Redes de Computadores

Resumo

Rafael Rodrigues

LEIC
Instituto Superior Técnico
2022/2023

Contents

1 Introduction

1.1 What is a Protocol?

A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.

- Hardware-implemented Protocol → Controls the flow of bits on the 'wire' between the 2 network cards.
- Congestion-Control Protocol → Controls the rate at which packets are transmitted between sender and receiver.
- Router Protocol → Determines a packet's path from source to destination.

1.2 Network Edge

1.3 Network Core

1.3.1 Packet Switching

- Comunicação dividida em pacotes, que partilham recursos com os restantes utilizadores.
- Cada um usa a largura de banda toda do canal. Recursos usados quando necessários.
- Não há alocação de canais.

1.3.2 Circuit Switching

- Reserva largura de banda em todo o percurso, somo se ligasse cabos fisicamente.
- Precisa de setup, não há partilha de recursos.
- Garantias de performance.

Packet Switching vs Circuit Switching

Packet Switching é excelente para comunicação bursty, cujas necessidade de recursos varia com o tempo.

1.4 Performance Metrics

1.5 Protocol Layers

To provide structure to the design of network protocols, network designers organize them in layers. Each layer provides its service by (1) performing certain actions within that layer and by (2) using the services of the layer directly below it.

Advantage → Layering provides a structured way to discuss system components.
Disadvantage → One layer may duplicate lower-layer functionality.

2 Application Layer

The application layer is where network applications and their application-layer protocols reside. The Internet's application layer includes many protocols, such as the HTTP protocol (which provides for Web document request and transfer), SMTP (which provides for the transfer of e-mail messages), and FTP (which provides for the transfer of files between two end systems). We'll see that certain network functions, such as the translation of human-friendly names for Internet end systems like www.ietf.org to a 32-bit network address, are also done with the help of a specific application-layer protocol, namely, the domain name system (DNS). We'll see in Chapter 2 that it is very easy to create and deploy our own new application-layer protocols.

An application-layer protocol is distributed over multiple end systems, with the application in one end system using the protocol to exchange packets of information with the application in another end system. We'll refer to this packet of information at the application layer as a message.

2.1 Principles of Network Applications

2.1.1 Network Application Architectures

- Client-Server Architecture:
  - Um host (servidor) para vários hosts (clientes).
  - Servidor sempre ligado, com endereço IP permanente.
  - Clientes não precisam de estar conectados continuamente, podem utilizar IPs dinâmicos.
  - Clientes não comunicam diretamente entre eles.
  - Escalável com server farms.
- P2P Architecture:
  - Não há servidor sempre ligado, peers comunicam entre si diretamente.
  - Peers não precisam de estar conectados continuamente e podem mudar de IP.
  - Bastante escalável, mas díficil de gerir.
- Híbrido
  Exemplo do Skype: Servidor dá aos clientes os endereços, mas chamadas voz são feitas diretamente entre os clientes utilizando uma aplicação Voice-over-IP P2P.

2.1.2 Processes Communicating

2.2 The Web and HTTP [Port 80]

É o protocolo da Camada de Aplicação usado na WWW (World Wide Web), usando o protocolo TCP da Camada de Transporte. Existem várias versões:

- HTTP 1.0
  - Versão não persistente.
  - Cada objeto transferido exige a criação e uso de uma sessão de TCP diferente.

- HTTP 1.1
  - Versão persistente, permitindo a transferência de vários objetos na mesma sessão de TCP.
  - Permite pipelining - se existem referências a um dado objeto numa dada transferência, o cliente pode pedi-los imediatamente, em vez de ter de esperar pela resposta de um pedido antes de fazer o próximo.
- HTTP 2.0
  - redução dos overheads dos cabeçalhos (são usados códigos para representar o conteúdo dos pedidos e respostas).
  - permite que várias tabs de um dado browser partilhem a mesma ligação TCP.
  - **Server Push**: O servidor analisa o HTML e vê que outros ficheiros são necessários (e consequentemente, serão pedidos) e envia-os assim que possível, não sendo necessário um pedido por parte do utilizador.
  - **Mitigação de HOL blocking**: Os objetos são divididos em pacotes e a transferência é intercalada, de forma a transmitir os ficheiros mais pequenos primeiro.
- HTTP 3.0
  - Invés do protocolo de comunicação, usa-se o protocolo QUIC - mistura do protocolo UDP com algumas diferenças e com encriptação.

