# Conteúdo


**1 Camada de Aplicação** **3**
- 1.1 Visão Geral . . . 3
  - 1.1.1 Arquiteturas . . . 3
  - 1.1.2 Ligação de Processos . . . 4
- 1.2 Protocolos Gerais . . . 4
  - 1.2.1 HTTP . . . 4
  - 1.2.2 FTP . . . 5
- 1.3 Correio Eletrónico . . . 5
  - 1.3.1 SMTP . . . 5
  - 1.3.2 Protocolos de acesso ao correio . . . 5
- 1.4 DNS . . . 6
  - 1.4.1 Serviços . . . 6
  - 1.4.2 Estrutura . . . 6
  - 1.4.3 Servidores DNS . . . 6
  - 1.4.4 Resolução de nomes . . . 7
  - 1.4.5 Cache e atualização de registos . . . 7
  - 1.4.6 Registos DNS . . . 7
- 1.5 Sumário . . . 7

**2 Camada de Transporte** **8**
- 2.1 Multiplexagem e Desmultiplexagem . . . 8
  - 2.1.1 Multiplexagem . . . 8
  - 2.1.2 Desmultiplexagem . . . 8
- 2.2 Protocolos . . . 9
  - 2.2.1 UDP . . . 9
  - 2.2.2 Princípios de Transporte Fiável . . . 10
  - 2.2.3 TCP . . . 11
- 2.3 Controlo de Congestão . . . 13
  - 2.3.1 Fim-a-Fim . . . 13
  - 2.3.2 Auxílio da Rede . . . 13
  - 2.3.3 Controlo de Congestão TCP . . . 13

**3 Camada de Rede** **15**
- 3.1 Funções Gerais . . . 15
  - 3.1.1 Routing . . . 15
  - 3.1.2 Forwarding . . . 16
- 3.2 Plano de dados . . . 16
  - 3.2.1 Anatomia do Router . . . 16

3.2.2 Endereçamento IP . . . . . 18
3.2.3 Encaminhamento Generalizado (SDN) . . . . . 19
3.3 Plano de Controlo . . . . . 19
3.3.1 Algoritmos de Routing . . . . . 19
3.3.2 Algoritmo Estado de Ligação (Link State) . . . . . 20
3.3.3 Vetor distância (Distance Vector) . . . . . 20
3.3.4 Comparação entre Algoritmos . . . . . 21
3.3.5 Routing Hierárquico . . . . . 22
3.4 Broadcast e Multicast . . . . . 22
3.4.1 Routing Broadcast . . . . . 22
3.4.2 Routing Multicast . . . . . 23

**4 Camada de Ligação de Dados** **26**
4.1 Visão Geral . . . . . 26
4.1.1 Serviços . . . . . 26
4.1.2 Implementação . . . . . 27
4.2 Protocolos de Acesso ao Meio . . . . . 27
4.2.1 Tipos de Ligações . . . . . 27
4.2.2 Protocolos de Acesso Múltiplo . . . . . 27
4.2.3 Multiple Access Control (MAC) . . . . . 28
4.2.4 Particionamento do Canal . . . . . 28
4.2.5 Acesso Aleatório . . . . . 28
4.2.6 Acesso Ordenado . . . . . 29
4.3 Redes LAN . . . . . 29
4.3.1 Endereço MAC . . . . . 29
4.3.2 Address Resolution Protocol (ARP) . . . . . 30
4.3.3 Ethernet . . . . . 30
4.3.4 Switches e Routers . . . . . 31
4.3.5 Virtual Local Area Network (VLAN) . . . . . 32
4.4 Redes em Data Centers . . . . . 33

**5 Outros Tópicos** **34**
5.1 Redes Sem Fios . . . . . 34
5.1.1 Elementos de uma Rede sem Fios . . . . . 34
5.1.2 Redes WiFi . . . . . 35
5.2 Segurança de Redes . . . . . 36

# Capítulo 1

# Camada de Aplicação

O objetivo da camada de aplicação é proporcionar um meio de comunicação entre processos.

## Conteúdos


**1.1 Visão Geral . . . . . 3**
1.1.1 Arquiteturas . . . . . 3
1.1.2 Ligação de Processos . . . . . 4
**1.2 Protocolos Gerais . . . . . 4**
1.2.1 HTTP . . . . . 4
1.2.2 FTP . . . . . 5
**1.3 Correio Eletrónico . . . . . 5**
1.3.1 SMTP . . . . . 5
1.3.2 Protocolos de acesso ao correio . . . . . 5
**1.4 DNS . . . . . 6**
1.4.1 Serviços . . . . . 6
1.4.2 Estrutura . . . . . 6
1.4.3 Servidores DNS . . . . . 6
1.4.4 Resolução de nomes . . . . . 7
1.4.5 Cache e atualização de registos . . . . . 7
1.4.6 Registos DNS . . . . . 7
**1.5 Sumário . . . . . 7**


## 1.1 Visão Geral

### 1.1.1 Arquiteturas

#### Cliente-Servidor

Na arquitetura cliente-servidor o servidor é um terminal sempre ligado com endereço IP permanente que serve o propósito de atender os pedidos dos clientes que, por contraste, não estão sempre ligados, têm endereço IP dinâmico e não comunicam entre si.

#### Peer-to-Peer (P2P)

Na arquitetura P2P não existe um servidor sempre ligado e os clientes comunicam diretamente entre si. Trata-se de uma arquitetura de gestão complexa, dado que os seus endereços IP são dinâmicos e auto-escalável pois os peers cobrem ambas as funções.

### 1.1.2 Ligação de Processos

Para receberem mensagens os processos têm de ter um identificador, que é composto pelo endereço IP da máquina e pelo número de porto.

#### Socket

A socket é um conceito do sistema operativo por onde os processos enviam e recebem mensagens.

## 1.2 Protocolos Gerais

### 1.2.1 HTTP

O protocolo HTTP segue o modelo cliente-servidor, usando o porto 80 e utiliza TCP como protocolo de transporte. No HTTP apenas um objeto é enviado através de uma mensagem e diz-se um protocolo do tipo pull (o cliente puxa o objeto do servidor).

#### HTTP não persistente

No modo não persistente cada ligação serve para transmitir um objeto e depois é fechada

#### HTTP persistente

No modo persistente o servidor mantém a ligação aberta depois de enviar uma resposta.

#### Cookies

O protocolo HTTP é stateless. O servidor não guarda nenhuma informação sobre pedidos anteriores do cliente. Para compensar este facto, o HTTP utiliza cookies. Através dos cookies os clientes passam a guardar estado, em vez dos servidores, ou seja, a aplicação que corre por cima do HTTP é stateful.

#### Web Caching

O HTTP usa web caching para satisfazer pedidos do cliente sem envolver o servidor de origem. O browser envia os pedidos para a cache, se estiver armazenado retorna o objeto, caso contrario pede o objeto ao servidor de origem.

#### GET Condicional

É possível especificar um pedido HTTP como sendo condicional. O objetivo é não enviar o objeto se a cache tem uma versão atualizado. Este tipo de pedido utiliza a clausula *if-modified-since*: com a data do objeto guardado em cache.

### 1.2.2 FTP

O protocolo FTP tem como objetivo transferir ficheiros de e para terminais remotos. Segue o modelo cliente-servidor, utiliza o porto 21 e é stateful, guardando a diretoria atual, etc.

#### Funcionamento

Quando um cliente contacta um servidor FTP no porto 21, usando TCP, é aberta uma ligação de controlo, que permite pedidos de autorização, pesquisa de diretoria, etc. Esta ligação de controlo diz-se out-of-band. Ao receber um comando de transferência de ficheiro é aberta uma ligação de dados, cuja única função é transferir o ficheiro. Esta ligação de dados diz-se in-band.

## 1.3 Correio Eletrónico

Os servidores de correio armazenam as mensagens destinadas ao utilizador. As mensagens que aguardam envio ficam numa fila de espera.

### 1.3.1 SMTP

O protocolo SMTP é utilizado na comunicação entre servidores de correio para enviar mensagens. Também é utilizado em comunicação cliente -> servidor, mas não sevidor -> cliente. Este protocolo utiliza TCP e o porto 25. A transferência é direta do servidor emissor para o servidor destino em 3 fases:

- Handshaking
- Transferência de mensagens
- Fecho

SMTP apenas utiliza ligações persistentes, vários objetos são enviados através de uma única mensagem e diz-se um protocolo do tipo push (o servidor empurra a mensagem para o destino).

