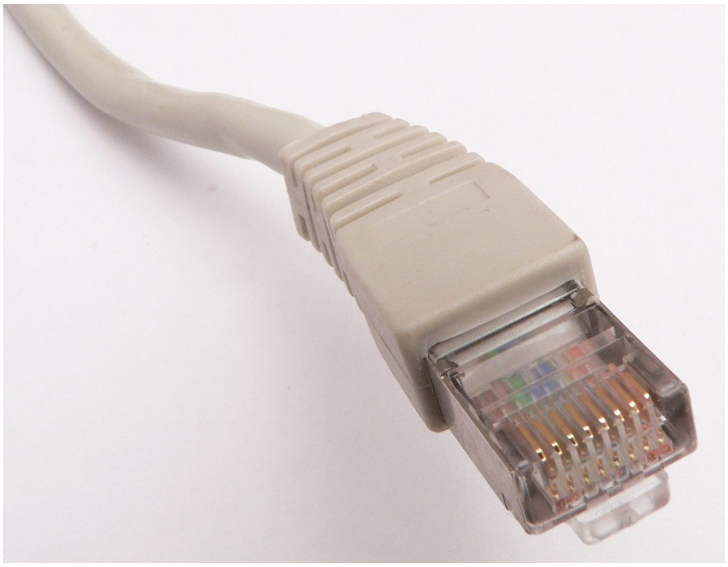
Data transmission

IGCSE Computer Science

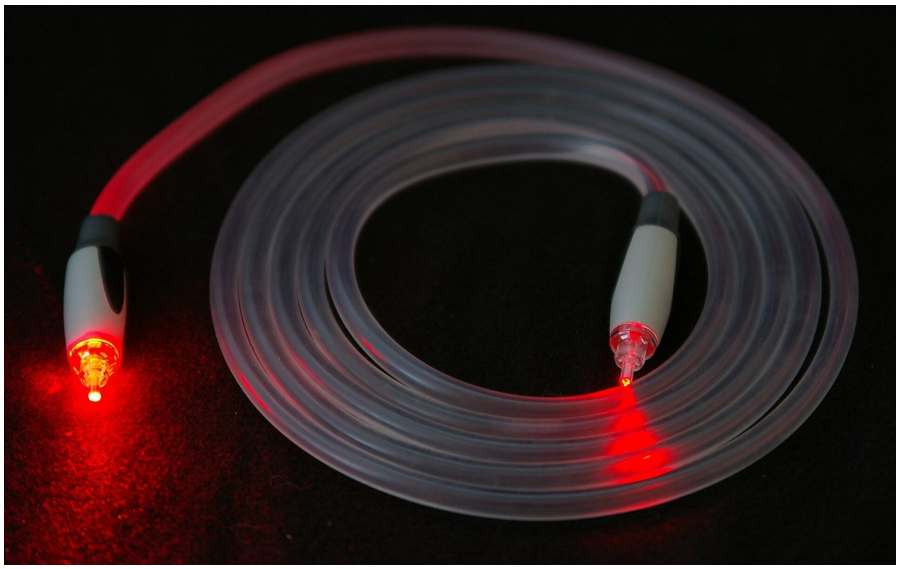
What is data transmission?

Data transmission 数据传输 means moving data from one place to another —for example from your computer to a website, or from a phone to a printer.

Data often travels along a cable. Two common ones are shown below.



An Ethernet cable with an RJ45 plug, often used to carry data on a wired network



A fibre-optic cable carries data as pulses of light along thin glass fibres

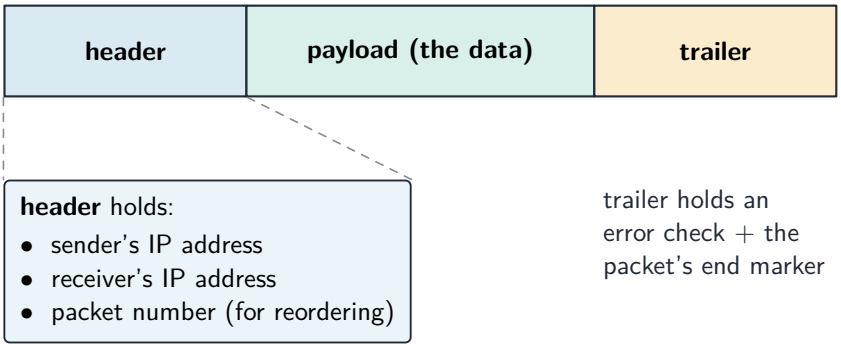
Before it is sent, the data is broken down into small, equal-sized pieces called **packets** 数据包. Each packet travels separately and they are put back together at the other end.