2.2.1 Non-Persistent and Persistent Connections

- Non-Persistent Connection:
  Cada par pedido/resposta é feito numa ligação TCP diferente.
  - 2 RTT por objeto HTML.
  - $2 + 2N\ RTT$
- Persistent Connection:
  Todos os pares pedido/resposta são feitos na mesma ligação TCP.
  - 2 RTT para o primeiro objeto HTML.
  - 1 RTT para os restantes objetos.
  - $2 + N\ RTT$

Com pipelining:

- 1 RTT para todos os objetos referenciados.
- $2 + 1\ RTT$

2.2.2 HTTP Message Format

Response status:

- 200 OK - O pedido teve sucesso. O objeto pedido está no fim da mensagem;
- 301 Moved Permanently - O objeto pedido trocou de sítio. A sua nova localização está no header Location;

- 400 Bad Request - O servidor não entendeu o pedido porque o pedido tem erros;
- 404 Not Found - O documento pedido não se encontra no servidor;
- 500 Internal Server Error - Erro do servidor;

2.2.3 User-Client Interaction: Cookies

Cookies permitem que o protocolo HTTP mantenha estado. São guardadas pelo utilizador.
O servidor envia cookies (cookie header) na resposta HTTP.

2.2.4 Web Caching

The Conditional GET

Quando é solicitado um objeto que o browser já tem guardado em cache, o browser envia um GET condicional ao servidor, para verificar que a cópia guardada em cache está atualizada.
O servidor responde, mas caso não tenha sido modificado, o objeto não é enviado de modo a poupar bandwidth e melhorar o desempenho.

2.3 Electronic Mail in the Internet

Os clientes compõem mensagens de e-mail que são transmitidas usando o protocolo SMTP para o servidor que hospeda os e-mails.
Depois, ainda através do protocolo SMTP, os e-mails são transmitidos pela internet até chegarem ao servidor que hospeda o endereço de destino.
Finalmente, este servidor entrega o e-mail usando o protocolo Mail Access.

2.3.1 SMTP [TCP Port 25]

Este protocolo é muito simples, apenas contendo 5 comandos:

- HELO ou EHLO - serve para dois servidores SMTP estabelecerem uma ligação entre sí;
- MAIL FROM - indica quem é o remetente (quem envia) do e-mail;
- RCPT TO - indica o destinatário do e-mail;
- DATA - indica os dados que se pretendem enviar;
- QUIT - fecha a ligação;

O protocolo apenas olha para o cabeçalho e apenas aceita caracteres ASCII de 7 bits.

2.3.2 Mail Access Protocols

Os protocolos usado pela aplicação de leitura de e-mail quando se liga ao servidor onde as mensagens estão armazenadas são:

- Post Office Protocol - Version 3 (POP3)
- Internet Mail Access Protocol (IMAP)
- HTTP

2.3.3 Comparison with HTTP

2.3.4 Mail Message Formats

2.3.5 Mail Access Protocols

2.4 Domain Name System (DNS) [Port 53]

- Associa nomes **(domínios)** a endereços IP.
- Base de dados distribuída por uma hierarquia de servidores de nomes **(name servers)**.
- Acedida através de um **resolver**.
- Utiliza maioritariamente o protocolo UDP, é usado TCP se a resposta for demasiado grande.

2.4.1 DNS Records

Resource Records (RR): (**name**, **value**, **type**, ttl)

The meaning of **name** and **value** depend on **type**:

- **type** = A:
  - **name**: hostname
  - **value**: IP address of the hostname
- **type** = NS:
  - **name**: domain
  - **value**: hostname of an authoritative DNS server knowing how to obtain IP address for the domain
- **type** = CNAME:
  - **name**: alias hostname
  - **value**: canonical hostname (real name)
- **type** = MX:
  - **name**: alias hostname of a **mail server**
  - **value**: canonical hostname (real name)

2.5 File Distribution

Distribution Time $\rightarrow$ Time it takes to get a copy of a file with size F to all N peers.

2.5.1 Client-Server Architecture

In the client-server architecture, none of the peers aid in distributing the file:

- The server must transmit one copy of the file to each of the N peers. Thus, the server must transmit $N \cdot F$ bits. Since the server's upload rate is u_s, the time to distribute the time must be at least $\dfrac{N \cdot F}{u_s}$

- Let d_{min} be the smallest download rate among the N peers. The peer with the d_{min} cannot obtain all F bits of the file in less than $\frac{F}{d_{min}}$ seconds. Thus, this is the distribution time.

$$D_{client-server} = max\left\{\frac{N \cdot F}{u_s}, \frac{F}{d_{min}}\right\}$$