### 1.3.2 Protocolos de acesso ao correio

São necessários protocolos adicionais para aceder ao correio, já que o SMTP é do tipo push.

#### POP3 (Post Office Protocol Version 3)

Protocolo simples, stateless

#### IMAP (Internet Mail access Protocol)

Protocolo com mais funcionalidades, stateful

#### HTTP

Usado no gmail, hotmail...

## 1.4 DNS

O DNS surge como resposta ao problema de como mapear entre um nome e um endereço IP. Trata-se de uma base de dados distribuída em que estão contidas correspondências entre nomes e endereços IP. Este protocolo usa o porto 53 e tipicamente utiliza UDP. Os servidores DNS são máquinas UNIX e correm o software BIND.

### 1.4.1 Serviços

- Traduções (resolução) de endereços
- Múltiplos nomes (host aliasing)
- Nomes de servidores de correio
- Distribuição de carga

### 1.4.2 Estrutura

O DNS não é centralizado pois tornava-se num ponto único de falha, com todo o tráfego concentrado nesse ponto e distante da maior parte dos clientes.
A solução é distribuir o serviço pelo mundo.

### 1.4.3 Servidores DNS

#### Servidores de DNS local

Não pertencem, estritamente, á hierarquia. Cada ISP tem pelo menos um servidor de nomes local. Possui uma cache com os pares nome/endereço mais recente. Quando uma máquina faz uma query ao DNS, este é enviado ao servidor de nomes local.

#### Servidores de Raíz

São contactados pelo servidor de nomes local quando este não sabe resolver um nome. Há 13 root servers no mundo.

#### Servidores TLD

São responsáveis pelos domínios com, org, net, edu e pelos domínios top level de países (ex: pt, es...).

#### Servidores DNS autoritário

Servidores DNS da organização responsável pelo domínio.

### 1.4.4 Resolução de nomes

#### Iterativa

O servidor contactado responde com o nome do servidor a contactar.

#### Recursiva

A responsabilidade da resolução de nomes passa para o servidor contactado.

### 1.4.5 Cache e atualização de registos

Sempre que um servidor de nome aprende uma nova tradução armazena-a numa cache, melhorando o desempenho. Estas entradas são removidas após um período de tempo. As entradas na cache podem estar desatualizadas.

### 1.4.6 Registos DNS

*DNS General resource record format*

| Name | TTL | Class | Type | Rdata |
|---|---|---|---|---|

**Type = A** (address)
- **Name** é o *hostname*
- **Rdata** é o endereço IPv4

**Type = CNAME** (canonical name)
- **Name** é um nome alternativo para outro nome
- `www.ibm.com` é, na verdade, `servereast.backup2.ibm.com`
- **Rdata** pode ser o nome oficial

**Type = NS** (name server)
- **Name** é o domínio (*e.g.* ul.pt)
- **Rdata** é o *hostname* do servidor autoritário para o domínio

**Type = MX** (mail server)
- **Rdata** contém o nome do servidor de mail associado a **Name**

## 1.5 Sumário

| | Requisitos | Protocolo de transporte | Estado | Modelo | Ligação de controlo | Tipo de operação |
|---|---|---|---|---|---|---|
| **HTTP** | Fiabilidade | TCP | Stateless | Cliente-servidor | in-band | pull |
| **FTP** | Fiabilidade | TCP | Stateful | Cliente-servidor | out-of-band | pull |
| **POP3** | Fiabilidade | TCP | Stateless | Cliente-servidor | in-band | pull |
| **IMAP** | Fiabilidade | TCP | Stateful | Cliente-servidor | in-band | pull |
| **SMTP** | Fiabilidade | TCP | Stateful | Cliente-servidor | in-band | push |
| **DNS** | Fiabilidade | UDP/TCP | Stateless | Cliente-servidor | in-band | pull |

# Capítulo 2

# Camada de Transporte

O objetivo da camada de transporte é oferecer um canal lógico de comunicação entre processos remotos. Os protocolos de transporte correm apenas nos computadores terminais e não na infrastrutura da rede (routers e switches).

## Conteúdos


**2.1 Multiplexagem e Desmultiplexagem . . . . . 8**
2.1.1 Multiplexagem . . . . . 8
2.1.2 Desmultiplexagem . . . . . 8
**2.2 Protocolos . . . . . 9**
2.2.1 UDP . . . . . 9
2.2.2 Princípios de Transporte Fiável . . . . . 10
2.2.3 TCP . . . . . 11
**2.3 Controlo de Congestão . . . . . 13**
2.3.1 Fim-a-Fim . . . . . 13
2.3.2 Auxílio da Rede . . . . . 13
2.3.3 Controlo de Congestão TCP . . . . . 13


## 2.1 Multiplexagem e Desmultiplexagem

Estes conceitos surgem da necessidade de distinguir processos remotos.

### 2.1.1 Multiplexagem

A multiplexagem recolhe dados de vários sockets e adiciona cabeçalho de transporte com informação sobre origem.

### 2.1.2 Desmultiplexagem

Na desmultiplexagem a informação no cabeçalho usada para entregar os segmentos ao socket correto.

#### UDP

O serviço de transporte UDP não requer ligação. Quando se cria um socket este vai receber um número de porto local. Ao receber um datagrama verifica o porto destino e direciona o segmento UDP para o socket com esse número de porto, logo datagramas IP com o mesmo porto destino, mas diferentes IP ou porto origem são direcionados para o mesmo socket no destino

#### TCP

Um socket TCP é identificado pelo tuplo <Endereço IP origem, porto origem, IP destino, porto destino>, logo o servidor pode suportar múltiplos sockets TCP para o mesmo porto destino

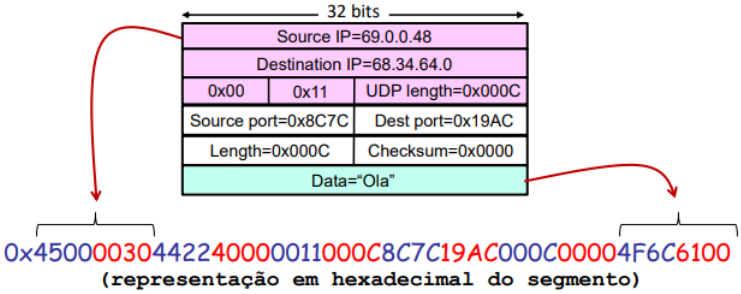
## 2.2 Protocolos

### 2.2.1 UDP

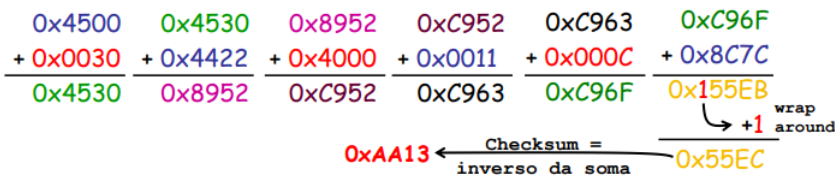
UDP é o protocolo de transporte mais simples. É um protocolo connectionless, sem handshaking e cada segmento é tratado como independente. Oferece um serviço best effort, sendo que os seus segmentos podem perder-se ou ser entregues à aplicação fora de ordem. Para conseguir fiabilidade, esta adiciona-se ao nível da camada de aplicação.

#### Checksum

Para ajudar a detetar erros utiliza-se o checksum. O checksum consiste na adição em complemento de 1 dos conteúdos do segmento ao ser enviado. Este valor é colocado no segmento e ao chegar ao destino, este calcula o checksum do segmento recebido da mesma forma e compara com aquele que foi atribuído inicialmente.




- **Soma do conteúdo em blocos de 16 bits (adição complemento de 1) :**



### 2.2.2 Princípios de Transporte Fiável

#### Deteção e Recuperação de Erros

O canal de comunicação pode trocar alguns bits do pacote (bit flipping). O checksum é usado para detetar estes erros.
Para recuperar estes erros utilizam-se ACKs (o recetor comunica explicitamente ao emissor que recebeu bem o pacote) e NACKs (o recetor comunica explicitamente ao emissor que o pacote tinha erros ou foi perdido). Em caso de NACK ou de não receber ACK após um determinado período de tempo o pacote é retransmitido.

#### Números de sequência

Pacotes incluem um número de sequência.