Using packets has benefits:

- if one packet is lost or damaged, only that packet is sent again, not the whole file;
- packets can take different routes, so a busy or broken route can be avoided;
- many users can share the same connection at the same time.

The structure of a packet

Every packet has three parts.



A packet has a header, a payload (the data) and a trailer; the header carries the addresses and packet number

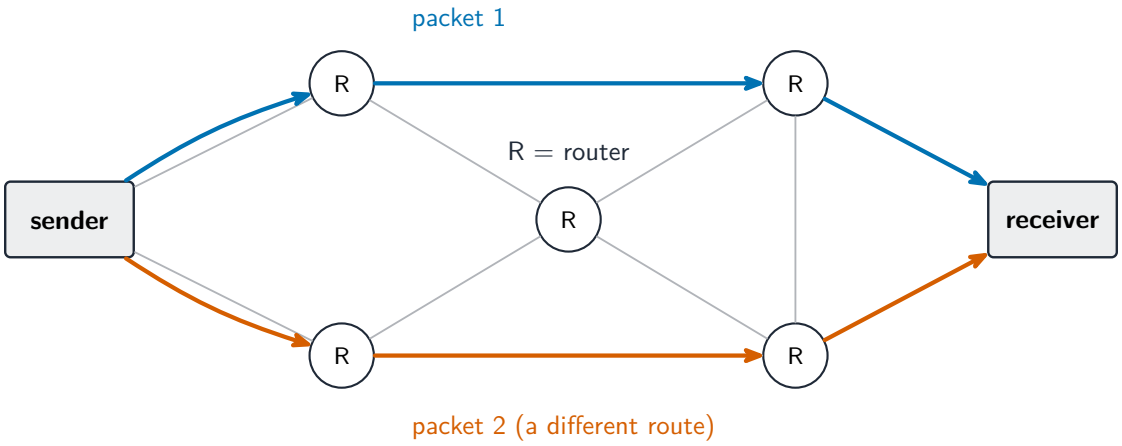
Part	What it holds
packet header 包头	control information (see below)
payload 有效载荷	the actual data being carried
trailer 尾部	shows where the packet ends, plus an error check

The **packet header** holds:

- the sender's IP address 地址 (where the packet came from),
- the receiver's IP address (where it is going),
- the **packet number** (so the packets can be put back in the right order).

Packet switching

Packet switching 数据包交换 is the way packets are sent across a network.



Routers forward packets, and different packets can take different routes to the same receiver

- Special devices called **routers** 路由器 read each packet's header and choose the best route for it.
- Each packet may take its **own path**, so packets can travel different ways.
- Packets may **arrive out of order**.
- When the last packet has arrived, the packets are **reordered** using their packet numbers.

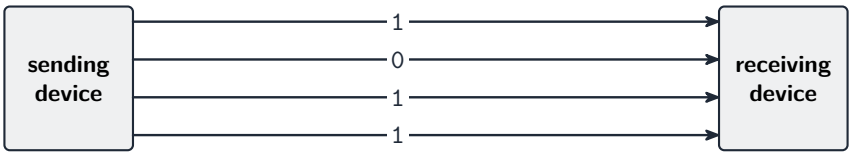
Methods of data transmission

Serial and parallel

	Serial 串行	**Parallel 并行**
How	one bit at a time, down one wire	several bits at once, down several wires
Distance	good over long distances	good over short distances only
Cost	cheaper (fewer wires)	more expensive (more wires)
Errors	bits stay in order	bits can arrive at slightly different times over long wires

Serial sends one bit after another along a single wire. **Parallel** sends several bits at the same time along several wires, so it is faster over short distances.