2.5.2 P2P Architecture

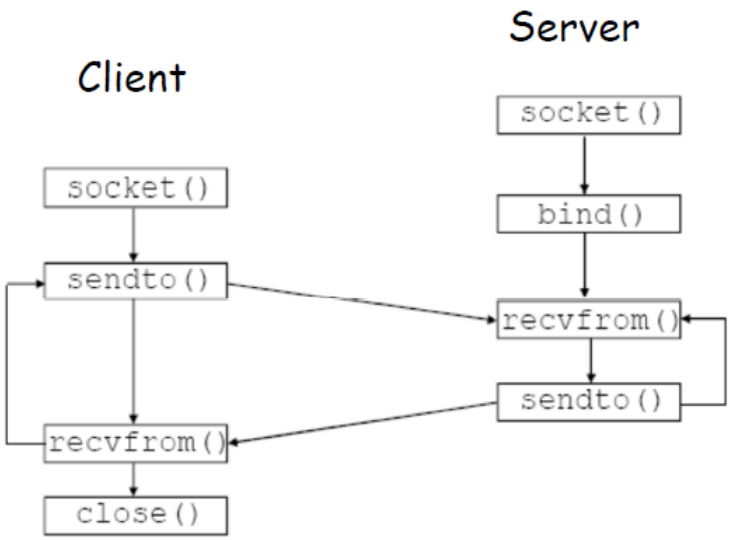
In the P2P architecture, when a peer receives some file data, it can use its own upload capacity to redistribute the data to other peers:

- The server must transmit F bits of the file at least once into its access link. Since the server's upload rate is u_s, the minimum distribution time is at least $\frac{N \cdot F}{u_s}$
- Let d_{min} be the smallest download rate among the N peers. The peer with the d_{min} cannot obtain all F bits of the file in less than $\frac{F}{d_{min}}$ seconds. Thus, this is the distribution time.
- The total upload capacity of the system as a whole is equal to $u_{total} = u_s + u_1 + u_2 + ... + u_N$. The system must upload F bits to each of the N peers, thus delivering a total of $N \cdot F$ bits. This cannot be done at a rate faster than u_{total}. Hence, the minimum distribution time is also at least $\frac{N \cdot F}{u_{total}}$

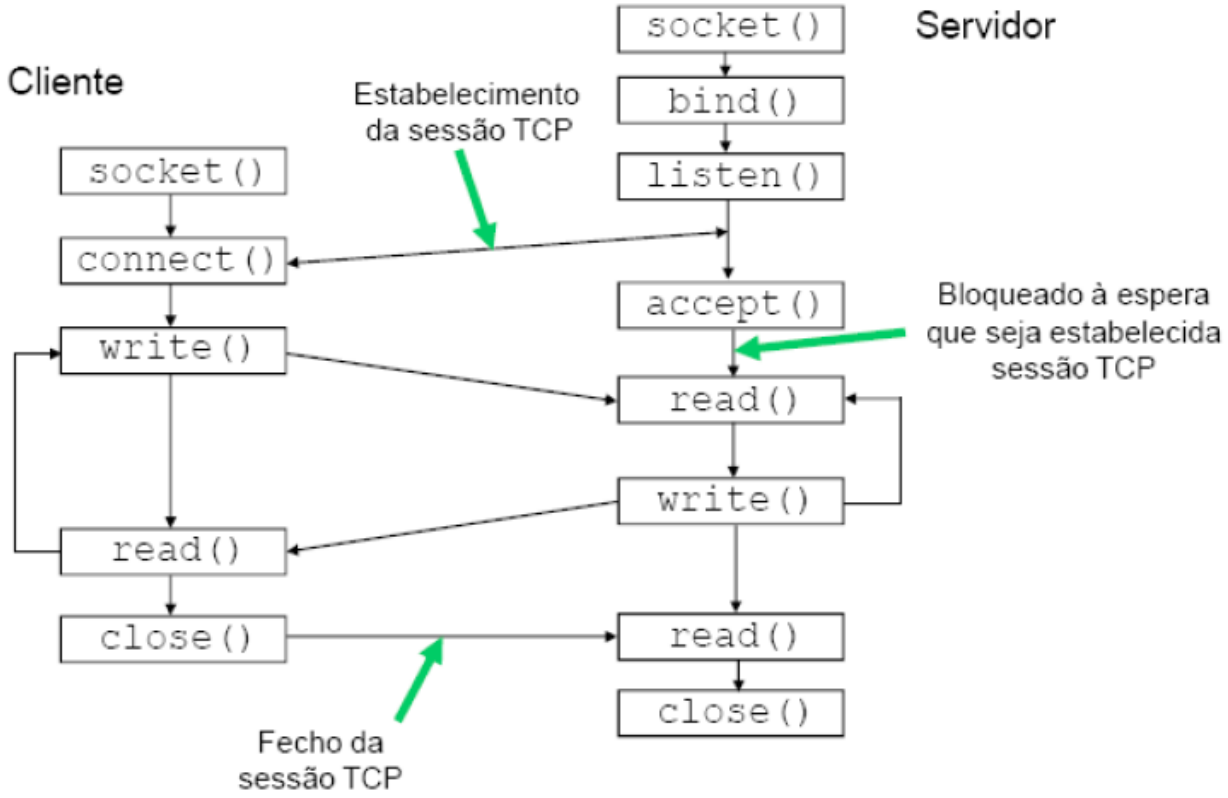
$$D_{client-server} = max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{N \cdot F}{u_{total}}\right\}$$

