TECNOLOGIAS DA INFORMAÇÃO E COMUNICAÇÃO

Vitor Rocio

Capítulo I – Conceitos básicos das TIC

Desde a pré-história que a informação e a comunicação são vitais para a espécie humana. A comunicação entre os membros dos grupos de caçadores da Idade da Pedra era fundamental para garantir o sucesso dos ataques coordenados a animais de grande porte. O desenvolvimento da linguagem humana foi a consequência desta necessidade.

Com o aparecimento das primeiras civilizações, começa a surgir a necessidade, no seio dos estados, de transmitir a informação de uma forma mais duradoura e eficaz. A invenção da escrita permitiu prolongar no tempo o registo da informação mais importante, podendo ser lida por várias pessoas em alturas diferentes. A escrita tem também a função de memória, sendo uma verdadeira extensão do cérebro humano.

A invenção da escrita influenciou tanto a forma de transmitir informação, que os historiadores consideram que a História propriamente dita só começou desde que se começou a registar os acontecimentos por escrito.

Ao longo do tempo, têm sido muitas as tecnologias da informação e comunicação, muitas das quais ainda hoje em uso: o papel, o ábaco, a imprensa, o telégrafo, a máquina de calcular. Só no século XX surgiram os computadores e as redes informáticas: são as tecnologias de tratamento e disseminação da informação por excelência, já que não possuem restrições quanto ao tipo de informação nem ao tipo de processamento que realizam.

1.1 Informação

Todos temos uma noção intuitiva de informação: o telejornal das oito divulga informação, somos informados das horas ao consultar um relógio, obtemos informação sobre o significado de uma palavra num dicionário. A informação é tratada pelo nosso cérebro, que filtra a que nos interessa e que, através do raciocínio, chega a conclusões que nos são úteis para tomar decisões. O cérebro humano é um órgão demasiado complexo para compreendermos totalmente como funciona e, em particular, não sabemos muito bem como lida com a informação. O modelo de informação usado nos computadores é, portanto, baseado no mundo físico exterior percepcionado pelos nossos sentidos e não numa eventual representação cerebral. Os tipos de informação com que interagimos mais frequentemente nos sistemas informáticos são os seguintes:

- **Texto**: uma sequência de caracteres (símbolos) que codificam um sistema de escrita de uma língua humana.
- **Imagem**: um padrão bidimensional de luz e cor, que reflecte o que a nossa visão capta. Esse padrão pode ser *estático*, quando não se altera com o tempo, ou *dinâmico*, quando depende do passar do tempo – neste caso designa-se vídeo.
- **Som**: um padrão de vibração do ar que reflecte o que é captado pelo nosso sentido da audição.

Destes três tipos de informação, o que nos é menos imediato, o texto (temos que aprender a lê-lo), é também o mais fácil de representar num computador. Basta atribuir um código numérico a cada letra e a mais alguns símbolos adicionais (espaço em

branco, pontuação, etc.) e representar o texto através da sequência de códigos correspondente à sequência de caracteres do texto. Por exemplo, usando o código ASCII (American Standard Code for Information Interchange), universalmente aceite nos sistemas informáticos, o texto TECNOLOGIA DE PONTA é representado da seguinte forma: 84, 69, 67, 78, 79, 76, 79, 71, 73, 65, 32, 68, 69, 32, 80, 79, 78, 84, 65.

O som e a imagem, embora constituam informação mais imediata para nós, têm uma representação mais complexa no computador. Devido a esta complexidade, só nos anos 90 se generalizaram os computadores pessoais multimédia, capazes de processar som e vídeo.

Uma imagem é um padrão bidimensional de **luz** e **cor**, logo, a primeira dificuldade é a representação da luz e da cor. A luz é representada por um número que identifica a intensidade luminosa. O tom de cor pode ser representada pelas percentagens relativas de três cores básicas (vermelho, verde e azul nos monitores; cyan, magenta e amarelo nas impressoras). A disposição espacial da luz e da cor é representada por um sistema de coordenadas cartesianas, ou seja, cada ponto da imagem é localizado espacialmente por duas coordenadas numéricas (x e y), que representam a distância horizontal e vertical a um dos cantos da imagem, tomado como referência. Para representar o vídeo, há que adicionar uma terceira coordenada ao sistema: o tempo.

O tempo também é um parâmetro importante na representação do som, além dos parâmetros típicos das ondas sonoras, como amplitudes, frequências e comprimentos de onda.

Do exposto, concluímos que tudo o que precisamos para representar uma grande variedade de informação são *números*. As tecnologias de informação e comunicação limitam-se a processar e transmitir gigantescas quantidades de números.

1.2. Números

Sabemos da matemática que há várias classes de números: naturais, inteiros, racionais, reais. Qual ou quais as que nos convêm para representar os tipos de informação que discutimos na secção anterior? Obviamente que serão os números reais, porque são a categoria mais abrangente. No entanto, para muitas aplicações, os números inteiros são suficientes e é vantajoso usá-los nestes casos, já que têm uma representação mais simples.

Os números inteiros (positivos, negativos e o zero) são infinitos, pelo que seria necessário um número infinito de dígitos para os representar. Na prática, porém, só se pode usar uma quantidade relativamente pequena: com 10 dígitos, por exemplo, é possível representar números inteiros até 9999999999, o que pode ser considerado um limite aceitável.

A quantidade de números reais também é infinita. O conceito de infinito é uma noção estranha: será que por serem também infinitos, existem tantos números reais como inteiros? Se assim fosse, poderíamos esperar que a representação dos números reais fosse semelhante à dos números inteiros. Mas, na realidade, não é isto que acontece. O matemático Georg Cantor demonstrou, em 1874, que existem muito mais números reais do que números inteiros (poderíamos dizer infinitamente mais…). Intuitivamente, este facto reflecte-se nos dois tipos de infinito presentes nos números reais: o infinitamente grande e o infinitamente pequeno. Para efeitos de representação, o infinitamente grande é limitado pela **amplitude** da representação, ou seja, o maior número real representável. O infinitamente pequeno traduz-se na **precisão** da representação, ou seja, o número de casas decimais.

Os números reais podem ser representados no computador de duas formas distintas: a representação em **vírgula fixa** e a representação em **vírgula flutuante**. A representação em vírgula fixa é a mais natural: a parte inteira do número é representada por um grupo de dígitos de dimensão fixa (tal como nos números inteiros) e a parte decimal também. A figura 1 ilustra a representação em vírgula fixa.

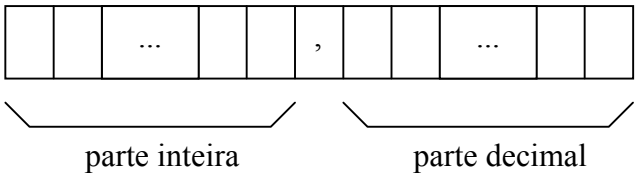


Figura 1: Representação em vírgula fixa

No entanto, a representação quase universalmente usada é a representação em vírgula flutuante. Nesta representação consideram-se apenas os dígitos significativos (eliminam-se os zeros à esquerda e à direita), e guarda-se a posição da vírgula decimal. A seguência de dígitos significativos chama-se **mantissa** e a posição da vírgula é o **expoente**. Por exemplo, na representação do número 56000, a mantissa é 56 e o expoente é 3 (três casas à direita da mantissa). No caso de um número com casas decimais, por exemplo, 0,0037, a mantissa é 37 e o expoente é -4 (quatro casas à esquerda da mantissa). Outro exemplo: 7,5 – a mantissa é 75, o expoente é -1. A figura 2 ilustra esquematicamente a representação em vírgula flutuante.

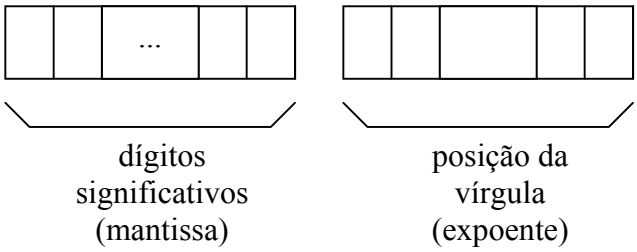


Figura 2: Representação em vírgula flutuante

A representação em vírgula flutuante tem a vantagem de não desperdiçar dígitos que representam zero e, dessa forma, aumenta a gama de valores que podem ser representados com os mesmos dígitos. A amplitude da representação em vírgula flutuante corresponde ao número de dígitos usados no expoente e a precisão corresponde ao número de dígitos da mantissa.

O grande inconveniente dos números em vírgula flutuante é a complexidade das operações e a imprecisão dos resultados. Por exemplo, um resultado que devia ser zero pode ser, após uma operação com vírgula flutuante, algo como 0,00000193. Pequenos erros na precisão dos números podem levar a resultados inesperados. Foi o que aconteceu em 1994, quando o fabricante Intel lançou o seu novo microprocessador Pentium, que tinha um erro (*bug*) nas operações de vírgula flutuante.

1.3. Representação binária

Do exposto nas secções anteriores, concluímos que os números inteiros e reais são a base da representação da informação nos sistemas informáticos. Por sua vez, estes números são representados, com certas limitações, por sequências de dígitos. Cada dígito é um de vários símbolos que representam valores numéricos pequenos: no sistema decimal há dez dígitos. No entanto, os computadores electrónicos utilizam apenas dois dígitos (representação binária), por razões de ordem prática: é muito mais difícil lidar com 10 níveis de tensão eléctrica num circuito. Felizmente, continua a ser possível representar toda a informação com apenas dois dígitos. O número de dígitos de um sistema de numeração designa-se por **base**: o sistema decimal é de base 10 e o sistema binário é de base 2.

Pensemos no processo de contagem na base 10: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. A seguir ao 9 vem 10, isto é, voltamos a usar o dígito 0 mas juntamos-lhe um 1 à esquerda. Quando chegamos a 19, voltamos a 0 mas temos de incrementar o dígito da esquerda: 20. E assim sucessivamente. Quando chegamos a 99, temos de juntar mais um dígito à esquerda: 100. Etc, etc… Na representação binária é a mesma coisa mas só temos dois dígitos. Vejamos como se processa a contagem: 0, 1. Como não há mais dígitos, temos de voltar ao 0, juntando-lhe um 1 à esquerda: 10. Não se deve confundir este número binário que representa dois com o número decimal dez (representado por 10 no sistema decimal). Continuando, temos 11 e depois 100, já que se esgotaram os dígitos em ambas as posições. Depois vem 101, 110, 111, 1000 e assim sucessivamente. A tabela 1 mostra a numeração binária e decimal correspondente para os 10 primeiros números naturais.

