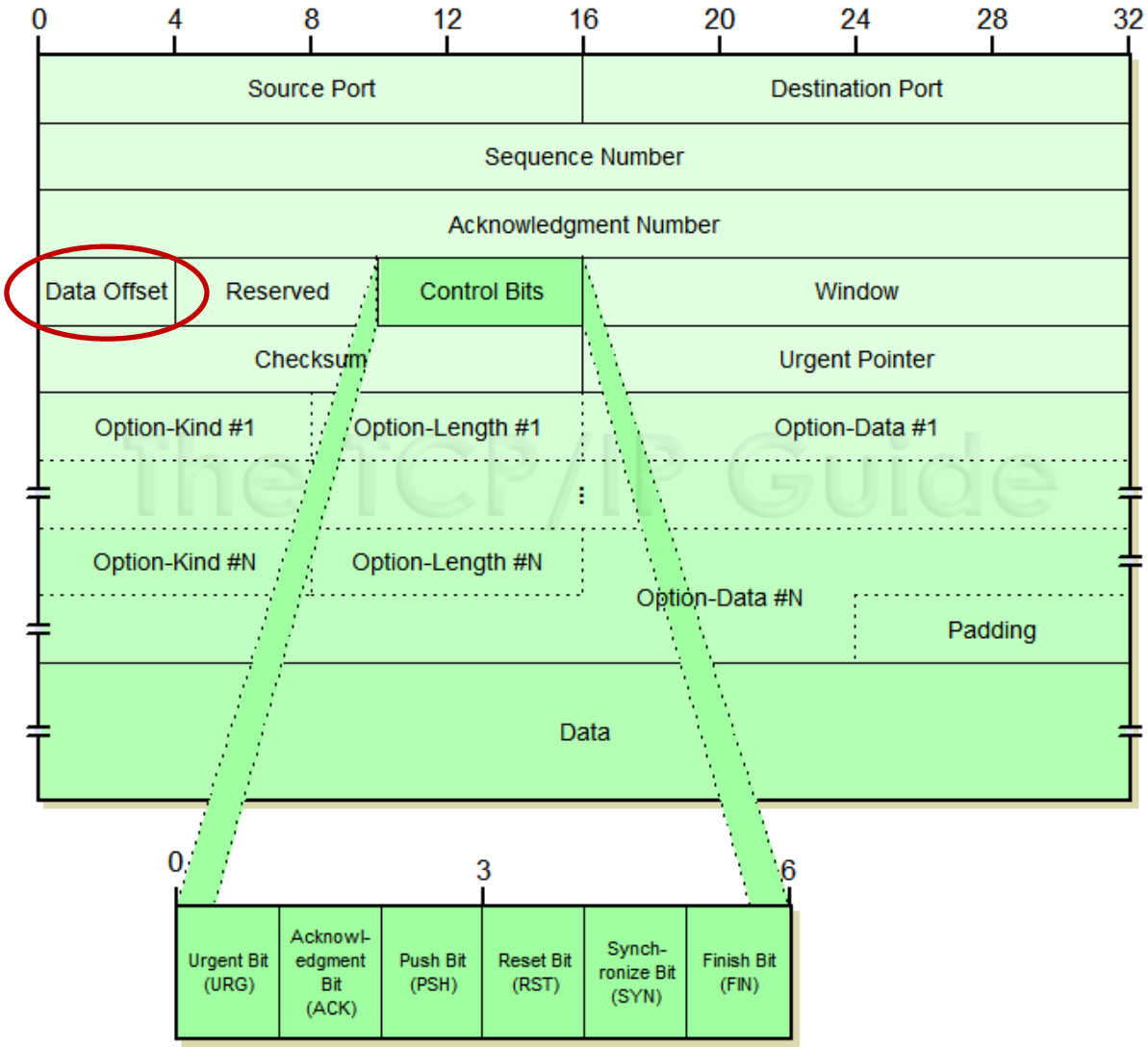
- Pacotes de/para um dado endereço IP
  `host 172.18.5.4`
- Pacotes destinados a uma gama de endereços
  `dst net 192.168.1.0/24`
- Pacotes UDP de/para uma dada porta (neste caso, pacotes DNS)
  `udp port 53`
- Respostas DNS (usando notação genérica para extrair campos)
  `udp[0:2] = 53`
- Pacotes TCP de/para a gama de portas 1500-1550
  `tcp portrange 1500-1550`
- Pacotes com comprimento menor que 100 bytes, incluindo os cabeçalhos da camada de ligação de dados
  `len < 100`