2.6 Socket Programming

UDP



TCP



TCP vs UDP

TCP	UDP
read() and write()	sendto() and recvfrom()
Byte stream (no byte is lost)	Messages may be lost
Bytes read with read() may correspond to several write()	Preserves boundary between messages
Bytes written with write() may need to be read with several read()	Each message read with recvfrom() correspond to a single sendto()

3 Transport Layer

The Internet's transport layer transports application-layer messages between application endpoints. In the Internet there are two transport protocols, TCP and UDP, either of which can transport application layer messages. TCP provides a connection-oriented service to its applications. This service includes guaranteed delivery of application-layer messages to the destination and flow control (that is, sender/receiver speed matching). TCP also breaks long messages into shorter segments and provides a congestion-control mechanism, so that a source throttles its transmission rate when the network is congested. The UDP protocol provides a connectionless service to its applications. This is a no-frills service that provides no reliability, no flow control, and no congestion control. In this book, we'll refer to a transport-layer packet as a segment.

3.1 User Datagram Protocol (UDP)

- Um socket UDP é identificado pelo tuplo: (dest IP, dest port)
- Quando recebe datagrama UDP, entrega ao socket com esse IP:porto
- Sem ligações nem garantias de ordem ou entrega
- Sem controlo de congestão
- Checksum (opcional):

  Este campo dos segmentos serve para detetar erros num dado segmento transmitido. Quando o segmento é enviado, o emissor calcula e guarda a soma dos dois endereços IP enviados (considerados números para ser possível fazer a soma) no campo Checksum.

  Quando o segmento é recebido, o recetor faz a mesma soma e verifica-se se o resultado é igual ao valor que está no campo Checksum.

  Se tiver existido alguma corrupção, ou seja, pelo menos algum bit que tenha sido mal transmitido, a soma dará um valor diferente e então o erro será detetado. Se for esse o caso, o pacote é apenas ignorado/dropped (no caso de TCP, o segmento seria antes pedido novamente ao emissor).

  Contudo, este sistema continua a não ser muito confiável - podem existir várias combinações de erros que gerem a mesma soma e um erro pode também estar no checksum.

  Uma melhoria que se pode fazer é considerar mais valores para o cálculo da soma do checksum, de forma a aumentar a entropia do erro - pode-se, por exemplo, também considerar valores dos cabeçalhos de outras camadas.

Headers pequenos, é simples e rápido. Usado para streaming, DNS e SNMP.

3.2 Reliable Data Transfer (RDT)

3.2.1 Stop-and-Wait

Protocolo de janela deslizante em que a janela de transmissão tem dimensão igual a 1, e janela de receção também tem dimensão igual a 1.

Eficiência de utilização de linha: $\frac{\frac{L}{R}}{\frac{L}{R} + RTT}$

3.2.2 Sliding Window

Este esquema tem uma "janela" de tempo onde se podem enviar pacotes.
Depois dessa janela acabar (ou seja, quando se chegar ao limite de N pacotes pendentes), espera-se pela receção de pelo menos um pacote para poder enviar mais.
O ideal seria dimensionar o envio de tal forma que não exista tempo perdido.
Contudo, ainda podem existir vários tipos de perdas, como apresentado no RDT 3.0. Para resolver esses problemas, existem "sub-protocolos".

Go-Back-N (GBN)

Go-back-N é um protocolo em que a janela de transmissão pode ter dimensão $N > 1$, mas a janela de receção tem sempre dimensão = 1.