Tabela 1: Numeração binária e decimal

Decimal	Binário
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

Cada dígito de uma representação binária é denominado **bit** (abreviatura de *binary digit* – dígito binário). Se dispusermos de N bits, é possível representar $2x2x\ldots x2 = 2^N$ números diferentes. Um grupo de 8 bits é designado por **byte**. O leitor provavelmente já conhece os termos quilobytes, megabytes e gigabytes: tratam-se precisamente das unidades múltiplas do byte. No entanto há uma diferença em relação ao sistema decimal: enquanto que um quilograma são 1000 gramas, um quilobyte são 1024 bytes. A razão desta diferença é que se torna muito mais conveniente usar uma potência de

dois ($1024 = 2^{10}$) nas unidades de contagem de dígitos binários[1], da mesma forma que é conveniente usar potências de 10 ($1000 = 10^3$) no sistema decimal.

As potências de 2 têm um papel essencial na determinação do valor de um número binário. Enquanto que no sistema decimal temos a casa das unidades, das dezenas, das centenas, dos milhares, etc., no sistema binário temos a casa das unidades, dos 2, dos 4, dos 8, dos 16, etc. Estes números são as sucessivas potências de 2 ($1 = 2^0$, $2 = 2^1$, $4 = 2^2$, $8 = 2^3$, $16 = 2^4$) e correspondem ao "peso" de cada dígito binário. Suponhamos que queremos determinar quanto vale o número binário 10011011. Em primeiro lugar, atribuamos a cada dígito o peso correspondente:

1	**0**	**0**	**1**	**1**	**0**	**1**	**1**
128	64	32	16	8	4	2	1
$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Agora, tal como no sistema decimal, basta multiplicar o valor de cada dígito pelo respectivo peso e somar tudo. É claro que no sistema decimal, já está tudo feito, pelo que sabemos intuitivamente quanto vale, por exemplo, 1799 (=1x1000 + 7x100 + 9x90 + 9x1), mas no sistema binário, temos de fazer as contas. A tarefa é, no entanto, simplificada, já que basta somar os pesos dos dígitos 1 (multiplicar um dígito 0 pelo seu peso dá 0). Assim, para o exemplo acima, o resultado é:

$128 + 16 + 8 + 2 + 1 = 155$

O número binário 10011011 vale, então, 155. Como passámos de uma representação em base 2 para uma representação em base 10 (que é mais natural para nós, seres humanos), este procedimento chama-se **conversão de um número binário para decimal**.

A operação inversa, a **conversão de um número decimal para binário** é ligeiramente mais complicada: exige uma série de divisões sucessivas por 2. Vejamos, por exemplo, como se converte o número 232 em binário. Temos de dividir o número por 2 e olhar para o resto da divisão:

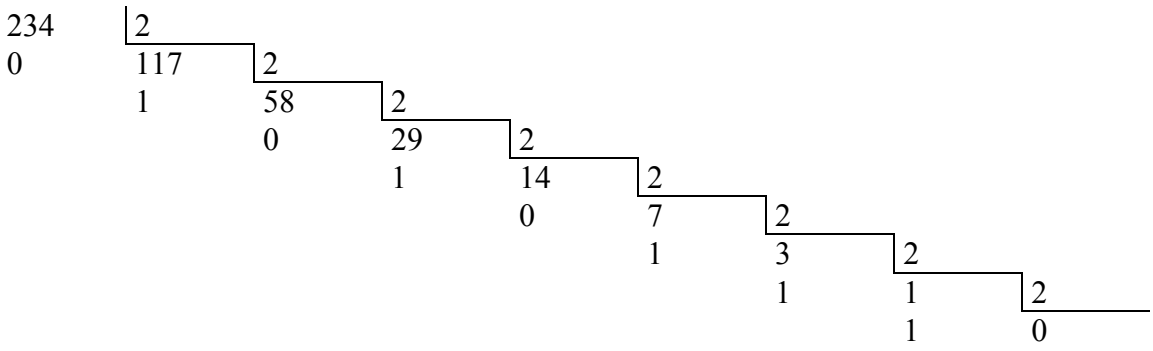
234	| 2
0	117

O resto é 0, pelo que o dígito binário das unidades é 0. Continuemos a dividir o quociente que obtivemos:

234	| 2	
0	117	2
	1	58

Agora obtivemos 1, pelo que o dígito binário seguinte (o de peso 2) é 1. Continuando as divisões sucessivas por 2:

[1] Repare-se que um byte = 8 bits = 2^3 bits



O processo termina quando chegamos a um quociente 0. O número binário que pretendemos é, finalmente, 11101010, que é a sequência, por ordem inversa, dos restos obtidos.

Como vimos, em última análise, toda a informação é representada num computador por uma sequência enorme de zeros e uns. Para além das bases 2 e 10, são importantes outras bases de numeração como 8 e 16, pela sua facilidade de conversão de e para a base 2. De facto, para converter um número na base 2 para uma base 2^N basta agrupar cada N dígitos do número binário e substituí-los pelo correspondente dígito em base 2^N. Por serem muito usadas, estas bases recebem nomes especiais – a base 8 é **octal** e a base 16 é **hexadecimal**. Uma particularidade da representação hexadecimal resulta da necessidade de usar mais de dez dígitos diferentes: os dez algarismos de 0 a 9 mais as letras A, B, C, D, E e F.

1.4. Digital vs. Analógico

A informação representada desta forma diz-se **digital**, já que é constituída por uma série de *dígitos* binários. Mesmo os números reais representados em vírgula flutuante são apenas um conjunto de dígitos. Os números reais têm uma característica, no entanto, que os distingue dos números inteiros: é impossível representar digitalmente, com exactidão, a maioria dos números reais – pensemos, por exemplo, no número π, com a sua sequência infinita de casas decimais que nunca se repete. Este facto tem a ver com a natureza infinita dos números reais, tal como foi descoberta por Cantor: amplitude infinita e precisão infinita.

As grandezas físicas medem-se recorrendo a números reais: os comprimentos são números reais de metros, os tempos são números reais de segundos. A informação veiculada por números inteiros é **discreta**, podendo ser representada de forma digital, enquanto que a informação veiculada por números reais é **contínua**, podendo ser representada apenas aproximadamente em forma digital. Um dispositivo que representa um número real de forma contínua diz-se **analógico**, por exemplo, um termómetro que indica a temperatura através do comprimento de uma coluna de mercúrio. Hoje em dia, também dispomos de termómetros digitais, que indicam o valor numérico da temperatura num visor, com uma precisão de uma ou duas casas decimais. As tecnologias que convertem informação analógica em digital e vice-versa são uma parte fulcral das modernas tecnologias de informação: a imagem num ecrã (analógica)

resultou da conversão da informação digital guardada na memória da placa gráfica do computador; a placa de som converte o som captado analogicamente por um microfone em informação digital. Ao processo de conversão de informação analógica em informação digital chama-se **digitalização** ou **discretização**. Além da gravação digital de som, o processo de leitura de uma imagem num *scanner* também é um exemplo muito conhecido de digitalização.

1.5. Hardware e software

As palavras hardware e software são tão comuns na nossa linguagem quotidiana que quase não vale a pena defini-las: o hardware são os equipamentos físicos e o software são os programas que correm sobre o hardware. À luz dos conceitos introduzidos nas últimas secções, podemos ter uma perspectiva complementar: o software é, de facto, a informação digital manipulada por (e que manipula) os dispositivos físicos.

A questão essencial não é o hardware e o software serem conceitos distintos, mas sim o facto de serem conceptualmente independentes. Em princípio:

a) um pedaço de software (programa) com determinada funcionalidade pode correr em qualquer computador.

b) um computador pode correr qualquer tipo de programa.

O surgimento dos modernos computadores digitais permitiu cumprir o requisito b), já que são máquinas de carácter genérico com capacidade para executar qualquer processo automático (algoritmo) sob a forma de um programa.

As linguagens de programação, desenvolvidas desde os anos 50, respondem ao outro requisito, permitindo codificar um algoritmo de forma independente da máquina onde vai ser executado. Na prática, porém, existem dificuldades relacionadas com as especificidades de cada máquina, a que o programador não pode ser alheio. Estas especificidades fazem com que executar um mesmo programa em máquinas diferentes não seja trivial. Nos anos 90, com a expansão da Internet, foi desenvolvida a linguagem Java que permite a execução de um programa em várias máquinas, interpondo uma "máquina virtual" entre o software e o hardware propriamente dito. A máquina virtual Java não deixa de ser também um pedaço de software mas "esconde" os detalhes do hardware ao programador.

1.6. Definições de informação

Consideremos uma imagem em branco. A informação contida na imagem é quase nula. Do ponto de vista da representação em computador, no entanto, esta imagem irá ocupar o mesmo espaço que uma paisagem, por exemplo, pois ambas são representadas por uma matriz bidimensional de pontos coloridos. Temos a sensação que a imagem em branco está a desperdiçar espaço, já que não contém qualquer informação. É nesta ideia que se baseiam as definições de informação propostas por Claude Shannon em 1948 e por Chaitin e Kolmogorov nos anos 60. A teoria da informação de Shannon permite substituir qualquer sequência de dígitos por outra, mais reduzida, que contém exactamente a mesma informação: por exemplo, a imagem em branco pode ser substituída pela informação de um único ponto branco e pelo número de vezes que esse ponto é repetido na imagem. Os vulgares programas de compactação (WinZip, WinRar,

Arj, etc.) baseiam-se na teoria de Shannon para reduzir o tamanho dos ficheiros, sem qualquer perda de informação.

A teoria da informação algorítmica, desenvolvida por Chaitin e, independentemente, por Kolmogorov, vai mais longe ao dizer que a informação contida numa sequência de dígitos é igual ao tamanho do menor programa de computador que consegue gerar essa sequência. O problema com esta definição é que não é possível construir automaticamente esse programa, tornando impraticável a construção de compactadores com base na teoria de Chaitin. No entanto, a definição de Chaitin é a mais interessante do ponto de vista matemático, já que propõe uma medida absoluta de informação, enquanto que a definição de Shannon mede apenas a informação média.

1.7. Breve História das Tecnologias de Informação e Comunicação

O computador digital, tal como o conhecemos, é uma invenção do século XX. No entanto, houve anteriormente projectos de dispositivos mecânicos para efectuar cálculos e computações. Notavelmente, Schickard, em 1623, Pascal, em 1642 e Leibniz, em 1671 construíram máquinas de calcular que tiveram algum sucesso.

Em 1834, Charles Babbage, professor de matemática em Cambridge, projectou uma máquina chamada "Analytical Engine", onde introduziu as noções de unidade central de processamento e de memória. A máquina era programável mas, pela sua complexidade, Babbage não conseguiu obter financiamento para a sua construção.