serial: one bit at a time, one wire

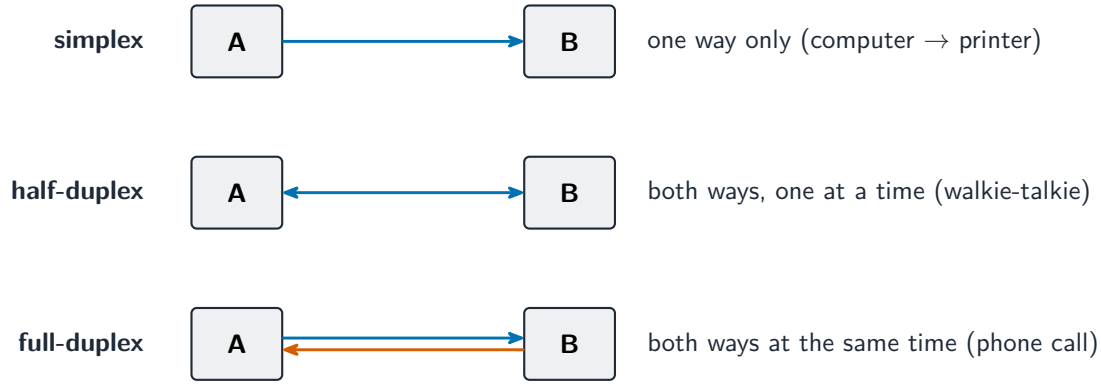


parallel: several bits at once, several wires

Serial sends one bit at a time down one wire; parallel sends several bits at once down several wires

Simplex, half-duplex and full-duplex

These describe the *direction* data can travel.



Simplex is one way only; half-duplex is both ways but one at a time; full-duplex is both ways at once

Method	Direction
simplex 单工	one direction only (e.g. computer → printer)
half-duplex 半双工	both directions, but only one at a time (e.g. walkie-talkie)
full-duplex 全双工	both directions at the same time (e.g. phone call)

Choosing a method

The choice depends on the **distance**, the **speed** needed, the **cost**, and whether data must travel both ways at once.

USB

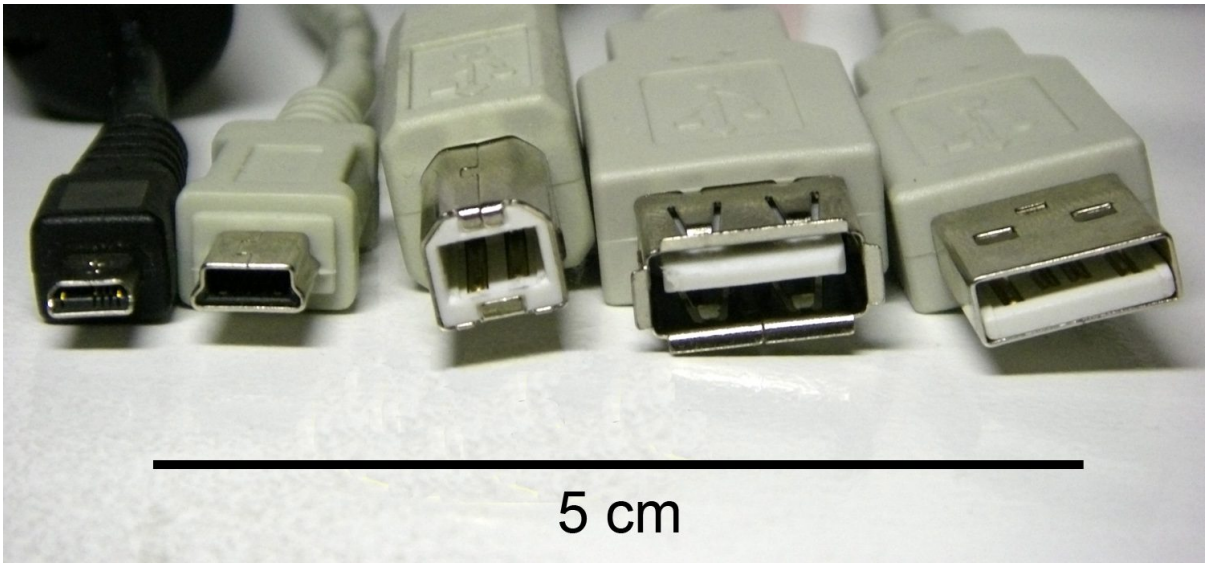
The **universal serial bus** 通用串行总线 (USB) is a common **interface** 接口 (a connection point) for joining devices to a computer.

Benefits:

- the same connector is used by many devices (one standard);
- the device is detected automatically and the correct driver is loaded;
- it carries power, so it can charge or run a device;
- the connector only fits one way, so it is hard to plug in wrongly.

Drawbacks:

- the cable can only be a limited length (a few metres);
- older USB versions are slower than newer ones;
- transfer speeds can be lower than some other connection types.



A few of the USB connector shapes. All use the same USB standard, so devices are detected automatically.