- Emissor pode ter até N pacotes enviados à espera de ACK.
- ACKs são cumulativos ($ACKn \Leftrightarrow ACK1, ACK2, \cdots, ACKn$)
- Timer para pacote sem ACK enviado há mais tempo, se expira **reenvia todos os pacotes seguintes** contidos na janela.

Selective Repeat (SR)

Selective Repeat é um protocolo em que a janela de transmissão tem a mesma dimensão da janela de receção.

- Emissor pode ter até N pacotes na janela.
- ACKs individuais, não cumulativos.
- Um timer por pacote unACKed enviado, quando o tempo expira só reenvia esse pacote.

3.3 Transmission Control Protocol (TCP)

- Um socket TCP é identificado pelo tuplo: (src IP, src port, dest IP, dest port)
- Há 1 socket por ligação (cliente), cada uma abre um socket novo.
- Entrega fiável e ordenada.

4 Network Layer

The Internet's network layer is responsible for moving network-layer packets known as datagrams from one host to another. The Internet transport-layer protocol (TCP or UDP) in a source host passes a transport-layer segment and a destination address to the network layer, just as you would give the postal service a letter with a destination address. The network layer then provides the service of delivering the segment to the transport layer in the destination host.

The Internet's network layer includes the celebrated IP protocol, which defines the fields in the datagram as well as how the end systems and routers act on these fields. There is only one IP protocol, and all Internet components that have a network layer must run the IP protocol. The Internet's network layer also contains routing protocols that determine the routes that datagrams take between sources and destinations. The Internet has many routing protocols. As we saw in Section 1.3, the Internet is a network of networks, and within a network, the network administrator can run any routing protocol desired. Although the network layer contains both the IP protocol and numerous routing protocols, it is often simply referred to as the IP layer, reflecting the fact that IP is the glue that binds the Internet together.

4.1 Overview of Network Layer

The primary role of the **data plane** is to forward datagrams from its input links to its output links.

The primary role of the **control plane** is to coordinate these local, per-route forwarding actions so that datagrams are ultimately transferred end-to-end, along paths of routers between source and destination.

4.1.1 Forwarding and Routing: The Data and Control Planes

The primary role of the network layer is to move packets from the sending host to a receiving host. Thus, there are 2 important network-layer function:

1. Forwarding → When a packet arrives at a router's input link, it must move the packet to the appropriate output link. A packet might also be blocked from exiting a router or duplicated and sent over multiple outgoing links.
   Forwarding is router-local, and is the key function performed by the data-plane functionality.

2. Routing → The network layer must determine the path taken by packets as they flow from a sender to a receiver. To do so, the network layer uses multiple routing algorithms. Routing refers to the network-wide. **Routing is a network-wide process**.

4.2

4.3 The Internet Protocol (IP)

4.3.1 Fragmentation

4.3.2 Adressing

Dynamic Host Configuration Protocol (DHCP) [Server Port 67, Client Port 68]

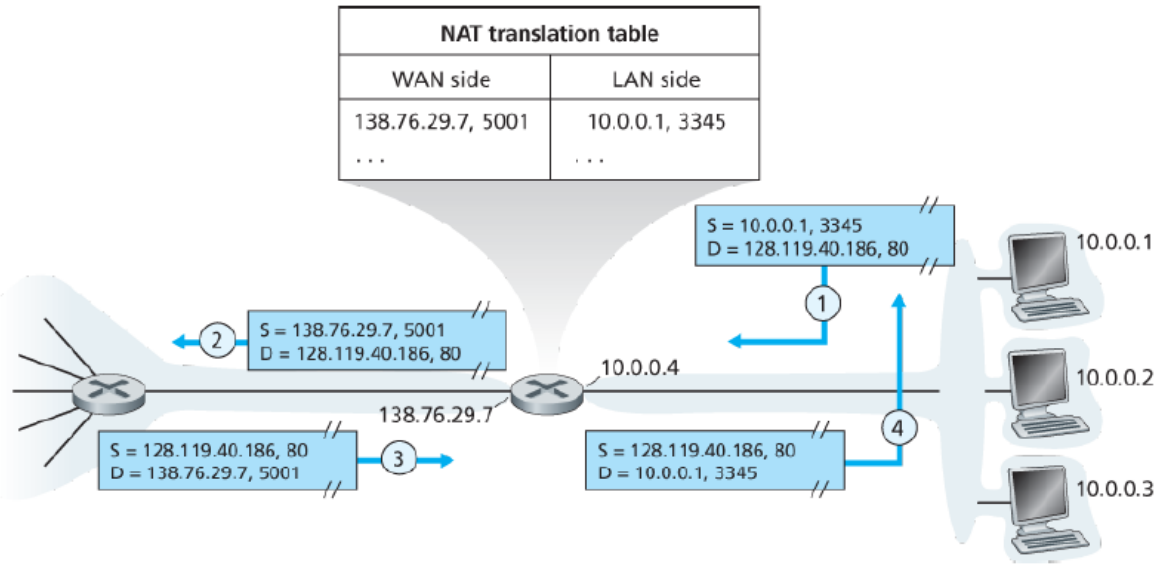
- Mensagens de DHCP são em UDP.
- Endereços são emprestados por um tempo finito (lease time).
- Após 50% do lease time, o cliente tenta renovar o IP (**DHCP Request**).
  Se não conseguir, tenta de novo a 85%.