Durante o século XX, o computador electrónico surgiu como resultado da investigação em matemática e dos desenvolvimentos tecnológicos no campo da electrónica. No final do século XIX e no início do século XX, colocou-se a questão da formalização da matemática. Após muitos séculos de teorias matemáticas precisas e do desenvolvimento da lógica matemática, os investigadores começaram a interrogar-se se os resultados em matemática não poderiam ser obtidos por um processo puramente automático, através da lógica. Se assim fosse, um computador que aplicasse as regras da lógica a um conjunto de princípios auto-evidentes, poderia obter toda e qualquer conclusão verdadeira em matemática.

Esta "máquina da verdade" acabou por se revelar uma utopia, ao nível da "máquina do movimento perpétuo" que muitos tinham tentado descobrir em séculos anteriores. Foi o matemático Kurt Gödel que frustou as expectativas dos outros investigadores, ao demonstrar, em 1931, que é impossível formalizar qualquer teoria que envolva conjuntos infinitos: nem sequer a aritmética de números inteiros pode ser formalizada de forma completa. Este resultado deitou o sonho da "máquina da verdade" por terra, mas ironicamente, abriu o caminho para as bases teóricas do computador.

O matemático Alan Turing reformulou o teorema de Gödel em termos de um dispositivo teórico a que chamou "máquina de Turing". Inicialmente, o dispositivo continha uma fita infinita com zeros e uns, e um programa, que dirigia as operações da máquina. Cada máquina tinha um programa diferente. A ideia revolucionária de Turing foi codificar o programa sob a forma de zeros e uns na fita. Desta forma, a máquina lia o programa na fita, interpretava-o e depois seguia as respectivas instruções. Deixou de haver necessidade de uma máquina específica para cada programa: uma única máquina executava qualquer programa que lhe fosse fornecido na fita. A esta máquina chamou-se **máquina universal de Turing** e constitui o modelo teórico dos computadores que usamos actualmente. Turing traduziu o resultado de Gödel em linguagem computacional da seguinte forma: não existe nenhum processo automático (leia-se

programa) capaz de determinar se um programa da máquina universal de Turing termina ou não a sua execução.

As ideias de Turing foram desenvolvidas por John von Neumann que propôs a arquitectura em que os programas e os dados são guardados na mesma memória do computador. Os primeiros computadores baseados neste princípio foram construídos nos anos 40, usando a tecnologia das válvulas electromecânicas. A invenção do **transístor**, em 1947, proporcionou uma revolução na tecnologia dos computadores, dando origem à chamada segunda geração de computadores. As válvulas foram substituídas por transístores, componentes mais pequenos e mais fiáveis. A invenção que deu origem à geração seguinte (3ª) de computadores foi o **circuito integrado**: um componente que incorpora um circuito electrónico numa pequena pastilha de silício. Os primeiros computadores à base de circuitos integrados foram construídos nos anos 60, dando origem à proliferação de computadores verificada hoje em dia. Os anos 70 viram o nascimento do microprocessador, lançado pela companhia Intel e, em 1974 surge o primeiro computador de uso doméstico. Em 1981 a IBM lança o IBM PC, do qual derivam quase todos os computadores que temos hoje em cima das nossas secretárias.

Durante os anos 80 e 90, as tecnologias que mais contribuíram para a expansão das TIC foram os sistemas operativos de janelas e a Internet. Os sistemas operativos de janelas tornaram mais acessível e imediato o uso do computador: a Apple foi pioneira nesta área, introduzindo o sistema MacOs em 1984, que acompanhava o seu novo computador Macintosh. No ano seguinte, a Microsoft lançou a primeira versão do Windows, que evoluiu para o sistema operativo que é hoje usado na grande maioria dos computadores pessoais tipo PC.

A Internet foi montada em 1969 (na altura chamava-se ARPANET) e inicialmente ligava os computadores do departamento de defesa norte-americano. No final dos anos 70, a ARPANET foi libertada para uso civil e os primeiros utilizadores foram as grandes universidades. Só em meados dos anos 80 é que começaram a surgir os primeiros serviços de Internet que agora usamos: os nomes de domínio (e.g. microsoft.com ou univ-ab.pt) e os protocolos de e-mail e transferência de ficheiros.

A World Wide Web (WWW) é o serviço mais familiar da Internet e o responsável pela explosão da Internet nos anos 90. A WWW foi inventada em 1989 por Tim Berners-Lee, um físico do CERN (Centro Europeu de Pesquisa Nuclear), como forma de partilhar documentos hiper-ligados entre a comunidade de investigação em física.

Obviamente que estes desenvolvimentos de software não seriam possíveis sem o correspondente avanço do hardware: embora a ideia básica se mantenha desde os anos 40, os computadores tornaram-se muito mais rápidos, com muito mais capacidade de armazenamento de dados e programas, e mais compactos.

2 - Estrutura e Funcionamento dos Computadores

Organização de um computador

Os computadores são constituídos por três grandes subsistemas: a **unidade central de processamento** (CPU – central processing unit), a **memória** e os **dispositivos de entrada e saída** (I/O – input/output).
Para funcionar, o computador precisa de um motor que faça progredir o processamento da informação, de espaço para armazenamento dessa informação e de mecanismos para receber e transmitir informação de/para o exterior. Cada uma destas funções corresponde a um subsistema. A ligação entre os três subsistemas tem de ser muito eficiente, já que existem transferências intensivas de informação durante o funcionamento do computador. A figura 1 mostra os três sub-sistemas e os canais de transferência de informação entre eles. Cada canal de transferência de informação é genericamente designado por ***bus*** (barramento, nas traduções mais comuns para português).

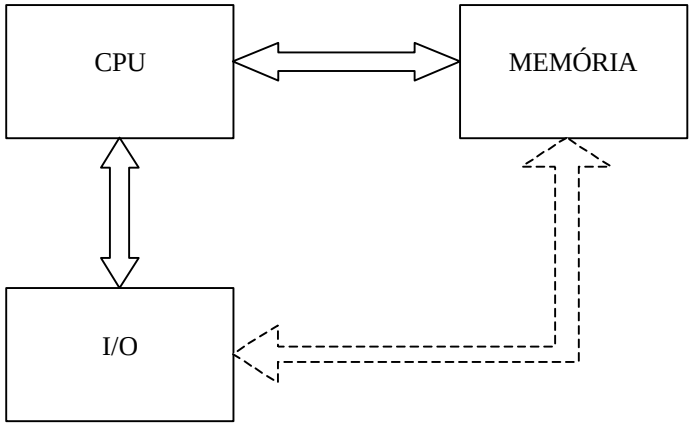


Figura 1 - Arquitectura do computador

O CPU está encarregue da gestão da memória e do subsistema de I/O. No entanto, por vezes há interesse em que o subsistema de I/O aceda à memória sem passar pelo CPU. Por exemplo, o carregamento de informação de um CD para memória pode ser feito directamente. Para isso, criaram-se canais privilegiados de acesso à memória, chamados DMA (direct memory access), tendo em conta que a grande maioria das operações de um computador se resume a transferir informação de um lado para o outro. O DMA permite libertar o CPU para realizar tarefas menos "monótonas", e melhorar o desempenho geral do sistema.
No caso dos computadores pessoais que usamos quotidianamente, o CPU e a memória encontram-se na chamada "**motherboard**", uma placa de circuitos integrados que constitui o elemento mais central do computador. Geralmente, as "motherboards" actuais já permitem a substituição do CPU e várias configurações de memória através da inserção de placas mais pequenas em suportes apropriados. Os dispositivos de entrada e saída (unidades de disco, teclado, rato, ecrã, impressora, microfones, altifalantes) são ligados, directamente, à "motherboard" através de tomadas próprias ou,

indirectamente, via placas de expansão que são inseridas em suportes apropriados. Estes suportes estão ligados a um *bus* de acesso ao CPU. As placas de expansão mais comuns são a **placa gráfica** (ou placa de vídeo), que faz a ligação ao monitor, e a **placa de som**, onde são ligados os altifalantes e o microfone.

Unidade Central de Processamento

A unidade central de processamento (CPU) ou processador é o motor de um computador. Há quem diga, mais poeticamente, que é o "coração" ou o "cérebro" do computador.
O processador é essencialmente uma máquina de estados que executa instruções a um determinado ritmo. O estado do processador em cada momento é determinado por um conjunto de registos, que mais não são do que pequenos espaços de armazenamento de dados (à semelhança da memória, mas internos ao processador). Em cada passo, o processador lê a próxima instrução e executa-a, modifica os seus registos e passa para o estado seguinte.
O ritmo a que o processador executa instruções é determinado pela sua **velocidade**, que se mede em ciclos por segundo. Um ciclo por segundo é uma unidade a que se chama **hertz** (símbolo Hz). Quando se aquire um computador novo, uma das características que os vendedores gostam de apontar é a velocidade do processador, que actualmente andam na ordem dos poucos gigahertz (1 GHz = 1000 milhões de hertz). A razão por que se mede a velocidade em ciclos por segundo e não em instruções por segundo é que as instruções de um processador têm durações de execução diferentes, consoante a sua complexidade, enquanto que um ciclo tem sempre uma duração fixa. A velocidade do processador é imposta exteriormente por um cristal que emite impulsos eléctricos a um ritmo constante – o mecanismo de geração desses impulsos chama-se o **relógio** (clock). Às vezes é possível aumentar a velocidade do processador aumentando o ritmo do relógio (processo chamado *overclocking*), mas é uma operação arriscada, já que o próprio circuito integrado do processador pode não estar desenhado para funcionar a velocidades superiores.

O "fetch cycle"

O processador tem um modo de funcionamento muito simples e que pode ser ilustrado pela seguinte "receita":

Passo 1: ler a próxima instrução
Passo 2: descodificar a instrução lida
Passo 3: executar a instrução
Passo 4: voltar ao passo 1

Este procedimento chama-se "fetch cycle" e é tudo o que o processador faz. Parece incrivelmente simples, especialmente se tivermos em conta que apenas existem duas ou três centenas de instruções diferentes.
Como é que surge então toda a diversidade de funções que um computador exibe? O segredo está na sequência específica de instruções que é fornecida ao computador, sequência essa a que chamamos **programa**. Este aspecto é análogo à diversidade de mensagens que se conseguem obter com apenas 26 letras.