#### Protocolo Stop-and-Wait

Emissor envia um pacote, pára e espera pela resposta do recetor. Ao receber a resposta envia o próximo pacote. Após um tempo de timeout, se não tiver obtido resposta, reenvia o pacote.

#### Pipelining

O emissor permite que haja múltiplos pacotes in-flight (pacotes enviados mas dos quais ainda não recebeu ACK). É necessário armazenar estes pacotes em buffers no emissor e, por vezes, no destinatário mas possibilita taxa de transferência efetiva $N$ vezes melhor, com $N$ pacotes in-flight. Dois protocolos com pipelining são o Selective Repeat e o Go-Back-N (O TCP utiliza um híbrido destes dois).

#### Go-Back-N

Emissor pode ter até $N$ pacotes in-flight. O destinatário envia ACKs cumulativos. O ACK recebido é sempre do último pacote recebido na ordem correta, ignorando pacotes recentes bem recebidos mas fora de ordem. Destinatário não guarda em buffer segmentos fora de ordem.
O emissor mantém temporizador apenas para segmento mais antigo sem ACK. Se o temporizador expirar retransmite todos os pacotes seguintes.

#### Selective Repeat

Emissor pode ter até $N$ pacotes in-flight. O destinatário envia ACKs individualizados (um por pacote recebido) e coloca num buffer os pacotes fora de ordem. O emissor mantém um temporizador para cada pacote sem ACK. Quando o temporizador expirar, retransmite-se apenas aquele pacote.

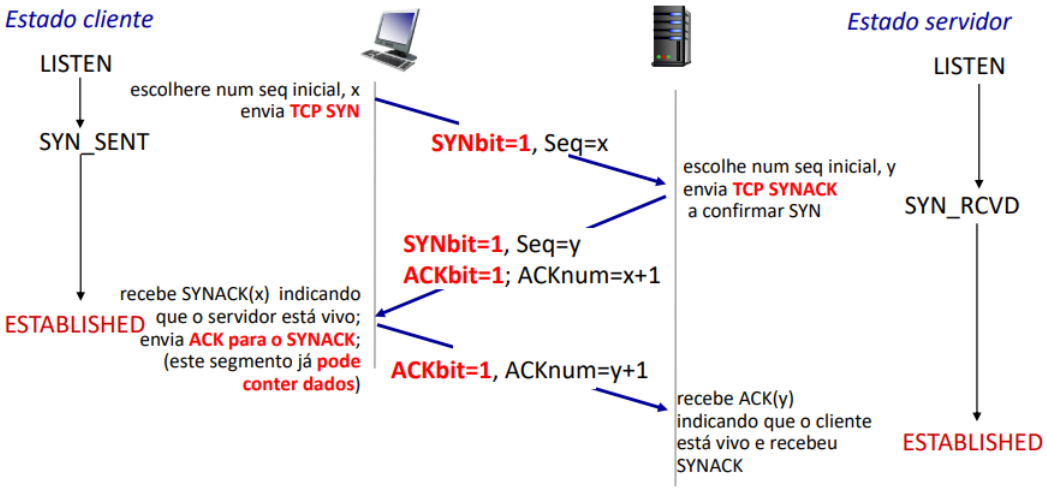
**Sumário**

| Mecanismo | Utilidade |
|---|---|
| Checksum | Detetar erros de bit |
| ACK | Avisa emissor de que o pacote foi corretamente recebido |
| NACK | Avisa emissor de que o pacote foi não foi recebido corretamente |
| Temporizador (timer) | Usado para retransmitir pacote quando ele (ou ACK) se perde |
| Números de sequência | Usados para detetar pacotes em falta ou duplicados |
| Janela, pipelining | Permite enviar múltiplos pacotes sem receber confirmação, aumentando a utilização da rede e portanto a taxa efetiva de transferência de dados |

### 2.2.3 TCP

TCP tem dados em full duplex, é ponto-a-ponto (1:1), tem controlo de fluxo e de congestão (para não entupir o destinatário e a rede, respetivamente), utiliza um protocolo pipelined e assegura uma entrega fiável e ordenada de streams de bytes.

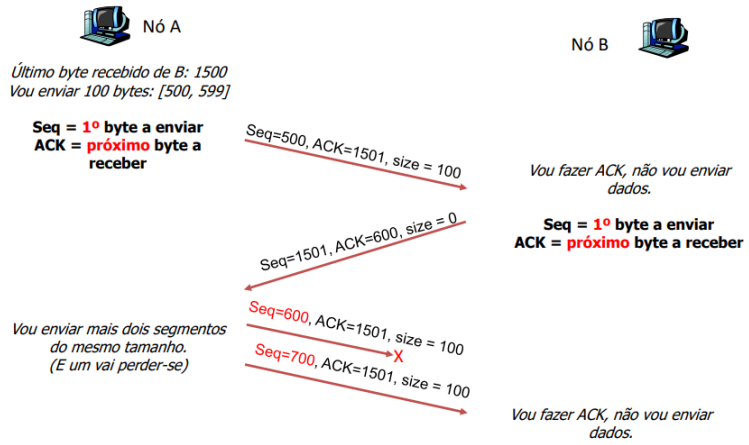
**Estabelecimento de Ligação TCP**




**Números de Sequência e ACKs**

No TCP usa-se como número de sequência o primeiro byte do segmento. O ACK é cumulativo e diz qual o próximo byte esperado.
O TCP não especifica como tratar segmentos que chegam fora de ordem, logo

pode-se guardar em buffer (como selective repeat), ou pode descartar segmento (como go-back-n).




### Timeout e RTT

Timeout tem de ser maior que o RTT. Para estimar o RTT utiliza-se a $AmostraRTT$ (média de várias medidas recentemente) em:

$$RTT_{Estimado} = (1 - \alpha) * RTT_{Estimado} + \alpha * AmostraRTT$$

Tipicamente $\alpha = 0.125$. O timeout é o RTT estimado mais uma margem de segurança:

$$Timeout = RTT_{Estimado} + 4 * DesvioRTT$$

### Fast Retransmit

O período de timeout é normalmente longo, o que causa um atraso grande. Se apenas um segmento se perder é natural que se recebam múltiplos ACKs duplicados, se o emissor receber 3 ACKs duplicados é muito provável que o segmento tenha sido perdido, não vale a pena esperar pelo timeout e reenvia-se o segmento mais antigo sem ACK.

### Estabelecimento e fecho de ligação TCP

Antes de trocarem dados, o emissor e recetor fazem um handshake e no final cliente e o servidor têm de fechar cada um dos lados da ligação.

### Controlo de fluxo TCP

Em TCP o recetor publicita no cabeçalho o espaço livre no buffer: o valor rwnd (receive window). O emissor limita o número de pacotes in flight aos valor rwnd, garantindo que o buffer não vai encher.

| | Pipelined | ACK | Timer emissor | Retransmissão | Buffer "out-of-order" |
|---|---|---|---|---|---|
| **Stop & wait** | Não | Individualizado | 1 | ACK | Não |
| **Go-back-N** | Sim | Cumulativo | 1 | ACK + seguintes | Não |
| **Selective repeat** | Sim | Individualizado | 1 por pacote | ACK | Sim |
| **TCP** | Sim | Cumulativo | 1 | ACK | Depende |

**Sumário**

## 2.3 Controlo de Congestão

O controlo de fluxo previne entupir o recetor; o controlo de congestão previne entupir a rede. Este problema manifesta-se na perda e atraso de pacotes. Existem duas abordagens ao controlo de congestão

### 2.3.1 Fim-a-Fim

A rede não envia feedback explícito a indicar congestão, esta é inferida pelo computador terminal, quando observa perda de pacotes ou atrasos.

### 2.3.2 Auxílio da Rede

Os routers enviam explicitamente feedback aos computadores terminais com informação no cabeçalho dos pacotes a indicar que há congestão. Ou definindo a taxa máxima a que o emissor pode enviar dados, de forma explícita

### 2.3.3 Controlo de Congestão TCP

Existem duas formas para um emissor detetar que há congestão: Indiretamente, através da perda de pacotes ou detetados por timeout ou através de 3 ACKs duplicados. Um emissor pode controlar a congestão limitando a sua janela de congestão (cwnd = número de segmentos in flight permitidos), assim limitando a taxa de transferência:

$$Tamanho_{janela} = min(rwnd, cwnd)$$

O TCP implementa um algoritmo de controlo de congestão que opera em 3 fases:

#### Slow Start

Começa lento, com $cwnd = 1$ e aumenta exponencialmente até primeira perda de pacote ($cwnd+ = 1$ com cada ACK, i.e. $cwnd$ duplica com cada RTT).