Outras mensagens:

- Decline: cliente rejeita oferta
- NACK: servidor não consegue satisfazer pedido
- Release: cliente informa que já não quer usar endereço
- Inform: cliente pede parâmetros extra

Network Address Translation (NAT)

1. Quando um utilizador da rede privada envia um pacote, o NAT substitui o endereço IP interno do remetente pelo externo, e memoriza o correspondente.
2. Quando recebe uma resposta o NAT restaura o IP interno.



4.4 Software Defined Network (SDN)

A abordagem usando SDN para a implementação da camada de rede considera que apenas o plano de dados (**Data Plane**) é implementado em cada encaminhador.

4.5 Routing Algorithms

4.5.1 Link-State (LS) - Djkstra

Um encaminhador pode receber mais do que uma cópia de um dado LSP (Link State Packet).

4.5.2 Distance-Vector (DV) - Bellman-Ford

4.6 Broadcast e Multicast

Broadcast

O objetivo de um Broadcast é enviar um pacote para todos os outros nós da rede.
Contudo, fazer um broadcast pode gerar problemas entre routers.

Numa situação em que existe um anel que fará com que o pacote seja enviado de volta para routers que já o receberam - a este problema chama-se Flooding.

Para o resolver, um nó só faz broadcast de um pacote se ainda não fez broadcast desse mesmo pacote (Os routers guardam os identificadores dos pacotes) - Controlled Flooding.

Outra alternativa seria construir uma Spanning Tree - árvore que cobre todos os routers com o menor número de ligações e sem loops.

Multicast

O objetivo de um Multicast é uma mensagem ser entregue a um grupo de destinatários específico.

É usado um endereço IP especial, juntamente com protocolos para gestão local de pertença a um grupo e para a criação de uma árvore de distribuição.

5 Link Layer

The Internet's network layer routes a datagram through a series of routers between the source and destination. To move a packet from one node (host or router) to the next node in the route, the network layer relies on the services of the link layer. In particular, at each node, the network layer passes the datagram down to the link layer, which delivers the datagram to the next node along the route. At this next node, the link layer passes the datagram up to the network layer.

The services provided by the link layer depend on the specific link-layer protocol that is employed over the link. For example, some link-layer protocols provide reliable delivery, from transmitting node, over one link, to receiving node. Note that this reliable delivery service is different from the reliable delivery service of TCP, which provides reliable delivery from one end system to another. Examples of link-layer protocols include Ethernet, WiFi, and the cable access network's DOCSIS protocol. As datagrams typically need to traverse several links to travel from source to destination, a datagram may be handled by different link-layer protocols at different links along its route. For example, a datagram may be handled by Ethernet on one link and by PPP on the next link. The network layer will receive a different service from each of the different link-layer protocols. In this book, we'll refer to the link-layer packets as frames.

5.1 Introduction to the Link Layer

5.2 Error Detection and Correction Techniques

Os dados, por diversos motivos, podem sofrer erros de transmissão, fazendo com que um ou mais bits venham trocados.
Para detetar esses erros, existem várias formas:

5.2.1 Parity Checks

Forma antiga de detetar erros.
No fim da transmissão, era acrescentado um bit que dependia do número de bits totais da mensagem:

- Se o número de bits fosse par, colocava-se um 0;
- Se o número de bits fosse ímpar, colocava-se um 1.

Não é uma forma muito boa pois não deteta um número par de erros.

5.2.2 Cyclic Redundancy Check (CRC)

Muito usado atualmente. É feita uma operação matemática, usando divisão de polinómios.