As instruções do processador

Um **programa** de computador é constituído por uma sequência de instruções, cuidadosamente preparada por um programador. Usando uma analogia do tipo "máquina de fazer salsichas", um programa é visto por um utilizador como uma caixa fechada onde entram dados por uma extremidade e saem os resultados pretendidos pela outra extremidade, como se ilustra na figura 2:

Figura 2 - O programa

Com esta perspectiva, não é difícil compreender o tipo de instruções que um processador executa:

- **instruções de transferência de informação**: servem para copiar e mover dados de um lado para o outro: da memória para os registos e vice-versa; dos registos para o subsistema de I/O e vice-versa.
- **instruções de transformação de informação**: são de quatro tipos:
    - **operações aritméticas**: efectuam cálculos numéricos sobre os dados
    - **operações lógicas**: aplicam operadores da lógica aos dados (e, ou, negação lógica)
    - **deslocação e rotação**: alteram a sequência dos bits de um registo ou posição de memória.
    - **comparações**: efectuam comparações (menor, maior, igualdade) sobre os dados
- **instruções condicionais**: conferem capacidade de decisão aos programas permitindo executar uma instrução A se uma determinada condição for verdadeira ou uma instrução B caso contrário. As condições são normalmente baseadas no resultado de uma instrução de comparação.
- **instruções de salto e chamada**: permitem alterar a ordem de execução das instruções saltando para outro ponto do programa. São muitas vezes associadas às instruções condicionais, quando é preciso executar um grupo de várias instruções sob determinada condição.
- **outras instruções**: entrar em modos especiais de funcionamento do processador, parar o processador, não fazer nada durante algum tempo, etc.

A seguinte sequência exemplifica um programa (traduzido para português) que pode ser executado num CPU:

- *copiar* conteúdo da posição 2412 da memória para o registo A
- *somar* 6 ao registo A
- *comparar* o conteúdo do registo A com o número 10
- *se* for menor, *saltar* para a posição <reprovado> do programa
- *transferir* a frase "aprovado" para o écrã

- <reprovado>: *transferir* a frase “reprovado” para o écrã

Repare-se na analogia com a “máquina de fazer salsichas”: os dados entram na posição 2412 da memória e os resultados saem pelo écrã.
Deixa-se como exercício para o leitor a categorização de cada uma das instruções que aparecem neste programa.

A memória

A utilização que fizemos acima do termo “posição” para indicar o local onde se situam tanto dados como instruções do programa não é acidental, mas reflecte o duplo papel da memória num sistema informático: o de armazenar dados e o de armazenar programas. No início da história dos computadores, os dados e os programas eram guardados em memórias diferentes; havia a memória de dados e a memória de programas. A introdução da memória comum para guardar dados e programas constituiu uma mudança muito importante na forma de organizar a informação no computador. A esta nova organização chama-se **arquitectura de Von Neumann**, e todos os computadores actuais se baseiam nela.
A memória de um computador não é mais do que uma sequência de “compartimentos” identificados pela sua “posição” e que contêm informação (ver figura 3). Cada compartimento contém uma quantidade fixa de bits (8, 16, 32 ou 64, conforme o computador). Independentemente do tamanho do compartimento, a capacidade da memória mede-se em bytes (isto é, grupos de 8 bits), sendo mais comum actualmente o uso dos seus múltiplos (kilobytes - Kb, megabytes - Mb, gigabytes - Gb).

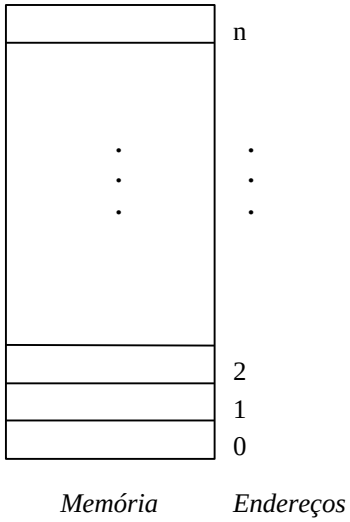


Figura 3 - A memória

A posição de um compartimento é designada tecnicamente por **endereço**. A comunicação entre o processador e a memória é sempre feita nos seguites termos: “dá-me os dados que estão no endereço x” ou “coloca os dados y no endereço z”. A informação que circula entre o processador e a memória é, portanto, de três tipos:

- endereços
- dados
- informação de controlo

A informação de controlo corresponde ao tipo de operação que o processador quer executar na memória: ler dados ou escrever dados. Os três tipos de informação são transmitidos em simultâneo, pelo que o bus que liga o processador à memória está dividido em três partes: o **bus de endereços**, o **bus de dados** e o **bus de controlo**. Adiante veremos que o bus de controlo pode ter funções adicionais.

Memória secundária

A memória de que falámos até agora designa-se por memória principal e tem a vantagem de ser acessível com grande rapidez pelo processador. Infelizmente, como é constituída por circuitos electrónicos, só cumpre a sua função (a de memorizar os dados) enquanto for alimentada por corrente eléctrica. Assim que se desliga o computador, este sofre uma "amnésia" que desagradavelmente sentimos quando perdemos o nosso trabalho mais recente após uma falha de corrente eléctrica.
Para colmatar este inconveniente, existem dispositivos de **memória secundária** (o disco, as disquetes, os CDs, os DVDs) que armazenam dados de forma mais duradoura. Infelizmente, também têm um aspecto menos bom: o acesso processa-se de uma forma muito mais lenta. O computador normalmente usa a memória principal e só ocasionalmente transfere dados de/para a memória secundária.
Por causa desta organização, os utilizadores de computadores sabem que devem frequentemente "gravar" o seu trabalho (isto é, transferir a informação que compilaram para um dispositivo de memória secundária), não vá o diabo tecê-las.
Dos vários dispositivos de memória secundária, o **disco duro** (ou só *disco*, como é vulgarmente conhecido) é o mais importante. Está normalmente instalado no interior da caixa do computador, tem uma capacidade muito grande (medida em Gb) e uma velocidade de acesso mais rápida que os outros dispositivos de memória secundária.
As disquetes são muito utilizadas (embora cada vez menos), pois podem ser transportadas de computador para computador. O inconveniente das disquetes é que têm mantido a mesma capacidade (máximo: 1,44Mb), mesmo com a evolução do tamanho de programas e dados, o que as torna pouco práticas para transferir ficheiros maiores. Por essa razão têm sido usados outros dispositivos como as "zip drives" e as "pen disk" que cumprem a mesma função mas têm capacidades muito maiores. Até os CDs e DVDs (graváveis e regraváveis) são utilizados para transferir informação entre computadores, apesar da maior lentidão de acesso. Os CDs e DVDs são preferíveis quando se pretende guardar informação de forma permanente (por exemplo, um trabalho acabado) uma vez que têm uma durabilidade bastante maior que as disquetes e os outros dispositivos e não são influenciáveis por campos magnéticos.

Dispositivos de entrada e saída

Os dispositivos de entrada e saída são controlados pelo processador da mesma forma que a memória, ou seja, via um bus de endereços, um bus de dados e um bus de controlo. Na realidade, são os mesmos buses de acesso à memória, e existe uma linha

do bus de controlo que indica se o processador está a aceder à memória ou a um dispositivo de I/O.
Cada dispositivo está associado a um endereço onde o computador lê ou escreve informação. Nalguns dispositivos só se pode ler informação (o teclado, o rato, o CD-ROM); noutros só se pode escrever (o ecrã, a impressora); noutros são permitidas as duas operações (o disco, as disquetes).
Existe, no entanto, uma diferença fundamental entre a memória e os dispositivos de entrada/saída. Enquanto que a memória é passiva, ou seja, não toma nunca a iniciativa de comunicar com o processador, os dispositivos de entrada e saída podem ser activos, iniciando processos de comunicação com o processador. Por exemplo, quando se carrega numa tecla ou se mexe o rato, os dispositivos de entrada/saída correspondentes precisam de indicar ao processador que ocorreu um evento. A forma que os dispositivos têm para fazer esta indicação é através de um pedido de interrupção. Um **pedido de interrupção** (*interrupt request*) é efectuado por um canal especial que liga um dispositivo de entrada/saída ao processador, a que se chama linha de IRQ (interrupt request). Quando o processador recebe um pedido de interrupção, pára temporariamente o que estava a fazer e executa um pequeno programa que atende o pedido. Por exemplo, se o pedido de interrupção ocorreu porque o utilizador mexeu o rato, o processador actualiza a posição do ponteiro no écrã e volta novamente àquilo que estava a fazer.

Conflitos de hardware

Por vezes, ao instalar um novo componente de hardware no computador, surgem conflitos com outros dispositivos que impedem que o hardware funcione correctamente. Estes conflitos podem surgir quando há coincidência nas seguintes especificações para dispositivos diferentes:

- pedidos de interrupção (IRQ)
- endereços dos dispositivos de entrada/saída
- canais de DMA
- memória principal usada pelos dispositivos

Por exemplo, se dois dispositivos tiverem a mesma linha de IRQ, poderá surgir um conflito. No entanto, nem sempre isso acontece. Há casos em que dois dispositivos podem perfeitamente partilhar a mesma linha de IRQ sem haver problemas. Em geral, o sistema operativo sabe informar se dois dispositivos estão em conflito ou não.
Se houver conflito, a forma de o resolver é atribuir outra linha de IRQ ao dispositivo em questão (ou endereço de I/O, ou canal de DMA ou seja o que for que está a causar o conflito). A resolução nem sempre é pacífica, já que em alguns casos, os dispositivos têm parâmetros definidos de fábrica que impedem que sejam alterados.

3 - Sistemas Operativos

O **sistema operativo** (SO) é a primeira camada de software de um sistema informático. Actualmente, todo o restante software (chamado **software de aplicação**) está dependente do SO.
Os SO mais antigos (por exemplo, o MS-DOS) não obrigavam as aplicações a dependerem do SO, e muitos programas manipulavam directamente o hardware do computador, por razões de eficiência. Hoje em dia, este tipo de comportamento não é aconselhado, principalmente por duas razões. Primeiro, porque o hardware varia muito de máquina para máquina e não é viável escrever n versões de um mesmo programa para acomodar diversos tipos de hardware. Por outro lado, a manipulação directa dos recursos de hardware pode levar a que programas construídos com fins maliciosos (como os vírus) possam afectar directamente esses recursos.
Os actuais sistemas operativos (exemplos: Windows, Linux, MacOS) cumprem de forma eficaz as duas funções essenciais dos sistemas operativos, que são as seguintes:

1. Disponibilizar uma máquina virtual, que é uma extensão da máquina física.
2. Gerir os recursos do sistema.

Além de cumprirem estas funções, os actuais SO não permitem que um programa aceda directamente ao hardware nem que faça uma gestão própria dos recursos. Por exemplo, um programa que tencione utilizar o espaço de memória onde está alojado um outro programa do sistema é impedido de efectuar essa operação. Quem utiliza o sistema operativo Windows já deparou com certeza com a mensagem "Este programa efectuou uma operação ilegal e será encerrado", os utilizadores de Unix/Linux estão mais habituados à mensagem "Segmentation fault" – estas mensagens significam que o programa tentou aceder a um recurso (frequentemente, a determinados endereços de memória) ao qual não tem direito. Estas mensagens não significam necessariamente que o sistema tem um vírus, já que é muito fácil um programador cometer um erro que faça com que o programa, em determinadas circunstâncias, tente executar uma operação ilegal. Infelizmente, o controlo de qualidade no software (assim como noutras áreas) não impede que todos os programas produzidos sejam isentos de erros.
Voltando às funções do SO, a disponibilização de uma máquina virtual facilita a tarefa do programador, na medida em que este não tem de se preocupar com os detalhes de funcionamento de cada pedaço de hardware, nem com a gestão da memória ou do disco. Por outro lado, os serviços que o SO disponibiliza para estas tarefas podem ser consideradas muito limitativas, pelo que tem de haver um certo equilíbrio entre flexibilidade e segurança.