#### Congestion Avoidance

Mecanismo AIMD (Additive Increase,Multiplicative Decrease):

Aumento da janela é aditivo (conservador);

Redução é multiplicativa (radical)

$$cwnd = cwnd/2$$

(ou 1) quando há perda de pacote.

**Recuperação**

Quando a perda é detetada por timeout $cwnd = 1$; slow start até ssthresh; congestion avoidance depois.

Quando a perda detetada através de 3 ACKs duplicados existem duas abordagens. No TCP Reno a abordagem é mais conservadora, $cwnd = cwnd/2$. O TCP Tahoe é mais radical, $cwnd = 1$.

# Capítulo 3

# Camada de Rede

A camada de rede tem como objetivo o transporte de datagramas entre computadores. Os protocolos desta camada correm nos nós terminais e nos routers, que examinam os cabeçalhos IP de todos os datagramas.

## Conteúdos


**3.1 Funções Gerais . . . . . 15**
3.1.1 Routing . . . . . 15
3.1.2 Forwarding . . . . . 16
**3.2 Plano de dados . . . . . 16**
3.2.1 Anatomia do Router . . . . . 16
3.2.2 Endereçamento IP . . . . . 18
3.2.3 Encaminhamento Generalizado (SDN) . . . . . 19
**3.3 Plano de Controlo . . . . . 19**
3.3.1 Algoritmos de Routing . . . . . 19
3.3.2 Algoritmo Estado de Ligação (Link State) . . . . . 20
3.3.3 Vetor distância (Distance Vector) . . . . . 20
3.3.4 Comparação entre Algoritmos . . . . . 21
3.3.5 Routing Hierárquico . . . . . 22
**3.4 Broadcast e Multicast . . . . . 22**
3.4.1 Routing Broadcast . . . . . 22
3.4.2 Routing Multicast . . . . . 23


## 3.1 Funções Gerais

### 3.1.1 Routing

Determina o cominho origem-destino. Esta função é assegurada no plano de controlo por um algoritmo de routing, que determina a rota através da rede. Equivalente a planear uma viagem. Existem duas abordagens:

#### Tradicional

Na abordagem tradicional os algoritmos de routing estão implementados nos routers. Componentes individuais do algoritmo correm em cada router e todos interagem entre si.

#### Redes Definidas por Software (SDN)

Na abordagem SDN estes algoritmos estão implementados em servidores remotos, que interagem com os agentes locais em cada router.

### 3.1.2 Forwarding

Mover pacotes que chegam para a saída apropriada. Esta função é assegurada no plano de dados pelas tabelas de forwarding locais de cada router que comparam bits do cabeçalho de cada datagrama e o direcionam para a saída correspondente. Equivalente a passar um cruzamento.

## 3.2 Plano de dados

### 3.2.1 Anatomia do Router

Um router é um computador com características especificas, nomeadamente o facto de ter várias placas de rede. O seu objetivo é a comutação de pacotes e inclui um módulo de comutação de alta velocidade. As suas funções chave são correr protocolos de routing (tradicionais ou SDN) e encaminhar datagramas da entrada para a saída.

#### Porta de Entrada

Dado o endereço de destino do datagrama, faz-se a sua procura na tabela de encaminhamento e este é enviado para a saída correta. O objetivo é que o processamento seja feito á velocidade da linha, caso contrário criam-se filas/buffer, isto é, os datagramas estão a chegar mais rápido que a velocidade do módulo de switching.

#### Tabela de Encaminhamento

A tabela de encaminhamento está guardada em memória na porta de entrada. Como existem demasiados endereços IP para serem listados individualmente as entradas são agregadas na tabela.

#### Notação CIDR

Nas tabelas de encaminhamento utiliza-se a notação CIDR com Longest Prefix Matching, ou seja, a tabela tem o seguinte aspeto:

| Gamas de endereços IP | Interface ligação |
|---|---|
| 10000000 00010000 00010*** ********* | 0 |
| 10000000 00010000 00011000 ********* | 1 |
| 10000000 00010000 00011*** ********* | 2 |
| Caso contrário | 3 |

Que equivale a:

| Gamas de endereços IP | Interface ligação |
|---|---|
| 128.16.16.0/21 | 0 |
| 128.16.24.0/24 | 1 |
| 128.16.24.0/21 | 2 |
| 0.0.0.0/0 | 3 |

Neste exemplo, para a interface 0 seguem os pacotes cujo endereço IP, em binário seja idêntico aos primeiros 21 bits do endereço 128.16.16.0. No caso das interfaces 1 e 2, ambas usam o mesmo IP como referência , no entanto a interface 1 considera 24 bits deste endereço e a 2 apenas 21. Como se utiliza Longest Prefix Matching, endereços que correspondam a todos os primeiros 24 bits do endereço seguem para a interface 1. Endereços que correspondam a pelo menos 21, mas menos que 24 seguem na 2.

### Módulo de comutação

O módulo de comutação transfere os pacotes do buffer de entrada para o buffer de saída. A taxa de comutação é a taxa a que os pacotes são enviados da entrada para a saída; para N entradas a taxa deveria ser N vezes o valor da velocidade da linha. Existem três tipos de módulos de comutação: memória, bus (barramento) e rede de interligação (crossbar).

Na primeira opção os pacotes são copiados para a memória do sistema; a velocidade está limitada pela sua largura de banda (cada datagrama passa duas vezes).

Com um bus a taxa de comutação é limitada pela largura de banda do barramento.

Por fim, a rede de interligação ultrapassa as limitações do barramento, já que várias ligações podem usar a rede ao mesmo tempo. Tipicamente os datagramas são fragmentados para facilitar a comutação.

### Portas de saída

Nas portas de saída pode ser necessário buffering quando o módulo de comutação é mais rápido que a linha. É utilizado uma algoritmo de escalonamento para decidir qual o próximo pacote a transmitir.

### 3.2.2 Endereçamento IP

Um endereço IPv4 é um identificador de 32 bits para uma interface de rede. No endereço, os bits mais significativos identificam a subrede e os bits menos significativos identificam o computador/router. Uma subrede define-se como sendo um conjunto de dispositivos que conseguem comunicar entre si sem necessitar de um router. O endereço IP de um computador pode ser atribuído manualmente ou automaticamente.

#### Dynamic Host Configuration Protocol (DHCP)

O DHCP permite que um computador obtenha dinamicamente uma endereço IP a partir de um servidor da rede, permitindo assim a reutilização de endereços e facilitando a entrada de utilizadores móveis na rede. A atribuição ocorre em 4 fases:

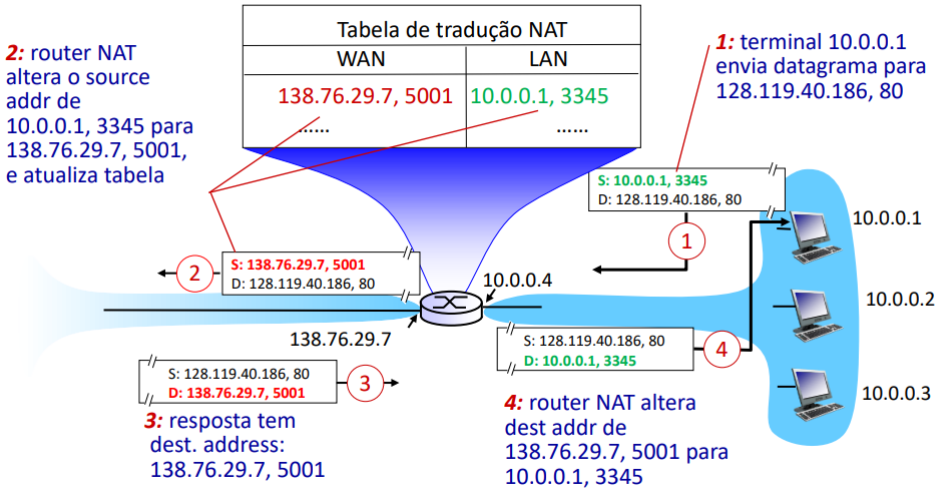
- O computador faz broadcast do DHCP Discover
- O(s) servidor DHCP propõe um IP com o DHCP Offer
- O computador pede o IP com o DHCP Request
- O servidor DHCP confirma com o DHCP ack

O servidor retorna ainda mais informação, como o endereço do router mais próximo, nome e IP do servidor DNS, máscara de rede, etc.