Image: Techtonic, Public domain (commons.wikimedia.org)

Why we check for errors

While data travels, **interference** 干扰 can change it. Three things can go wrong:

- **data loss** —some bits do not arrive;
- **data gain** —extra bits are added;
- **data change** —some bits are flipped.

So the receiver checks whether the data arrived correctly.

Error detection methods

Parity check

A **parity check** 奇偶校验 adds one extra bit, the **parity bit** 奇偶校验位, to each byte. There are two types:

- **even parity** —the total number of 1s in the byte (including the parity bit) must be **even**;
- **odd parity** —the total number of 1s must be **odd**.

Example (even parity): the data `1011001` has four 1s, which is already even, so the parity bit is `0`:

```
parity bit + data
    0        1011001
```

To find an error: the receiver counts the 1s. If even parity was agreed but a byte arrives with an **odd** number of 1s, an error has happened during transmission.

A parity check cannot find every error. If two bits flip, the count may still look correct.

Checksum

A **checksum** 校验和 is a value worked out from all the data before sending. The receiver works out the checksum again from the data it received. If the two values do not match, an error has occurred and the data is resent.

Echo check

In an **echo check** 回送校验, the receiver sends the data back to the sender. The sender compares it with the original. If they differ, an error happened. (This needs the data to be sent twice, so it is slow.)

Automatic Repeat reQuest (ARQ)

Automatic Repeat reQuest 自动重传请求 (ARQ) uses messages called **acknowledgements** 确认.

- The receiver sends a **positive acknowledgement** if a packet arrived correctly, or a **negative acknowledgement** if it did not.
- The sender starts a **timeout** 超时 timer. If no positive acknowledgement comes back in time, the sender sends the packet again.

Check digit

A **check digit** 校验码 is an extra digit placed at the end of a number, worked out from the other digits. It is used to find mistakes when a number is **typed in**.

When the number is entered, the check digit is calculated again and compared. It can catch:

- an **incorrect digit** entered,
- a **transposition error** 换位错误 (two digits swapped, e.g. 21 typed as 12),
- a digit **left out or an extra digit** added,
- a **phonetic error** (a digit that sounds similar, e.g. 13 and 30).

Check digits are used in **barcodes** 条形码 and ISBNs (book numbers).

Encryption

Why we need encryption

Encryption 加密 scrambles data so that it cannot be understood if it is **intercepted** 拦截 (caught) by the wrong person. The readable data is called **plaintext** 明文; after encryption it becomes **ciphertext** 密文.

Encryption does **not** stop data from being intercepted. It only stops the data from being understood.

Symmetric encryption

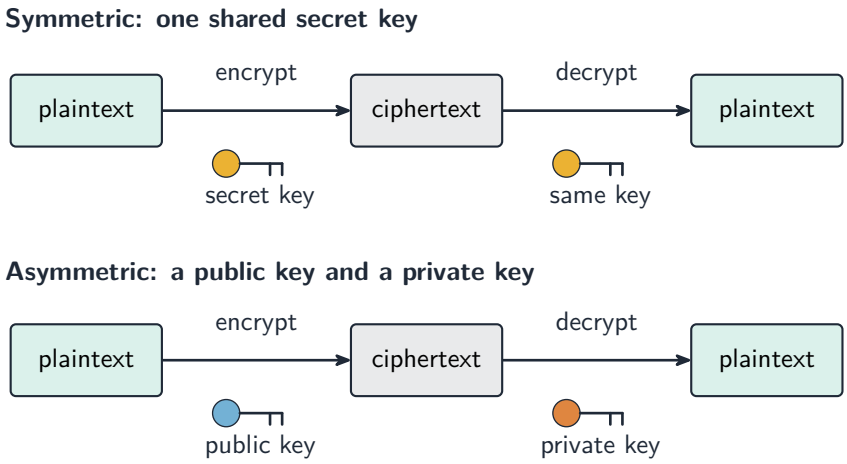
Symmetric encryption 对称加密 uses a single **key** 密钥 to both encrypt and decrypt the data. The sender and receiver must share this same secret key. The risk is that the key itself could be stolen while being shared.

Asymmetric encryption

Asymmetric encryption 非对称加密 uses two different keys:

- a **public key** 公钥 that anyone can have, used to encrypt;
- a **private key** 私钥 that is kept secret, used to decrypt.

Data encrypted with the public key can only be decrypted with the matching private key, so the key never has to be shared. This is safer than sharing one secret key.



Symmetric uses one shared key; asymmetric uses a public key to encrypt and a private key to decrypt