3.1. A "shell"

Até agora falámos da relação entre o SO e o programador, mas a maioria dos utilizadores só se apercebe do aspecto exterior que o SO exibe, a chamada "**shell**". Nos SO actuais essa shell é essencialmente uma interface gráfica (GUI – graphical user interface), em que os objectos – programas e ficheiros - são manipulados através de uma

representação pictórica (ícones) no ecrã. No entanto, a interface com o utilizador não tem de ser gráfica e há muitos utilizadores que se sentem mais confortáveis com uma interface de linha de comandos textuais. As interfaces de linhas de comandos eram vulgares nos tempos do MS-DOS, já que a tecnologia para mostrar gráficos no ecrã ainda estava pouco desenvolvida. Mesmo assim, ainda têm muita utilidade para quem pretende ter um controlo mais fino sobre os programas que são executados no computador. Muitos utilizadores do SO Linux, por exemplo, não prescindem da utilização de uma shell de comandos, como o "bash", que tem uma infinidade de funções para edição e manipulação de comandos que é difícil encontrar noutro tipo de shell.
A shell é uma componente do sistema operativo que é independente das restantes componentes, no sentido em que é possível substituí-la por outra, mais conveniente. Esta facilidade é mais conhecida dos utilizadores Linux, já que têm à escolha entre vários tipos de shell, tanto interfaces gráficas (KDE, Gnome), como linhas de comandos (c-shell, bash, ksh). Mesmo o Windows permite definir diferentes temas e configurações na sua interface gráfica.

3.2. Tipos de SO

Existem muitos tipos de SO, consoante o tipo de funcionamento que se pretende. Estamos mais habituados aos sistemas operativos "para toda a obra" utilizados nos computadores pessoais, embora existam sistemas operativos específicos para computadores de bolso (PDAs), telemóveis, redes de computadores, dispositivos com restrições críticas de tempo (real-time), etc.
Os sistemas operativos são classificados quanto ao número de utilizadores que comportam, em **monoutilizador** e **multiutilizador**. Os sistemas monoutilizador estão preparados para suportar apenas um utilizador e não promovem a separação efectiva dos dados e processos de vários utilizadores. São exemplo de sistemas monoutilizador o sistema MS-DOS e versões anteriores do Windows.
Os sistemas multiutilizador estão contruídos para suportar vários utilizadores e efectivamente separam os ficheiros de cada utilizador, bem como os programas que cada utilizador executa. Garantem a protecção dos dados de cada utilizador, impedindo que outros utilizadores consultem e/ou alterem os respectivos ficheiros, e permitem que cada utilizador execute apenas os programas para os quais possui autorização. Exemplos de SO multiutilizador são as mais recentes versões do Windows e o Unix e suas variantes, como o Linux.
Nos sistemas multiutilizador, existe um utilizador com mais privilégios do que os outros, o chamado **super-utilizador**, que tem autorização para executar programas que configuram e alteram o próprio sistema operativo, que gerem os recursos do sistema, e que definem as características dos outros utilizadores. A função de super-utilizador é normalmente desempenhada por um administrador de sistemas altamente qualificado, dados o poder que possui. Por exemplo, pode a qualquer momento apagar ficheiros do sistema, criar/eliminar contas de utilizador, encerrar programas que estão a correr, etc.
Por uma questão de segurança, é sempre desejável que, mesmo para o administrador do sistema, qualificado ou não, estes privilégios estejam sempre disponíveis, já que é muito fácil cometer um erro com consequências desastrosas (como apagar todos os ficheiros, por exemplo). Um administrador de sistema deve sempre ter uma conta de super-utilizador onde realiza todo o trabalho de administração do sistema que necessite de

privilégios e uma outra conta de utilizador normal onde realize o seu trabalho que não necessita de privilégios especiais.
Independentemente do número de utilizadores, um SO é classificado quanto ao número de programas que pode executar ao mesmo tempo, em **monoprocessamento** e **multiprocessamento**. Um SO monoprocessamento só executa um programa de cada vez. Um segundo programa só pode ser iniciado depois de o primeiro ter terminado. Um SO multiprocessamento pode ter em execução vários programas simultaneamente. No entanto, um CPU, como vimos no capítulo anterior, funciona sequencialmente, isto é, uma instrução de cada vez. Como é que um SO em multiprocessamento pode executar vários programas ao mesmo tempo? Há basicamente duas formas de o fazer: ou existem vários CPUs e os programas são distribuídos pelos CPUs ou só há um e terá de haver uma partilha do tempo que o CPU dedica a cada programa. Esta segunda solução designa-se por "**time-sharing**" e é a solução adoptada nos vulgares computadores pessoais com SOs como Linux e Windows.
A conjugação do multiprocessamento com um sistema multiutilizador permite que vários utilizadores estejam a correr vários programas no mesmo computador, sem que nenhum deles se aperceba necessariamente dos outros.

3.3. Ficheiros

A gestão dos ficheiros no disco é uma das componentes mais importantes do SO. O subsistema que gere os ficheiros chama-se **sistema de ficheiros** (filesystem). Tal como a shell, os actuais SOs disponibilizam vários sistemas de ficheiros, embora só seja possível escolher um para cada disco no processo de instalação do SO. Exemplos de sistemas de ficheiros são o NTFS (New Technology File System), o FAT32, ext2fs, o HFS (Macintosh Hierarchical File System) ou o nfs (network file system). Muitas vezes um mesmo sistema de ficheiros é usado em sistemas operativos diferentes, fazendo com que uma unidade de disco normalmente usada por um SO possa ser utilizada por outro SO. Por exemplo, num computador onde estejam instalados dois sistemas operativos (embora só seja possível utilizar um de cada vez) é útil que os ficheiros criados num dos SO possa ser acedido através do outro SO. Isto só é possível se ambos possuírem o mesmo sistema de ficheiros.
Os ficheiros em disco são normalmente arrumados em **directorias** ou **pastas**, que se organizam numa estrutura em árvore (ver figura 1). Uma cadeia de pastas é designada por **caminho** (path), e normalmente representada pelos nomes das pastas separadas por / ou \. Por exemplo, documentos/curso/informatica representa um caminho que começa na pasta documentos, passa pela pasta curso que está contida dentro de documentos e termina na pasta informatica que está contida em curso.

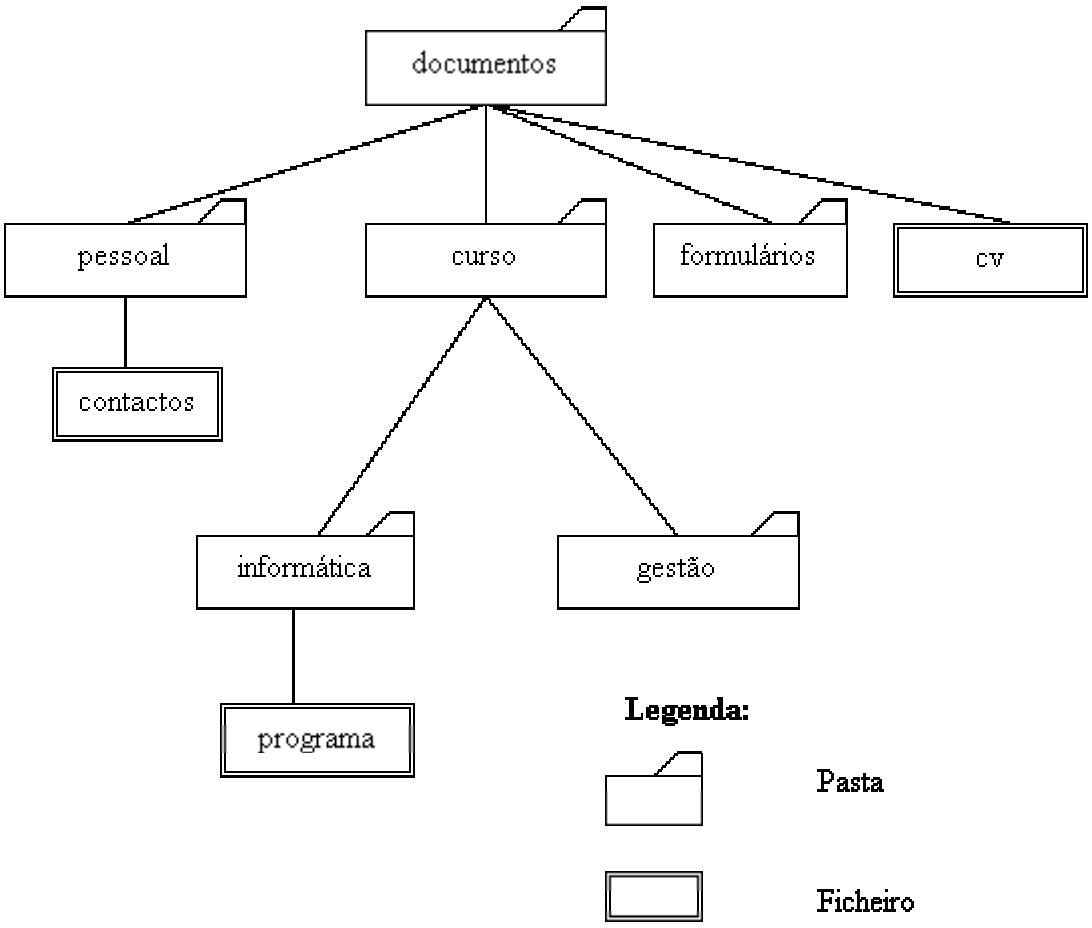


Figura 1 - Árvore de pastas e ficheiros

Compete ao sistema de ficheiros gerir esta estrutura de pastas e ficheiros. Obviamente, esta estrutura é virtual, não correspondendo à forma como está organizado o disco, fisicamente. O disco é, tal como a memória central, uma sequência linear de espaços para guardar bytes, acessíveis através de um endereço numérico.