O servidor DHCP escolhe o IP a partir do espaço de endereçamento do ISP.

#### Network Address Translation (NAT)

O NAT é necessário pois para a rede, a rede local usa apenas 1 endereço IP (para permitir uma maior flexibilidade na rede local). O funcionamento do NAT é simples:



O NAT usa os 16 bits do porto, permitindo $2^{16}$ ligações simultâneas com um único endereço

#### Problema NAT transversal

O problema do NAT transversal surge pois um cliente pode querer contactar um servidor utilizando o seu endereço local. Existem três soluções para este problema:

- Configurar o NAT estaticamente
- Usar um dispositivo Universal Plug and Play (UPnP) e o protocolo Internet Gateway Device (IGD), automatizando o processo anterior
- Utilizando relaying. Os clientes ligam-se a um intermediário que faz a passagem de pacotes entre ligações

#### ICMP

O ICMP é usado pelos computadores para trocarem informações sobre a camada de rede. A ferramenta traceroute usa ICMP para fornecer informação sobre todo o percurso desde ao emissor até ao destino.

#### IPv6

Devido ás limitações do IPv4 adotou-se o IPv6, desta vez com 128 bits e com o campo checksum removido. Durante o período de transição os routers utilizam pilhas protocolares duplas para conseguirem operar com ambos os tipos de endereço simultaneamente e túneis para que nos routers IPv4 os datagramas IPv6 sejam enviados como carga de um datagrama IPv4.

### 3.2.3 Encaminhamento Generalizado (SDN)

Em SDN o conceito de encaminhamento é generalizado utilizando uma nova abstração, o flow. Um flow é definido pelos campos de cabeçalho:

- Pode ser IP destino como tradicionalmente: $< dst\ IP >$
- Pode ser IP emissor + IP destino: $< src\ IP; dst\ IP >$
- Pode ser o 5-tuple que representa uma ligação TCP:

$$< src\ port; src\ IP; dst\ port; dst\ IP; protocol >$$

O principal protocolo SDN é o OpenFlow.

## 3.3 Plano de Controlo

### 3.3.1 Algoritmos de Routing

O objetivo de um algoritmo de routing é determinar qual o melhor caminho a percorrer no grafo da rede. Algoritmos de routing classificam-se quando á globalidade da sua informação:

- Nos algoritmos globais os routers têm noção da topologia completa da rede distribuem informação local com todos (Ex. link state)
- Nos algoritmos com informação parcial os routers só conhecem os nós vizinhos e distribuem informação global (de toda a rede) apenas com vizinhos (Ex. distance vector)

e como sendo estáticos ou dinâmicos

- Nos algoritmos estáticos as rotas mudam muito lentamente e as tabelas de encaminhamento podem ser configurados manualmente
- Nos algoritmos dinâmicos as rotas mudam rapidamente e são feitas atualizações periódicas em resposta a mudanças

### 3.3.2 Algoritmo Estado de Ligação (Link State)

O algoritmo estado de ligação mais comum é o algoritmo de Dijkstra. Neste algoritmo a topologia da rede e os custos da ligação são conhecidos por todos os nós através do link state broadcast (broadcast com informação sobre as ligações do nó). Cada nó calcula os percursos de custo mínimo entre si e todos os outros e recalcula periodicamente ou quando alguma ligação sofre alteração.

```
#0 Entrada
   - nó_a_testar: TODOS
   - nó_resolvido: NENHUM

#1 Encontrar o nó ao qual eu
  chego com custo mínimo

#2 Adicionar esse nó a
  nó_resolvido e remover de
  nó_a_testar.

#3 Repetir #1 e #2 até
  nó_a_testar estar vazio
```

### 3.3.3 Vetor distância (Distance Vector)

#### Equação de Bellman-Ford

Seja $d_x(y)$ o custo do percurso de custo mínimo entre x e y, então

$$d_x(y) = min\ \{c(x,v) + d_v(y)\}$$

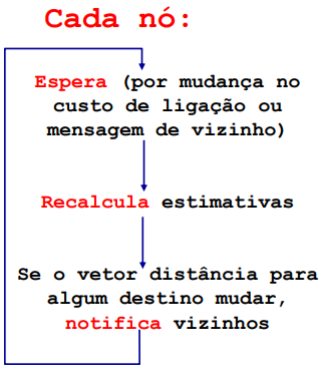
em que $c(x,v)$ é o custo para vizinho $v$ e $d_v(y)$ é o custo do vizinho $v$ para o destino $y$.

#### Algoritmo Vetor Distância

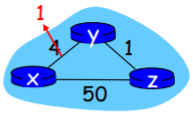
Neste algoritmo $D_x(y)$ representa uma estimativa do custo mínimo entre $x$ e $y$. O nó $x$ conhece o custo para cada vizinho $v$: $c(x,v)$ e guarda os vetores distância dos vizinhos. Para cada vizinho $v$, guarda: $D_v = [D_v(y) : y \in \mathbb{N}]$

A ideia chave é que cada nó envia a sua estimativa de vetor distância aos vizinhos periodicamente, quando recebe um novo vetor distância atualiza o seu usando a equação de Bellman-Ford.

Este algoritmo é iterativo e assíncrono, sendo cada iteração local causada pela mudança de custo de uma das ligações locais ou pela receção de um novo vetor distância de um vizinho. É também um Algoritmo distribuído pois cada nó apenas notifica os seus vizinhos quando o seu vetor distância sofre alteração.




Quando um nó deteta mudança local este atualiza informação e recalcula vetor distância se o vetor sofrer alteração, notifica vizinhos e alteração propaga-se rapidamente:

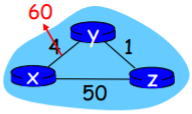



*#1: y* deteta mudança, atualiza VD, informa vizinhos.

*#2: z* recebe atualização de *y*, atualiza, calcula novo custo mínimo para *x*, envia aos seus vizinhos.

*#3: y* recebe atualização de *z*, atualiza a sua tabela. Como o custo não sofre alteração, pára.

Se o custo aumentar temos o problema count-to-infinity e a alteração propaga-se devagar:




Neste caso, *y* deteta mudança e atualiza VD com **distância para x = 60**, informa vizinhos.

*PROBLEMA: nó z tem custo para x=5 através do y! Mas o y não sabe pois não tem visão global da rede!*

**Criou-se um loop (ciclo) na rede!**

$d_x(y) = \min \{c(x,v) + d_v(y)\}$

### 3.3.4 Comparação entre Algoritmos

Tipo de computação:

- Link state: centralizada
- Distance vector: distribuída

Complexidade das mensagens:

- Link state: com $n$ nós e $E$ ligações, $O(nE)$ mensagens enviadas (informação local trocada globalmente)
- Distance vector: troca de mensagens apenas entre vizinhos (informação global trocada localmente)

Tempo de convergência:

- Link state: algoritmo $O(n^2)$ sendo possível reduzir para $O(n \log n)$
- Distance vector: tempo de convergência varia pois pode haver loops ou o problema count-to-infinity

### 3.3.5 Routing Hierárquico

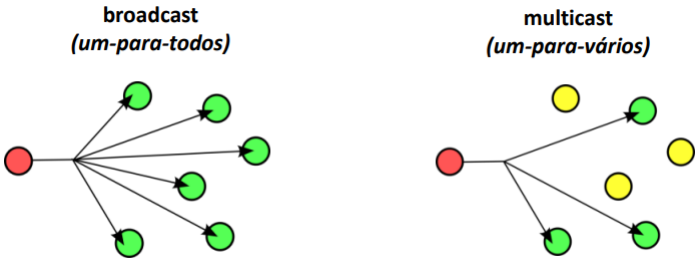
Até agora apenas se considerou um cenário ideal, o que não corresponde á verdade. O routing na Internet é hierárquico, com os routers agregados em regiões, os sistemas autónomos (autonomous systems, AS). Routers do mesmo AS correm o mesmo protocolo de routing intra-AS, mas routers de outro AS podem correr protocolos distintos.

#### Interligação entre AS

As diferentes redes são ligadas através de routers na fronteira do seu AS, que correm o protocolo de routing inter-AS. No caso da Internet, este protocolo é o BGP (Border Gateway Protocol). A tabela de encaminhamento é configurada pelos dois protocolos: intra-AS e inter-AS.