- Para gerar mensagem com CRC:

  Dada uma mensagem: 1001 1010
  Dado um polinómio gerador: $x^3 + x^2 + 1 \rightarrow 1101$ $(n = 3)$

  1. Fazemos shift left da mensagem correspondente ao grau do polinómio (n):
     1001 1010 $\rightarrow$ 100 1101 0000

2. Fazemos a divisão da mensagem shiftada pelo polinómio gerador, obtendo um resto:
   $100\ 1101\ 0000 \div 1101 = quociente + 101$
3. A mensagem que será transmitida é a soma da mensagem shiftada com o resto:
   $100\ 1101\ 0000 + 101 = 100\ 1101\ 0101$

- Para verificar se a mensagem recebida tem erros:

  Dada uma mensagem recebida: 10 1010 0111
  Dado um polinómio gerador: $x^3 + x^2 + 1 \ \rightarrow \ 1101$

  1. Fazemos a divisão da mensagem recebida pelo polinómio gerador, obtendo um resto:
     $10\ 1010\ 0111 \div 1101 = quociente + 001$
  2. Se o resto da divisão for diferente de zero significa que foi detetado um erro.

5.3 Multiple Access Links and Protocols

Quando é feito um broadcast FF:FF:FF:FF:FF:FF, este é partilhado por todos os membros da rede. Isto implica que o canal possa sofrer colisões muito facilmente, bastando que um nó receba dois ou mais sinais ao mesmo tempo.

Protocolos Multiple Access Control - algoritmos distribuídos que determinam como é que os nós partilham um canal, ou seja, quando é que um dado nó pode transmitir.
Existem vários protocolos que implementam o acesso de formas diferentes:

5.3.1 Channel Partitioning Protocols

- Dividir o canal em "pedaços" menores (intervalos de tempo, frequências, códigos, ...);
- Alocar um pedaço para o nó para uso exclusivo.

Frequency Division Multiple Access (FDMA)

Neste tipo de particionamento, o espectro de canal é dividido em faixas de frequência. Assim, é atribuída a cada estação uma faixa de frequência fixa mas o tempo de transmissão não utilizado nas faixas de frequência não é aproveitado.

Time Division Multiple Access (TDMA)

O acesso ao canal é feito em rondas. É atribuído a cada host uma fatia temporal em cada ronda. Também aqui, o tempo não utilizado não é aproveitado.

Code Division Multiple Access (CDMA)

Cada utilizador tem acesso à frequência completa, durante todo o tempo. São usados diferentes códigos para distinguir os utilizadores.

5.3.2 Dynamic Allocation Protocols (Taking-Turns)

- Os nós tomam a vez (mas os nós com mais dados para enviar podem transmitir com mais frequência);

Poll / Select

Um computador central controla a atividade dos outros.

Token Passing

Um token que dá a permissão de transmissão é passado entre os hosts.
Quando um host acaba de transmitir, passa o token para outro host.

Esta medida não é ideal se não existir muito tráfego, pois existe dependência que os hosts transmitam mensagens.

5.3.3 Random Access Protocols

- O canal não é dividido, logo permite colisões;
- É possível recuperar das colisões.

ALOHA

É um protocolo simples que não tem sincronização.
Existe uma colisão se duas ou mais tramas se sobrepuserem, sendo programada uma retransmissão para um instante futuro aleatório.

Slotted ALOHA

Igual ao protocolo anterior, mas o tempo é dividido em slots temporais do mesmo tamanho.
Os nós só podem começar a transmitir no início de um slot.

Carrier Sense Multiple Access (CSMA)

Baseia-se em não causar interrupções:

- Se o canal estiver calado, transmite a trama inteira;
- Se o canal estiver ocupado, adia a transmissão.

Contudo, ainda podem existir colisões devido ao tempo de propagação, o que implica que toda a transmissão de um pacote tenha que ser descartada.

CSMA with Collision Detection (CSMA\CD)

Versão aprimorada do protocolo anterior, onde consegue detetar colisões mais rapidamente. Ao ser detetada uma colisão, as transmissões são imediatamente abortadas, reduzindo a ocupação do canal.

5.4 Summary of MAC Protocols

Channel Partitioning Protocols:

- Eficientes com uma carga alta e constante de todos os nós;
- Ineficientes em cargas baixas ou desequilibradas.

Dynamic Allocation Protocols (Taking-Turns):

- Compartilham o canal de forma eficiente e justa em cargas altas;
- Baixa carga: ineficiente, pois os nós ativos precisam esperar pelos nós sem/com pouca atividade para "passarem a vez";

Random Access Protocols:

- Eficientes em carga baixa: um único nó pode utilizar unicamente o canal;
- Carga alta: sobrecarga de colisão.