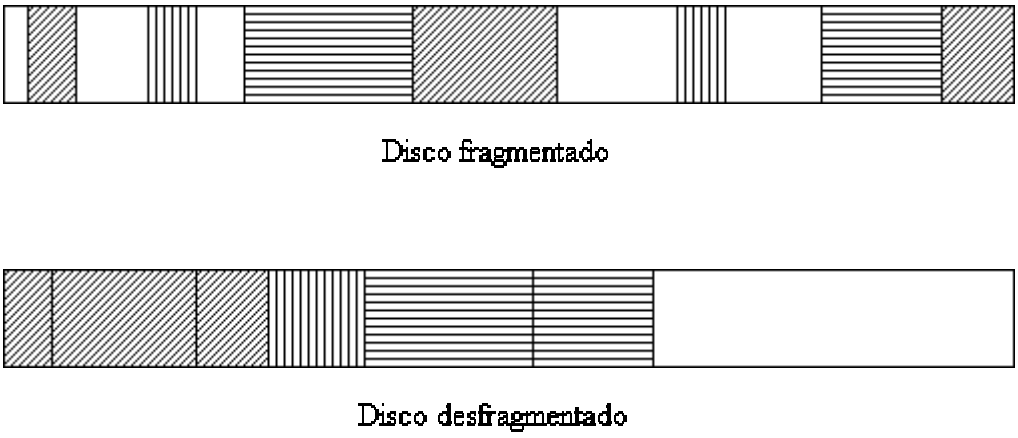


Figura 2 - Disco rígido com e sem fragmentação

O sistema de ficheiros fornece ao utilizador uma perspectiva dos dados em disco segundo a estrutura da árvore de pastas, facilitando a arrumação e organização da informação. O utilizador não tem de se preocupar com os endereços de disco onde estão os seus dados, acedendo aos ficheiros através do seu nome e da posição que ocupam na árvore de pastas. No entanto, esta gestão automática por parte do sistema de ficheiros pode gerar uma situação de alguma desarrumação no disco e que fazem com que se demore mais tempo ao manipular ficheiros. A esta condição chama-se **fragmentação** do disco, e corresponde ao facto de os fragmentos dos vários ficheiros e pastas estarem distribuídos pelo disco com muitos espaços não utilizados entre eles, mas demasiado pequenos para albergar um ficheiro com alguma dimensão. O desempenho do sistema de ficheiros degrada-se nesta situação, e a solução é utilizar um programa, chamado desfragmentador, que arruma os ficheiros no disco e junta todos os espaços vazios, formando uma zona em disco onde cabem ficheiros maiores. A figura 2 ilustra esta situação para o caso simples de três ficheiros, antes e depois de aplicado o desfragmentador.

3.4. Processos

Um **processo** é definido como um programa em execução. Um processo contém informação sobre a posição corrente no programa a ser executado, bem como sobre os conteúdos dos registos e posições de memória a serem utilizados pelo programa. Isto é, um processo contém informação sobre o estado de execução do programa.
É possível co-existirem vários processos a correrem o mesmo programa, uma vez que se distinguem pelo respectivo estado de execução. A cada processo é atribuído um número identificador único que permite ao sistema operativo controlar o conjunto de processos simultaneamente em execução. Num sistema multiprocessamento, em que os vários processos têm de partilhar o CPU, o sistema operativo atribui, rotativamente, um período de tempo a cada processo, findo o qual guarda o seu estado de execução e passa ao processo seguinte na fila. Quando chega novamente a vez do primeiro processo, o SO repõe o estado de execução guardado e volta a colocar o processo em funcionamento. Num sistema suficientemente rápido, o método rotativo transmite a ilusão de que todos os processos correm ao mesmo tempo. Esta ilusão é formalizada na noção de **processador virtual**, em que cada processo corre no seu próprio processador, embora este na realidade corresponda a uma fatia de tempo do processador real.
Cada processo utiliza recursos do sistema, como a memória, ficheiros em disco, periféricos, que, num sistema multiprocessamento podem estar a ser utilizados simultaneamente por outros processos.
Os processos também podem comunicar entre si e partilhar por mútuo acordo alguns recursos. A partilha de recursos pode originar conflitos entre os processos, a que se chamam problemas de **concorrência**. A resolução deste tipo de problemas é crítica para o bom funcionamento do sistema e é da responsabilidade não só do sistema operativo mas também do programador. Um dos mecanismos à disposição do programador para evitar um conflito é o **semáforo**, que tem uma funcionalidade semelhante ao semáforo de trânsito. Quando um programa pretende utilizar um recurso de forma exclusiva (por exemplo, alterar uma determinada secção de uma base de dados, com a garantia de que mais nenhum outro processo lê ou altera essa mesma secção), deve assinalar com um semáforo que mais ninguém deve executar operações sobre esse recurso. Quando o

programa termina a utilização do recurso, deve libertá-lo, apagando o semáforo. O sistema operativo disponibiliza e verifica os semáforos, garantindo a **exclusão mútua** de processos na região crítica.
Um outro problema que resulta da competição entre processos é o chamado "**deadlock**", ou o impasse perante recursos bloqueados. O deadlock pode ser ilustrado pela seguinte situação: um processo que pretenda utilizar a impressora, bloqueia-a com um semáforo e fica à espera que o ficheiro a imprimir, que está a ser utilizado por outro processo, fique desbloqueado. Por sua vez, o segundo processo bloqueou o ficheiro e está à espera que o primeiro processo desbloqueie a impressora: cada um dos processos está à espera do outro e ficam ambos parados. Uma forma de resolver um deadlock é eliminar um dos processos, permitindo ao outro prosseguir. Mais tarde, o processo eliminado será reiniciado. Outra forma, que é utilizada no caso das impressoras, é o "**spooling**". Neste método, não é permitido a um processo bloquear a impressora, mas a informação a imprimir deve ser enviada a um processo especial (o spooler) que guarda essa informação em disco até a impressora estar disponível. O processo que pretende imprimir pode continuar a sua execução, ficando a impressão a cargo do spooler.

3.5. Gestão de memória

Um dos recursos mais importantes e simultaneamente mais escassos, é a memória central. De facto, a velocidade de acesso faz com que seja a forma mais eficaz de temporariamente guardar a informação associada a um programa (incluindo o próprio programa), enquanto este está a ser executado. Mas a memória central também é uma componente extremamente dispendiosa, especialmente quando comparada com dispositivos de memória secundária como discos rígidos, CDs e DVDs. Ao longo da história dos computadores, sempre se verificou uma grande discrepância na capacidade de armazenamento da memória central e da memória secundária. No entanto, os tempos de acesso a memória secundária são muito superiores aos tempos de acesso a memória central.
A componente do sistema operativo que se encarrega de gerir este compromisso entre o uso de memória central e secundária é o **gestor de memória**.
Nos sistemas multiprocessamento, em que vários processos possuem informação para alojar na memória central, é natural que por vezes não haja memória suficiente para todos os processos. O gestor de memória coopera com o gestor de processos no sentido de guardar em memória a informação associada aos processos em execução, copiando temporariamente em disco (memória secundária) a informação associada a processos que não estão em execução. Este processo de cópia é designado por "**swapping**" e o espaço em disco que guarda a informação designa-se por ficheiro de "**swap**". O swapping liberta espaço na memória para alojar informação para outro processo entretanto posto em execução. Se um sistema tem muitos processos, o desempenho degrada-se quando a memória é pouca, devido às frequentes cópias de informação entre a memória e o disco e vice-versa. Esta degradação traduz-se numa maior lentidão do funcionamento geral do sistema, mas o processo de swapping tem o mérito de manter o sistema em funcionamento, mesmo com pouca memória.
Outro problema com a gestão de memória surge quando um único programa necessita de mais memória do que aquela que está disponível. O sistema operativo, através do gestor de memória, inclui um mecanismo de **memória virtual** que permite que tal

programa seja executado. A memória virtual é uma abstracção que faz com que qualquer programa tenha a ilusão de ter a memória suficiente para funcionar. Na realidade, o que o sistema operativo faz é manter parte da informação associada ao programa em memória central e outra parte em disco.
A técnica mais usada de memória virtual é a **paginação**, através da qual toda a informação associada a um programa é dividida em partes relativamente pequenas a que se chamam **páginas**. Existem várias estratégias para decidir quais as páginas que ficam em memória e quais as que são alojadas no disco, mas qualquer estratégia minimamente eficiente guarda em memória as páginas que são mais frequentemente usadas enquanto que no disco ficam as que só ocasionalmente são acedidas.
Todas estas técnicas são razoavelmente complicadas e o facto de o sistema operativo se encarregar delas permite ao programador concentrar-se na lógica do problema que está a resolver em vez de se preocupar com questões técnicas de gestão de memória.

3.6. Gestão de dispositivos

Sendo o sistema operativo a camada de software que faz a ponte entre o hardware e o sofware de aplicação propriamente dito, resta saber como é que são tratados os dispositivos de entrada e saída.
Enquanto que os processadores, as memórias e os discos têm um grau elevado de compatibilidade entre si em computadores do mesmo tipo, o sistema operativo pode ter em conta, sem grandes problemas, a relativa variedade destes tipos de componentes.
Já o mesmo não se passa com a enorme variedade de dispositivos de entrada e saída que um computador pode controlar. Os dispositivos variam bastante em tipo e em fabricante, e não é nada prático que o sistema operativo instalado num computador tenha em conta todas as possibilidades de periféricos que podem estar ligados.
Este problema é resolvido por recurso a componentes do SO que se designam por **gestores de dispositivos** (device drivers), tipicamente um para cada dispositivo a usar no sistema. Estes gestores de dispositivos são normalmente fornecidos pelos fabricantes e especificamente desenhados para cada sistema operativo que os suporta. Actualmente, para facilitar a instalação, os gestores de dispositivos mais comuns já vêm incluidos com o sistema operativo. No entanto, em caso de problemas de funcionamento, deve-se sempre obter a versão mais recente do gestor do dispositivo problemático junto do respectivo fabricante.
Os gestores de dispositivos têm um papel importante a cumprir na função 1 do SO que vimos acima (disponibilizar uma máquina virtual), porque apresentam ao programador uma funcionalidade uniforme para cada tipo de dispositivo. Por exemplo, ao construir um programa para imprimir um documento, o programador não precisa de conhecer os detalhes da impressora que vai ser usada, e só tem de considerar o "dispositivo virtual" fornecido pelo gestor de dispositivos da impressora. Além de facilitar o trabalho do programador, este método tem a grande vantagem de não ser necessário alterar o programa para que funcione com outra impressora: basta modificar o gestor de dispositivos da impressora, fornecido pelo respectivo fabricante.