- O intra-AS define as entradas relacionadas com os destinos internos
- O inter-AS (com ajuda do intra-AS) define as entradas relacionadas com os destinos externos

## 3.4 Broadcast e Multicast




### 3.4.1 Routing Broadcast

No routing broadcast os pacotes são enviados do emissor para todos os outros nós, o que é muito ineficiente devido á duplicação, tanto na fonte como na rede.

#### Flooding

Flooding acontece quando um nó recebe um pacote broadcast e envia uma cópia para todos os vizinhos. Isto pode criar ciclos e broadcast storms.

#### Flooding Controlado

O flooding controlado acontece quando um nó faz broadcast do pacote apenas se não o tiver feito anteriormente. Para isto, o nó guarda o ID de todos os pacotes que já fez broadcast, sendo necessário guardar este estado nos routers.

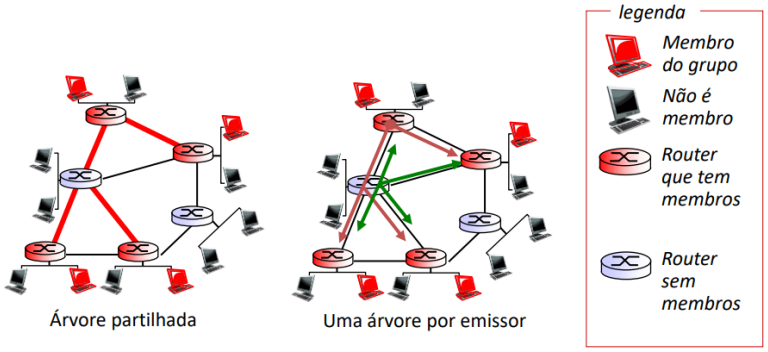
#### Spanning Tree

A spanning tree permite que nenhum nó receba pacotes repetidos. Consiste numa árvore à qual pertencem todos os nós da rede e estes encaminham pacotes apenas pela spanning tree. Existe ainda um nó central, para onde é enviada uma mensagem unicast join de cada nó da rede. Esta mensagem é encaminhada até chegar a um nó que ja pertence á spanning tree e essa ligação é adicionada.

### 3.4.2 Routing Multicast

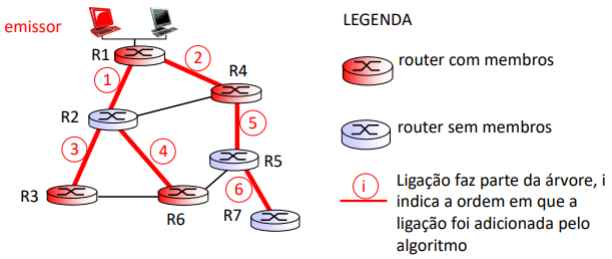
No routing multicast surge o problema de como ligar os routers multicast numa rede de routers unicast. A solução passa pelo uso de túneis, encapsulando o pacote multicast num pacote não-multicast. Esse pacote é então enviado usando unicast IP para o próximo router multicast, que por sua vez o desencapsula, á semelhança dos túneis IPv6.

O objetivo é então encontrar uma ou várias árvores que liguem os routers com membros de um dado grupo multicast. Estas árvores podem ser partilhadas (uma para todos os membros do grupo) ou baseadas no emissor (uma árvore por emissor).




#### Shortest Path Tree

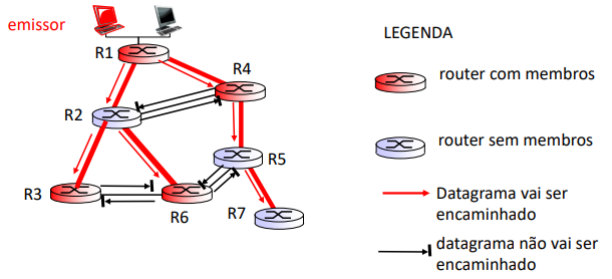
A shortest path tree é uma árvore baseada no emissor que contém caminhos mais curtos entre o emissor e os destinatários, usando o algoritmo de Dijkstra. Surge um problema, pois os destinatários teriam de ter visão da rede e conhecer os outros destinatários.




### Reverse Path Forwarding

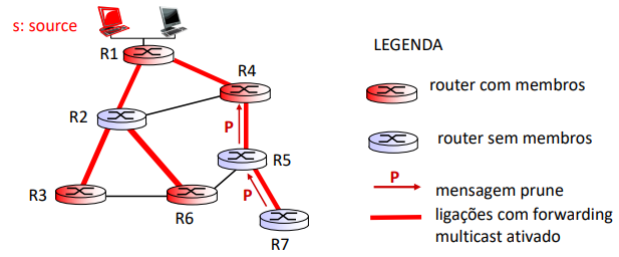
O reverse path forwarding é uma árvore baseada no emissor se usa a informação que o router já tem, do caminho mais curto entre si e o emissor. De seguida executa-se o seguinte algoritmo:

- Se o datagrama multicast é recebido na interface que é caminho mais curto para o emissor faz flooding controlado
- Caso contrário, o datagrama é ignorado




O resultado é uma shortest path tree baseada no emissor, mas invertida. Na eventualidade das ligações serem assimétricas esta pode não ser a melhor situação.

Assume-se inicialmente que todos os nós da rede são membros do grupo e progressivamente vai fazendo pruning. O processo de pruning consiste em excluir os nós não membros do grupo da rede multicast, dado que é ineficiente encaminhar pacotes para estes nós. A solução consiste em os routers sem membros enviarem mensagens prune em direção ao emissor até chegar a um router com membros.



### Steiner Tree

A steiner tree é uma árvore partilhada que contém o custo mínimo que liga todos os routers que têm membros. Esta seria uma solução ótima, porém não existe nenhum algoritmo eficiente que permita chegar a esse ótimo (apesar de existirem soluções aceitáveis). Não é usado na prática devido á elevada complexidade computacional, á necessidade de ter informação de toda a rede e de voltar a executar o algoritmo sempre que um router entra ou sai.

### Center-Based Trees

As center-based trees são árvores partilhadas em que um dos nós é designado como sendo o centro da árvore. Para um nó se juntar á árvore este envia uma mensagem unicast join para o router central e o percurso desta mensagem pelos router intermediários passa a ser um novo ramo da árvore.

# Capítulo 4

# Camada de Ligação de Dados

A camada da ligação de dados tem a responsabilidade de transferir um datagrama de um nó para o nó adjacente.

## Conteúdos


**4.1 Visão Geral . . . . . 26**
4.1.1 Serviços . . . . . 26
4.1.2 Implementação . . . . . 27
**4.2 Protocolos de Acesso ao Meio . . . . . 27**
4.2.1 Tipos de Ligações . . . . . 27
4.2.2 Protocolos de Acesso Múltiplo . . . . . 27
4.2.3 Multiple Access Control (MAC) . . . . . 28
4.2.4 Particionamento do Canal . . . . . 28
4.2.5 Acesso Aleatório . . . . . 28
4.2.6 Acesso Ordenado . . . . . 29
**4.3 Redes LAN . . . . . 29**
4.3.1 Endereço MAC . . . . . 29
4.3.2 Address Resolution Protocol (ARP) . . . . . 30
4.3.3 Ethernet . . . . . 30
4.3.4 Switches e Routers . . . . . 31
4.3.5 Virtual Local Area Network (VLAN) . . . . . 32
**4.4 Redes em Data Centers . . . . . 33**


## 4.1 Visão Geral

### 4.1.1 Serviços

#### Terminologia

Nós - Computadores terminais e routers

Ligações - Canais de comunicação que ligam nós adjacentes

Frame - Trama que encapsula datagrama

**Framing**

Encapsula os datagramas em frames e adiciona cabeçalho, possui protocolo de acesso ao canal se for partilhado e utiliza endereços MAC para identificar emissor e destino.

**Entrega Fiável**

A entrega entre nós adjacentes é fiável, á semelhança do TCP. Este tipo de entrega normalmente só é usado em ligações com alta taxa de erro.

**Controlo de Fluxo entre Nós Adjacentes**

**Deteção e/ou Correção de Erros**

Nota: Estes serviços não são comuns a todos os protocolos da camada.

### 4.1.2 Implementação

Esta camada existe em todos os terminais, switches e routers. É implementada num adaptador (NIC) ou num chip (Ethernet). Trata-se, portanto, de uma combinação de hardware, software e firmware.