5.5 Switched Local Area Networks

5.5.1 Link-Layer Addressing and ARP

MAC Addresses

Estes endereços têm propósitos diferentes dos endereços IP: os endereços MAC têm um propósito mais local enquanto que os endereços IP têm um contexto mais geográfico.
O que cria esta distinção é o facto de os endereços MAC serem intrínsecos das placas de rede, ou seja, são como uma fingerprint; Já os endereços IP podem ser alterados facilmente e é possível ser construído um sistema e uma divisão hierárquica da sua distribuição, através das máscaras de rede.

Para um dado endereço MAC, os três primeiros bytes identificam o fabricante enquanto que os os três últimos bytes identificam o equipamento.

Nestes endereços também é possível ser feito um broadcast, fazendo-o para o endereço FF:FF:FF:FF:FF:FF. Ao enviar um pacote para esse endereço, esse pacote será reencaminhado para todos os hosts da rede local.

Address Resolution Protocol (ARP)

O router precisa de saber o endereço MAC correspondente ao endereço IP recebido. Para tal, usa-se o protocolo ARP - Address Resolution Protocol:

1. O router faz um ARP Request, ou seja, um broadcast, enviando para o endereço FF:FF:FF:FF:FF a pergunta: Quem tem este endereço IP?

2. O host com o endereço IP responde com o seu endereço MAC.

De forma a não se estarem sempre a repetir estes pedidos, os hosts guardam numa tabela (tabela ARP), o mapeamento entre endereço IP e endereço MAC - (endereço IP, endereço MAC, TTL).

5.5.2 Ethernet

5.5.3 Link-Layer Switches

Self-Learning

Switches are self-learning, meaning the switch table is built automatically, dynamically and autonomously.

1. The switch table is initially empty.
2. For each incoming frame received on an interface, the switch stores in its table:

MAC Address	Interface	Current time
E6-E9-00-17-BB-4B	1	00:01

3. The switch deletes an address in the table if no frames are received with that address as the source address after some time (**aging time**).

Spanning-Tree Protocol (STP)

- Establishes a root node called the root bridge.
- Constructs a topology that has one path from/to every network node.
- Resulting tree originates from the root bridge.
- Redundant links that are not part of the shortest path are blocked.

Bridge Protocol Data Unit (BPDU)

Convention for Conf-BPDUs: Root ID . Root Path Cost . Bridge ID

A configuration C_1 is said to be better than C_2 if (ordered by decreasing importance):

1. C_1 Root ID is lower than that of C_2.
2. C_1 Root Path Cost is lower than that of C_2.
3. C_1 Bridge ID is lower than that of C_2.
4. C_1 Port ID is lower than that of C_2.

5.5.4 Virtual Local Area Networks (VLANs)

Existem switches que suportam o uso de VLANs - Virtual LANs, ou seja, permitem configurar redes virtuais para as suas portas.

As VLANs têm algumas vantagens:

1. dividem a rede em partes, reduzindo o domínio de broadcast o que, consequentemente, aumenta a bandwidth pois as mensagens não são enviadas para portas desnecessárias;
2. adicionam segurança, pois hosts em diferentes VLANs não conseguem comunicar entre si diretamente;

Para as VLANs comunicarem entre si, isso é feito através de routing (pelos Routers).

5.6 Wireless [802.11]

5.6.1 Architecture

5.6.2 MAC Protocol

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

Este algoritmo é bastante similar ao básico CSMA. O que difere dele é que, para além de esperar que o canal fique livre, ainda espera um tempo aleatório depois do canal se libertar, e esse tempo só é descontado quando o canal está livre. Assim, o "sortudo" que teve o tempo menor começa a transmitir e os outros voltam a ficar à espera que o canal fique livre.
Depois de uma transmissão, o AP devolve um ACK para confirmar o sucesso da transmissão. Isto ajuda a resolver o problema do terminal escondido - porém não o resolve de todo pois podem haver tempos em que o início das transmissões coincide.
Para resolver isto, surgiu a variante com RTS-CTS.

CSMA/CA com Request-To-Send e Clear-To-Send (RTS-CTS)

Esta variante consiste numa reserva do canal - o emissor, no lugar de começar a sua transmissão imediatamente, envia antes um pedido de reserva do canal. Desta forma, o AP irá-lhe reservar o canal e dar-lhe um OK, fazendo broadcast de um CTS com a informação de quem vai emitir. Assim, um outro host que não oiça o emissor, ouvirá certamente o AP e não irá transmitir.
Se tiver existido uma colisão nesse pedido, esta terá um impacto muito reduzido, pois foi apenas transmitido um pacote.