3.7. "Boot strapping"

O sistema operativo controla todo o hardware, gere os processos e coordena as actividades do computador, como vimos nas secções anteriores. Vamos agora considerar o problema de como é que o próprio SO é lançado inicialmente.
O sistema operativo é uma colecção de programas, muitos dos quais têm de ser lançados quando se liga o computador. Estes programas residem em disco, e têm de ser carregados em memória antes de serem executados, como qualquer outro programa.
Parece um problema sem solução, uma vez que é o próprio sistema operativo que se encarrega do carregamento dos programas em memória: é um problema do tipo "pescadinha de rabo na boca".
Mas a solução existe e reside num tipo de memória que ainda não mencionámos, de sigla ROM (read-only memory). Esta memória, por oposição à memória de tipo RAM (random access memory), de que temos falado até agora, não perde a sua informação quando a energia eléctrica é desligada. De facto, a memória ROM é gravada de fábrica e o seu conteúdo é fixo. É esta ROM que contém o primeiro programa a ser executado pelo computador, a que se chama programa de "**bootstrapping**" ou, abreviadamente, programa de "boot": é um pequeno programa que, após consulta de uma tabela situada no disco rígido, carrega o sistema operativo para a memória RAM. A tabela consultada pelo programa de boot (designada nos PCs por MBR – master boot record) contém informação sobre os sistemas operativos disponíveis e onde se situam. O MBR permite que o utilizador possa escolher entre vários sistemas operativos instalados no disco rígido. Por exemplo, é possível instalar um SO Windows e um SO Linux no mesmo computador, e o programa de "boot" encarrega-se de pedir ao utilizador qual pretende usar, após o que o sistema pretendido é carregado em memória. Este tipo de instalação tem a designação de "dual boot".

4 - Comunicação de dados e redes

O tratamento da informação por máquinas, a que chamamos computadores, torna muito mais fácil o manuseamento de grandes quantidades de dados. Por outro lado, também coloca o problema da transmissão desses dados entre computadores de uma forma prática. A comunicação de dados deve ser fiável, robusta e segura.
O objectivo desta unidade é descrever as principais tecnologias de hardware e software usadas na comunicação de dados.

4.1. Redes

Uma rede de computadores é um sistema em que vários computadores estão ligados, seja por cabos ou por sinais electromagnéticos, uns aos outros. Do ponto de vista da distância a que os computadores se encontram uns dos outros, as redes classificam-se em **locais** (LAN – local area network), **metropolitanas** (MAN – metropolitan area network), ou de **larga escala** (WAN – wide area network). As distâncias envolvidas em cada um destes tipos de redes resumem-se na seguinte tabela:

Tipo de rede	Distâncias
LAN	0-10 km
MAN	10-100 km
WAN	> 100 km

As redes locais são normalmente instaladas e geridas por uma organização. A rede de uma empresa, universidade ou instituto, quando se confina a uns poucos edifícios (muitas vezes apenas um), é administrada pela entidade a quem a rede pertence.
No caso das redes metropolitanas e de larga escala, a administração é distribuída por várias entidades. As entidades que possuem as infra-estruturas, as entidades que fornecem serviços de rede e que alugam as infra-estruturas, e os estados em cujo território estão instaladas as redes, que impõem as suas leis.
O exemplo mais imediato de rede de larga escala é a Internet, que conheceu uma enorme expansão em meados dos anos 90. A Internet é mesmo uma rede global, chegando a quase todos os locais do planeta. Não tem nenhuma gestão central, mas baseia-se num conjunto de normas que são aceites por todos os computadores que se lhe ligam. Alguns computadores nesta rede estão permanentemente ligados e disponibilizam informação a que outros computadores na rede podem aceder. Os computadores que fornecem esses serviços são os **servidores**, enquanto que os que acedem à informação são os **clientes** (por exemplo, os vulgares PCs que usamos em casa), e não têm de estar permanentemente ligados à rede.

As redes de computadores podem organizar-se de diferentes formas, sob o ponto de vista das ligações entre os computadores. Um esquema de ligações designa-se por **topologia** de rede, existindo vários tipos de topologias, como se mostra na figura 1.

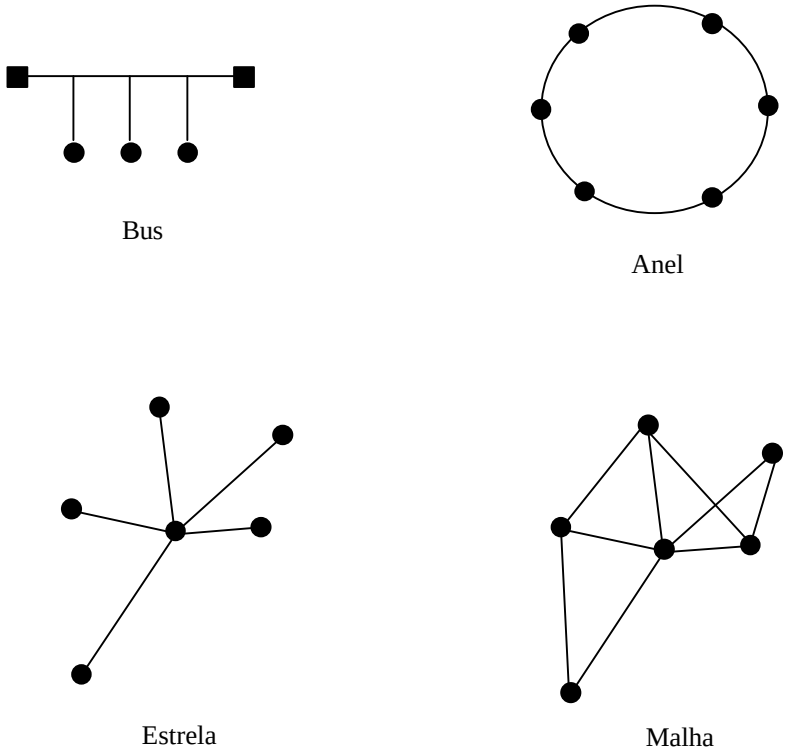


Figura 1 - Topologias de rede

As topologias mais comuns nas redes locais são o **bus** e o **anel**. Estas topologias permitem controlar os custos, que são proporcionais ao número de nós na rede. Numa rede em bus, existe uma linha eléctrica com duas extremidades, à qual se ligam os vários computadores. Os sinais eléctricos que constituem a comunicação estão acessíveis a todos os computadores em qualquer momento. Numa topologia em anel, os computadores estão ligados uns aos outros e uma mensagem enviada por um computador a outro tem de passar por todos os computadores intermédios. A diferença entre as duas topologias é que no bus, a informação está disponível simultaneamente para todos os computadores, enquanto que no anel, a informação é retransmitida de computador para computador.

Nas redes de larga escala, a topologia mais comum é a **malha**, já que permite que existam vários caminhos entre dois nós da rede. No caso de um troço de rede falhar ou um nó não estiver a funcionar, a comunicação entre os restantes nós não é interrompida.

4.2. Modelos de referência

A estrutura física da rede é o suporte de funcionamento de todo o sistema de comunicações. Mas, independentemente da topologia ou do meio utilizado para transmitir a informação, um utilizador de uma rede só precisa de saber que é possível transmitir dados entre um ponto A e um ponto B. Os modelos de referência constroem abstracções sobre a estrutura física por forma a evitar que os utilizadores tenham de se preocupar com os pormenores da infra-estrutura. Pelo facto de a comunicação de dados ser uma tecnologia bastante complexa, estes modelos são organizados em várias camadas de abstracção. O modelo mais utilizado é o modelo **OSI** (Open System

Interconnection), que possui sete camadas. Cada nó da rede funciona com as sete camadas (ver figura 2), e uma mensagem enviada de um nó para outro é transmitida através das camadas, de acordo com interfaces de serviço disponibilizadas pelas camadas inferiores, atravessa o meio físico até ao outro nó e sobe pelo sistema de camadas até chegar à camada superior. Cada uma das camadas preocupa-se com aspectos específicos da comunicação, e funciona de acordo com **protocolos** que permitem que a camada correspondente do nó receptor entenda a mensagem enviada. Desta forma estabelecem-se comunicações virtuais entre as várias camadas dos nós emissor e receptor, como indica a linha tracejada da figura 3.

Figura 2 - Modelo OSI

A primeira camada é a camada **física**, que envolve o meio físico através do qual são transmitidos os dados. As propriedades do sinal eléctrico, velocidade de propagação, ruídos, etc., bem como os sistemas eléctricos e mecânicos de ligação dos computadores à rede são definidas neste nível.

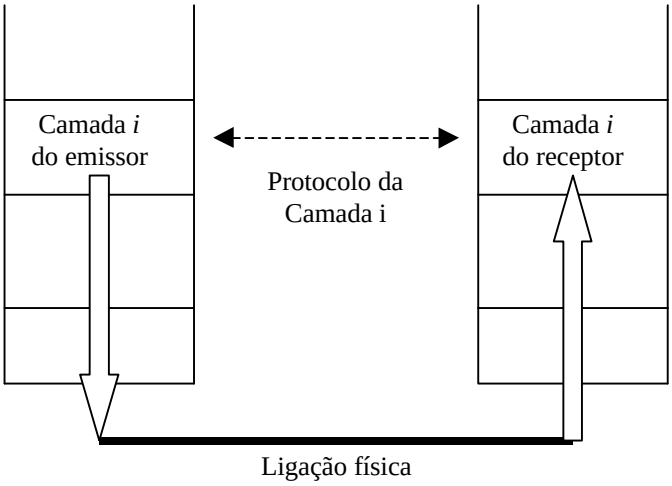


Figura 3 – Comunicação virtual entre camadas do mesmo nível

A camada **lógica** ou de **ligação de dados** é a que permite construir/identificar um pacote com os dados, com a informação do destinatário e controlo de erros. Cada pacote contém delimitadores de início e fim de pacote para que os interlocutores o possam reconhecer. O controlo de erros é efectuado através de um código que contém alguma informação sobre os dados. O código mais simples é apenas um bit, chamado bit de paridade, que indica se o número de 1's da mensagem é par ou ímpar. Se, por erro de transmissão, houver uma troca de um bit na mensagem, o número de 1's é alterado e o bit de paridade deixa de corresponder à mensagem recebida, o que origina uma indicação de que a mensagem não foi correctamente recebida.

A camada seguinte, a de **rede**, encarrega-se de suportar os mecanismos de encaminhamento (routing) dos pacotes de dados. Nesta camada, a rede é vista como uma malha de nós e a comunicação entre quaisquer dois nós terá de percorrer um caminho através dessa malha. O método para determinar este caminho (método de encaminhamento) pode ser bastante complexo, conforme os requisitos de eficiência e as características da rede.