Exemplos de filtros de captura

- Tráfego em qualquer VLAN 802.1Q
  `vlan`
- Pacotes ICMP Echo Request (pedidos ping) na VLAN 100
  `vlan 100 and icmp[icmptype] = icmp-echo`
- Pacotes trocados com o www.dcc.fc.up.pt excepto tráfego http
  `host www.dcc.fc.up.pt and not port 80`
- Pedidos de estabelecimento de conexão TCP (SYN mas não ACK)
  `tcp[tcpflags] & (tcp-syn | tcp-ack) = tcp-syn`
- Pedidos HTTP GET
  `port 80 and tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420`
  - Caracteres 'G' (0x47), 'E' (0x45), 'T' (0x54) e ' ' (0x20) logo a seguir ao cabeçalho TCP
  - Comprimento do cabeçalho TCP dado por `(tcp[12:1] & 0xf0) >> 2`

Cabeçalho TCP

Offset em palavras de 32 bits



Filtros de visualização

- Diferentes tipos de campos
  - Inteiros com e sem sinal entre 8 e 32 bits
  - Booleanos
  - Strings
  - Endereços: ethernet, IPv4 e IPv6
- Partes de campos
  - Notação `[início:comprimento]` ou `[início-fim]`
- Comparações tipo C / em inglês
  - `eq` , `==`
  - `ne` , `!=`
  - `gt` , `>`
  - `ge` , `>=`
  - `lt` , `<`
  - `le` , `<=`
- Agrupamento com parêntesis
- Combinação com operadores lógicos tipo C / em inglês
  - `and` , `&&`
  - `or` , `||`
  - `xor` , `^^`
  - `not` , `!`
- Operadores de pesquisa `contains` e `matches`

Exemplos de filtros de visualização

- Pedidos HTTP GET
  `http.request.method == "GET"`
- Pacotes destinados a um dado endereço IP
  `ip.dst == 10.1.2.3`
- Tráfego de/para um dado endereço IP
  `ip.addr == 10.1.2.3`
- Trafego que não venha nem vá para um dado endereço IP
  `!(ip.addr == 10.1.2.3)`

  CUIDADO: não usar `ip.addr != 10.1.2.3` para este fim, pois significa `ip.src != 10.1.2.3 or ip dst != 10.1.2.3`
- Excluir pacotes IPv4 internos a uma dada sub-rede
  `ip.addr != 172.16.1.0/24`
- Tráfego IPv4 interno a uma dada sub-rede
  `ip and !(ip.addr != 172.16.1.0/24)`

Exemplos de filtros de visualização

- Pedidos de estabelecimento de conexão TCP (SYN sem ACK)
  `tcp.flags.syn==1 and tcp.flags.ack==0`
- Pacotes ARP e ICMP
  `arp || icmp`
- Tramas contendo a sequência de bytes 61:62:63 (hex) algures
  `frame contains 61:62:63`
- Pedidos HTTP para área do utilizador "user" (com regexp)
  `http.request.uri matches "^/~user"`
- Tráfego de/para máquinas cujo último byte do endereço IP é 1
  `ip.addr matches "\\.1$"`
  - Necessário duplo escape para o '.'
- Apenas tramas marcadas (com Ctrl-M)
  `frame.marked==1`