## 4.2 Protocolos de Acesso ao Meio

### 4.2.1 Tipos de Ligações

**Ponto-a-Ponto**

Ex: ligação entre o computador e o switch Ethernet

**Broadcast (Meio Partilhado)**

- A Ethernet original
- HFC (redes cabo)
- 802.11 wireless LAN (WiFi)

### 4.2.2 Protocolos de Acesso Múltiplo

Num canal partilhado, duas ou mais transmissões simultâneas resultam em interferência. Dado um canal partilhado com taxa de transmissão máxima de $R$ bps, num protocolo ideal:

- Quando um nó quer transmitir, transmite à taxa de $R$ bps
- Quando $M$ nós querem transmitir, cada nó envia a uma taxa média de $\frac{R}{M}$
- É totalmente descentralizado, i.e. não é necessário nenhum nó para coordenar as transmissões e não é necessário sincronizar relógios entre nós
- É simples e fácil de implementar

### 4.2.3 Multiple Access Control (MAC)

O MAC é um algoritmo que determina como é que os nós partilham o canal sem causar colisão. Existem três classes de protocolos MAC: particionamento do canal, acesso aleatório e acesso ordenado.

### 4.2.4 Particionamento do Canal

Os protocolos MAC com particionamento do canal dividem o canal em pedaços mais pequenos (slots temporais, frequências, códigos) e cada nó pode usar cada pedaço de forma exclusiva. Geralmente estes protocolos são eficientes quando há muito tráfego, mas não quando há pouco tráfego, já que só é reservada $\frac{1}{N}$ da largura de banda para cada nó.

#### Time Division Multiple Access (TDMA)

No TDMA o acesso ao canal é feito em rondas. Cada nó tem um slot com tamanho fixo (tamanho = tempo de transmissão do pacote) em cada ronda. Slots não usados são desperdiçados

#### Frequency Division Multiple Access (FDMA)

No FDMA espetro de frequências do canal é dividido em várias bandas de frequência. A cada nó é atribuída uma banda e quando um nó não transmite dados a sua frequência é desperdiçada, não é utilizada por ninguém

### 4.2.5 Acesso Aleatório

Os algoritmos com acesso aleatório não dividem o canal em pedaços, portanto as colisões são permitidas mas posteriormente faz-se a recuperação das colisões. Geralmente estes protocolos são eficientes quando há pouco tráfego, mas geram muitas colisões com tráfego elevado.

#### ALOHA

Quando um nó tem frame para transmitir, transmite-a imediatamente. É simples mas há uma grande probabilidade de colisão.

#### Slotted ALOHA

No slotted aloha os nós começam a transmitir no início de um slot. Se 2 ou mais transmitirem num slot, todos detetam a colisão. Quando o nó tem uma frame para enviar, transmite-a no próximo slot, com uma probabilidade $p$, até ter sucesso. Desta forma, um único nó ativo pode transferir á capacidade máxima do canal, no entanto, é geralmente ineficiente e é necessário que os relógios dos nós estejam sincronizados.

#### CSMA

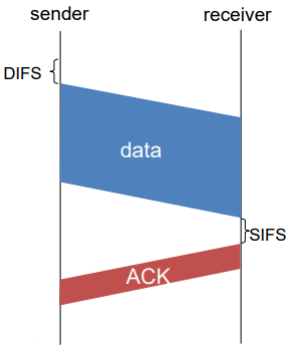
Os nós escutam antes de enviar. Se estiver ocupado adia a transmissão, caso contrário, transmite a frame. No entanto ainda podem ocorrer colisões devido ao tempo de propagação do meio. Quando isto ocorre, o pacote é desperdiçado.

#### CSMA/CD

Continua a escutar-se o canal enquanto se transmite, e adia-se transmissão se for detetada colisão, reduzindo assim o desperdício e detetando as colisões mais rapidamente. A deteção de colisões é fácil em redes com fios; mede-se a força do sinal e compara-se o sinal transmitido com o recebido, mas é mais difícil em redes sem fios pois o sinal recebido é muito influenciado pela força do sinal da transmissão local

#### CSMA/CA

Em CSMA/CA o objetivo não é necessariamente detetar colisões, mas sim evita-las. É usado em redes WiFi e procede da seguinte forma:




Emissor:

- Se o canal estiver livre livre DIFS segundos, transmitir trama
- Se o canal estiver ocupado começa o random backoff time e transmite quando o timer disparar. Se não houver ACK, aumentar random backoff time, repetir passo 2

O recetor deve returnar ACK depois de SIFS segundos se a trama for bem recebida (ACK é usado devido ao problema do terminal escondido).

### 4.2.6 Acesso Ordenado

Com acesso ordenado os nós vão usando o canal à vez. Os nós com mais para enviar podem eventualmente usar o canal durante mais tempo.

#### Baseado em Turnos (Polling)

Um nó principal convida os restantes nós a transmitir, à vez. Isto gera problemas de latência, overhead do polling e o facto de que o nó principal é um ponto único de falha.

#### Testemunho (Token)

Há um token de controlo que é passado de nó para nó, em sequência. Gera problemas semelhantes ao Polling.

## 4.3 Redes LAN

### 4.3.1 Endereço MAC

De forma a manter a independência entre camadas, localmente é utilizado o endereço MAC, em vez do endereço IP. Este endereço tem 48 bits e é único para cada adaptador de rede, servindo o propósito de enviar frames de uma interface

para outra à qual está fisicamente ligado, i.e., na mesma subrede IP. O endereço MAC não segue estrutura hierárquica, sendo possível migrar para outra rede mantendo o endereço, ao contrário do IP.

### 4.3.2 Address Resolution Protocol (ARP)

De forma a determinar o endereço MAC de uma interface a partir do seu endereço IP utiliza-se uma tabela ARP. Cada nó tem uma tabela com o mapeamento dos endereços IP/MAC para alguns dos outros nós da LAN, da seguinte forma:

$$< IP\ address;\ MAC\ address;\ TTL >$$

Em que TTL (time to leave) é o tempo a partir do qual o mapeamento é esquecido.

#### Mesma LAN

Quando uma interface A quer enviar um datagrama a outra interface B, mas o endereço MAC de B não está na tabela ARP de A, este faz broadcast de um query que contém o endereço IP de B, para pedir o MAC. Todos os nós na LAN recebem o ARP query e quando B recebe este pacote ARP responde a A com o seu endereço MAC. O par $< IP, MAC >$ de B é guardado na tabela ARP de A até o TTL expirar. Diz-se então que ARP é um protocolo plug-and-play, dado que os nós criam as suas tabelas ARP sem intervenção de um administrador da rede.

#### LAN Diferente

O routing entre redes LAN distintas é um pouco mais complexo. Sendo A e B duas interfaces e R um router:

- A cria um datagrama IP com endereço IP de A como emissor, e destino IP de B
- A cria uma frame com o endereço MAC de R como destino, e a frame contém o datagrama IP
- A frame é enviada de A para R
- A frame é recebida em R e o datagrama é passado para a camada IP
- R encaminha o datagrama com endereço IP emissor A, e destino B
- R cria uma frame com o endereço MAC de B como destino e o seu MAC como origem; a frame contém o datagrama IP

### 4.3.3 Ethernet

Ethernet foi a primeira tecnologia LAN a ter grande aceitação e é a tecnologia LAN com fios dominante. É mais simples e mais barata do que alternativas como as token LANs e a rede ATM e conseguiu ir evoluindo à medida que as velocidades aumentavam. Existem duas topologias físicas ethernet:

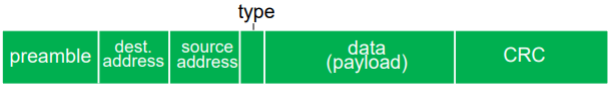
Bus - Todos os nós fazem parte do mesmo domínio de colisão

Estrela - Há um switch ativo no centro da rede e cada computador corre um protocolo Ethernet separado dos outros, e portanto não há colisões (em modo full duplex)

O ethernet é connectionless (não é necessário estabelecer nenhuma ligação) e não oferece garantias de fiabilidade (não são enviados ACKs).

#### Estrutura de uma frame Ethernet

O adaptador emissor encapsula datagramas IP (ou outro protocolo de rede) em frames Ethernet.