Na camada de **transporte**, surge pela primeira vez um serviço de comunicação de dados virtual, independente do meio físico de transmissão e da topologia da rede. Esta camada cria canais de comunicação entre utilizadores oferecendo serviços com diversas características. Uma das características mais importantes da comunicação a este nível tem a ver com a opção entre um serviço com ou sem **ligação**. Num serviço sem ligação, os pacotes são apenas enviados pelo emissor, sem que haja a garantia de recepção. Neste caso, a camada de transporte não acrescenta nenhuma funcionalidade à camada de rede.
Num serviço com ligação, a camada de transporte oferece um canal virtual de comunicação fiável. A recepção de pacotes é feita pela mesma ordem com que foram enviados e, no caso de se perderem pacotes na transmissão, a sua recuperação é assegurada. Neste tipo de serviço, o receptor deve sempre confirmar a recepção de uma mensagem, caso contrário, o emissor repete o envio. Todos os pacotes são numerados por forma a que o receptor possa ordenar os dados recebidos, eliminar pacotes duplicados, e determinar pacotes que faltam.

As camadas de **sessão** e **apresentação** raramente são utilizadas, justificando-se em casos muito específicos. A sua abordagem sai fora do âmbito deste texto.

A camada de **aplicação** é a que define as formas de comunicação a que estamos mais habituados (transferência de ficheiros, correio electrónico, terminal virtual, gestão de nomes, world wide web, etc.). Para cada forma de comunicação, define protocolos que são reconhecidos pelos intelocutores, embora não seja necessário todos os interlocutores usarem sempre o mesmo protocolo. Um exemplo é a utilização de protocolos seguros ou inseguros para a mesma aplicação.

4.3. O modelo Internet

A Internet utiliza a mesma filosofia do modelo OSI, embora possua menos camadas, como se mostra na figura 4.

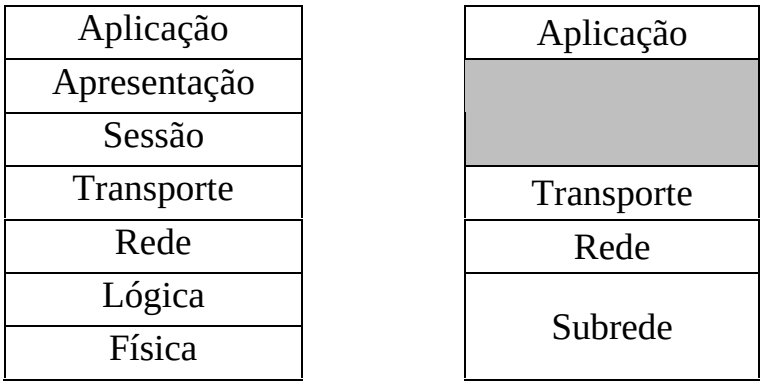


Figura 4 - Comparação entre os modelos OSI e Internet

Essencialmente, a camada de sessão e apresentação não existem na Internet, enquanto que a camada física e lógica estão agregadas num nesmo nível. A figura 5 mostra os protocolos mais usados em cada uma das quatro camadas do modelo Internet.

Aplicação	**SMTP**	**DNS**	**FTP**	**HTTP**	**TELNET**
Transporte	**TCP**			**UDP**	
Rede	**IP**	**ICMP**		**ARP**	**RARP**
Subrede	**Ethernet**	**Rede Telefónica**			**WiFi**

Figura 5 - Principais protocolos do modelo Internet

4.3.1. Camada de sub-rede

A norma **Ethernet** define as características de um tipo de rede local que é muito usado. Os suportes físicos deste tipo de rede são muito variáveis, utilizando cabos de fibra óptica, coaxiais, par entrançado, etc.. A rede Ethernet utiliza uma topologia tipo bus com um velocidade de transmissão entre os 10 e 100 Mbit/s. Actualmente, muitas redes Ethernet utilizam equipamentos que se chamam concentradores (hub) que agregam várias ligações a computadores. A topologia em bus continua a ser válida, já que o concentrador funciona conceptualmente como um bus.

Como na topologia em bus, todos os computadores partilham a mesma linha de comunicação, é necessário garantir que não há dois computadores a transmitirem dados simultaneamente. Caso contrário, dá-se uma situação a que se chama **colisão**. Para minimizar as hipóteses de colisão, cada computador executa os seguintes passos:

1. Escuta o bus para saber se está livre.
2. Se o bus estiver livre, lança a transmissão.
3. Se o bus não estiver livre, espera um pouco, e volta ao passo 1.
4. Escuta novamente o bus, para saber se houve colisão.

5. Se não houve colisão, termina o processo.
6. Se houve colisão, pára a transmissão e envia uma série de bits de reforço de colisão, para garantir que todos os outros computadores detectam a colisão.
7. Espera uma quantidade de tempo aleatório e volta a tentar a transmissão.

No caso de várias tentativas falhadas de transmissão, por sobrecarga da rede, o computador desiste e assinala um erro.

As redes telefónicas e de televisão por cabo são outras infraestruturas que podem servir de meio de transmissão na camada de sub-rede. Os computadores são ligados a estas redes através de aparelhos chamados **modems** (abreviatura de modulator/demodulator). Outra forma, cada vez mais utilizada, de transmissão dos dados é através de sinais de rádio (norma **WiFi**). Este tipo de ligação permite que os computadores não tenham de estar ligados por um cabo à rede, e é muito utilizado nos computadores portáteis.

4.3.2. Camada de rede

No modelo Internet, o protocolo base da camada de rede é o **IP** (Internet Protocol). Nesta camada, os dados são divididos por vários pacotes que são depois transmitidos pela rede. Cada pacote deve conter o endereço do remetente e o endereço do destinatário. O serviço oferecido pelo protocolo IP é sem ligação, o que, como vimos acima, não garante que os dados cheguem ao destino e, se chegarem, que venham pela mesma ordem. Relembremos que essa tarefa é da competência da camada de transporte.
A utilização do protocolo IP obriga a que cada equipamento que esteja ligado à rede possua um endereço, que é constituído por quatro bytes (32 bits). Os endereços IP são representados por quatro números decimais entre 0 e 255 separados por pontos. Exemplos:

172.16.1.135
10.0.1.201
192.1.1.255

Os endereços IP são constituídos por duas partes, o `netid`, que identifica a rede e o `hostid` que identifica a máquina. O netid constitui a parte mais à esquerda do endereço e o hostid é a parte mais à direita. No entanto, o número de bits associado a cada uma das partes não é sempre o mesmo, existindo três classes de endereços identificadas pelas letras A, B e C. A tabela seguinte mostra o número de bits usado para cada parte do endereço IP:

Classe	netid	hostid
A	8	24
B	16	16
C	24	8

Os endereços IP cujo `hostid` é 0 identificam uma rede como um todo e não um computador particular. Por outro lado, quando os bits do `hostid` são todos 1, o endereço resultante constitui um endereço de difusão (broadcast) para toda a rede. Por exemplo, o endereço 192.1.1.255 constitui um endereço de difusão da classe C.

A expansão da Internet na última década reduziu bastante a quantidade de endereços de IP disponíveis, estando já atribuídos a maioria dos endereços da classes A e B. Existe uma proposta de evolução do IP (chamada IPv6) em que os endereços passam a ter 128 bits, o que permite ligar muitos mais equipamentos à Internet.

4.3.3. Camada de transporte

O protocolo de transporte que garante a entrega de pacotes de dados pela ordem em que foram enviados chama-se **TCP** (Transfer Control Protocol). Na camada de transporte, o emissor coloca números de sequência nos pacotes enviados e o receptor verifica se os pacotes chegam pela ordem correcta. Caso falte algum pacote, o receptor envia um pedido de retransmissão desse pacote ao emissor, e só quando todos os pacotes são recebidos pela ordem correcta, o receptor acusa a recepção dos dados à camada superior. O TCP, juntamente com o IP constituem os mais importantes protocolos usados na Internet, pelo que muitas vezes se fala em TCP/IP como o protocolo de transmissão de dados pela Internet (na realidade são dois protocolos).

Existe ainda um outro protocolo muito usado na camada de transporte, o **UDP** (User Datagram Protocol), que não tem ligação como o TCP. Em vez de receber os pacotes pela ordem correcta, a camada de aplicação recebe os pacotes assim que eles chegam. O UDP tem vantagens quando a velocidade de transmissão é mais importante do que a fiabilidade. A difusão de áudio e vídeo em tempo real é a aplicação que mais usa este protocolo, porque a falta de um pacote ou outro não impede a percepção da mensagem por parte do receptor.

4.3.4. Camada de aplicação

No nível de aplicação proliferam os protocolos, a que correspondem às muitas aplicações que fazem uso da Internet. Para cada tipo de comunicação (e-mail, ficheiros, web, terminal virtual, etc.) existem um ou mais protocolos associados. A maior parte dos protocolos da camada de aplicação funciona num modelo cliente-servidor, em que existem determinadas máquinas (servidores), que aceitam pedidos de comunicação por parte de clientes, usando os protocolos adequados.
Referimos apenas os protocolos mais importantes:

DNS – Domain Name Service, envolve um servidor que aceita pedidos de identificação de nomes e que envia o respectivo endereço IP. Por exemplo, www.site.com é um nome que identifica uma máquina na rede. O servidor DNS indica qual é o endereço IP que corresponde a essa máquina para que a comunicação possa continuar. Os nomes são formas muito mais fáceis para nós, simples humanos, de identificar um interlocutor na rede.

HTTP – HyperText Transfer Protocol, protocolo usado na transmissão de páginas WWW (world-wide web). Envolve um servidor que mantém as páginas e aceita pedidos de outras máquinas na rede para visualizar essas páginas. As páginas são enviadas e o cliente pode visualizá-las através de um programa chamado browser.

SMTP – Simple Mail Transfer Protocol, usado na transmissão de mensagens de correio electrónico. O servidor SMTP aceita pedidos de envio de mensagens de correio electrónico por parte de um cliente devidamente autenticado (por nome de utilizador e password).

FTP – File Transfer Protocol, usado na transferência de ficheiros entre máquinas. Novamente um servidor FTP aceita pedidos de transferência de ficheiros, por parte de um cliente autenticado por nome de utilizador e password. Existem servidores de FTP que disponibilizam publicamente alguns ficheiros. Neste caso, a autenticação é feita normalmente por um nome de utilizador 'anonymous', sendo a password o endereço de email do cliente.

TELNET – Este protocolo é usado no estabelecimento de sessões remotas, em que o cliente acede ao servidor através de um terminal virtual.

Os protocolos FTP e TELNET, porque envolvem a transmissão de um nome de utilizador e password não são considerados seguros, já que alguém com acesso à rede pode facilmente obter esses dados. A forma de assegurar a confidencialidade é cifrar a comunicação, de forma a impedir que alguém que aceda aos dados consiga decifrá-los. Para esse fim são cada vez mais usados os protocolos SSH (Secure SHell) para estabeler terminais virtuais seguros e SFTP (Secure File Transfer Protocol) para transferência segura de ficheiros.