Preâmbulo - 7 bytes com um padrão 10101010 seguido por um byte com padrão 10101011, Usado para sincronizar emissor e recetor

Endereços MAC - 6 bytes para o emissor + 6 bytes para o destino

Tipo - Indica qual o protocolo de nível superior

CRC - Cyclic redundancy check, serve para detetar erros no destino. Se for detetado, a frame é descartada

#### Standards Ethernet

Existem diversos standards ethernet, com diferentes taxas e meios físicos de transmissão, no entanto todos partilham o protocolo MAC e o formato da frame

### 4.3.4 Switches e Routers

Antes de haver os switches, havia os hubs, simples repetidores de sinal. Os bits que chegavam a uma interface eram copiados para todas as outras interfaces e caso recebesse tramas em duas interfaces diferentes simultaneamente haveria uma colisão.

#### Switches

Switches armazenam e encaminham frames Ethernet. Examinam o endereço MAC das frames que recebe, encaminham as frames para uma ou mais saídas, e usa CSMA/CD. Para além disto, são transparentes e plug-and-play.

Para saber que computador está em cada uma das duas interfaces, cada switch tem uma tabela em que cada entrada contém:

$$< MAC;\ interface;\ TTL >$$

Esta tabela é criada a partir de um algoritmo de self learning, que funciona da seguinte forma:

- Guarda <interface de entrada, endereço MAC emissor>

- Indexa tabela usando <endereço MAC destinatário>
- Se encontrar entrada para <endereço MAC destinatário> então:

  Se destino estiver no segmento de onde chegou a frame, descarta-se

  Caso contrário encaminha-se frame pela interface indicada
- Caso contrário envia para todas as interfaces exceto aquela de onde chegou (flood)

Este algoritmo de self learning também é usado em interligações de switches, com a exceção de que se houver loops é necessário usar o protocolo spanning tree para remover ligações redundantes.

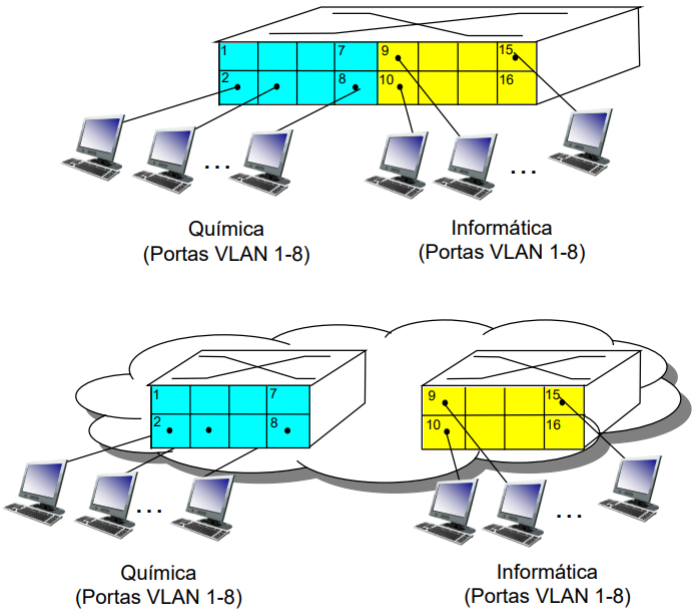
#### Switches vs Routers

Ambos funcionam em modo store-and-forward, a única diferença é que os routers examinam cabeçalhos da camada de rede e os switches examinam cabeçalhos da camada de ligação.

Ambos têm tabelas de forwarding, no entanto os routers computam as tabelas com algoritmos de routing, usando endereços IP enquanto que os switches aprendem como popular a tabela usando flooding, learning, e usam endereços MAC.

### 4.3.5 Virtual Local Area Network (VLAN)

São switches configurados para definir múltiplas redes LAN virtuais na mesma infrastrutura física, com o propósito de isolar o tráfego. O forwarding entre VLANs é feito por routing, tal como seria feito com switches separados.



## 4.4 Redes em Data Centers

Data centers têm imensos servidores no mesmo edifício, sendo necessário gerir e balancear a carga nos servidores e na rede. Para esse efeito faz-se routing ao nível da camada de aplicação:

- Recebe pedidos dos clientes
- Direcionam o tráfego dentro do data center
- Retorna resultado ao cliente (escondendo detalhes internos do data center)

A rede de um data center é do tipo Clos, possui muitas ligações, permitindo aumentar as taxas de transferência entre racks e dando uma maior fiabilidade devido à elevada redundância.

# Capítulo 5

# Outros Tópicos

## Conteúdos


**5.1 Redes Sem Fios . . . . . 34**
5.1.1 Elementos de uma Rede sem Fios . . . . . 34
5.1.2 Redes WiFi . . . . . 35
**5.2 Segurança de Redes . . . . . 36**


## 5.1 Redes Sem Fios

A comunicação usando ligações sem fios é um desafio pois está sujeita a múltiplas interferências, atenuações elevadas do sinal, reflexões, entre outros. No entanto é necessária, especialmente para lidar com utilizadores móveis.

### 5.1.1 Elementos de uma Rede sem Fios

Terminais - Podem ser estacionários ou móveis (portáteis, telemóveis...)

Estação Base - Funciona como intermediário responsável por enviar pacotes entre a rede com fios e a rede sem fios na sua área

Ligação - A ligação sem fios é usada para ligar os dispositivos móveis à estação base através de protocolos de acesso múltiplo que coordenam o acesso. Possui taxas de transmissão e distâncias muito variadas

Existem dois modos de operação:

#### Modo Infraestrutura

A estação base liga os dispositivos móveis à rede com fios. Utiliza o handoff, mecanismo para quando um dispositivo móvel muda de estação base.

#### Modo Ad Hoc

Não há estação base, os nós só transmitem para outros nós que estão dentro da sua área de cobertura, organizando-se em rede e encaminhando pacotes entre eles.

**Taxonomia das Redes Sem Fios**

| | Hop único | Múltiplos hops |
|---|---|---|
| Infraestrutura (e.g., APs) | Terminal liga-se a uma estação base (WiFi, WiMAX, rede celular) que por sua vez se liga à Internet | O terminal pode ter de enviar pacotes através de vários nós sem fios intermediários para se conseguir ligar à Internet (redes *mesh*) |
| Sem infraestrutura | Não há estação base e não há ligação à Internet (Bluetooth, redes ad hoc) | Não há estação base e não há ligação à Internet. Para alcançar outros nós pode ter de enviar pacotes por vários nós intermediários (redes MANET, VANET) |

**Problema do Terminal Escondido**

Para além dos problemas mencionados anteriormente existe ainda o problema do terminal escondido. Este problema acontece quando num conjunto de terminais nem todos os pares se conseguem escutar mutuamente e, portanto, não sabem que estão a interferir com a comunicação dos terminais que os ouvem ao mesmo tempo.

### 5.1.2 Redes WiFi

Existem vários tipos de redes WiFi, com diferentes frequências e larguras de banda, no entanto todos usam CSMA/CA como protocolo de acesso múltiplo e todos permitem ambos os modos infraestrutura e ad hoc.

**Arquitetura LAN 802.11**

O modo mais comum é o terminal sem fios comunicar com uma estação base (Access Point, AP). No modo infraestrutura uma célula (Basic Service Set, BSS) contém terminais sem fios e AP. No modo ad hoc só há terminais sem fios.

**Canais 802.11b**

Nas redes 802.11b o espetro é dividido em 14 canais em diferentes frequências, o administrador do AP escolhe a frequência para o seu AP. Para um terminal se associar com um AP faz um scan pelo canal à escuta de uma frame especial, a beacon frame, que contém o nome do AP (SSID) e o seu endereço MAC. De seguida escolhe o AP com que se quer associar (pode haver uma fase de autenticação). Tipicamente corre DHCP para obter um endereço IP para a subrede do AP.

Como os sinais são muito fracos e podem existir casos como o do terminal escondido, é muito difícil detetar colisões, portanto usa-se CSMA/CA e uma outra técnica para evitar colisões:

Emissor envia, primeiro, um pequeno pacote request-to-send (RTS) para a estação base usando CSMA. A BS faz o broadcast de um pacote clear-to-send CTS em resposta a um RTS. O CTS é escutado por todos os nós e o emissor pode transmitir a sua frame enquanto que os outros nós esperam.
A ideia é permitir que o emissor reserve o canal, assim evitando colisões de pacotes de dados longos

## 5.2 Segurança de Redes