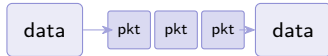


Data Transmission

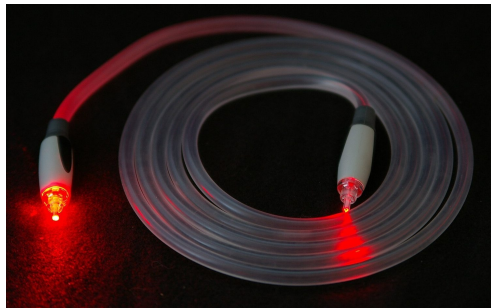
IGCSE Computer Science ■ Topic 2

IGCSE Computer Science



What we will cover

- 1 Packets & packet switching
- 2 Serial, parallel & duplex
- 3 USB
- 4 Error detection
- 5 Encryption
- 6 Recap



Data often travels along a cable

Ethernet cable

Copper twisted pairs ending in an RJ45 plug – the usual wired cable inside a room or building.

Fibre-optic cable

Thin glass fibres carrying data as *pulses of light*: much faster over long distances and unaffected by electrical interference.

Either way

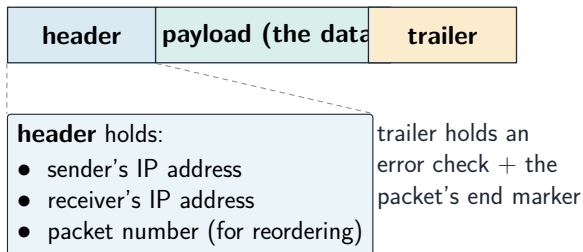
Data transmission 数据传输 just means moving data from one place to another – computer to website, phone to printer.

Two cables, two jobs: copper for the last few metres, fibre for the distance between buildings and cities.

1

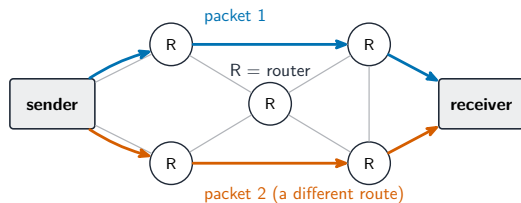
Packets

Packets



- the **packet header** 包头: the sender's IP address, the receiver's IP address and the **packet number** (so the parts can be reordered);
- the **payload** 有效载荷: the data itself;
- the **trailer** 尾部: shows where the packet ends, plus an error check.

Packet switching



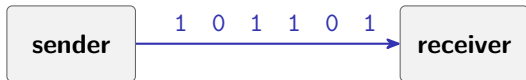
- **routers** 路由器 read each header and pick the best route
- each packet may take its **own path**
- so packets may **arrive out of order**
- the **packet numbers** put them back in order

Different routes means a busy or broken link can be avoided, and many users share one connection.

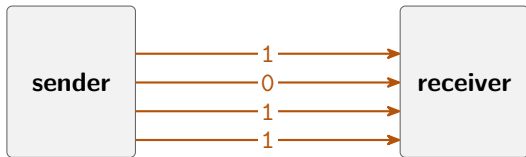
2

Transmission methods

Serial vs parallel



serial: one wire, one bit at a time



parallel: several wires at once

	serial 串行	parallel 并行
bits	one at a time	several at once
wires	one	many
distance	long	short only
cost	cheaper	dearer

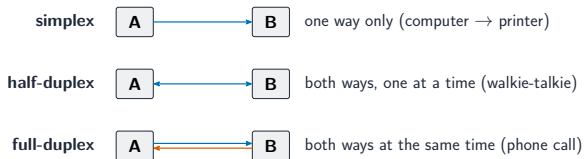
Over long parallel wires the bits arrive at slightly different times – which is why long links go **serial** (USB, the S is serial).

Choosing a method

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

if the need is...	choose
a long distance	serial : one wire, no skew
high speed over a short distance	parallel : many bits at once
low cost	serial : fewer wires
data both ways at the same time	full-duplex

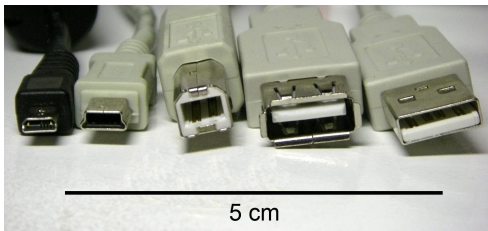
Direction: the three duplexes



- **simplex** 单工: one way only (PC → printer)
- **half-duplex** 半双工: both ways, one at a time (walkie-talkie)
- **full-duplex** 全双工: both ways at once (phone call)

Choose by **distance**, **speed**, **cost**, and whether data must flow both ways **at once**.

USB



The **universal serial bus** 通用串行总线 is one standard **interface** 接口:

- one connector for many devices
- auto-detected; driver loaded
- carries power; fits only one way

But: limited cable length, and older versions are slower.

USB: what you give up

Cable length

Only a few metres – beyond that the signal is unreliable, so USB cannot wire a building.

Older versions

An old USB port stays slow even with a new device: both ends must support the newer standard.

Transfer speed

Transfer speeds can be lower than some other connection types – Ethernet or Thunderbolt beat USB for large transfers.

USB wins on convenience (one connector, auto-detected, carries power) and loses on **distance and top speed**.

3

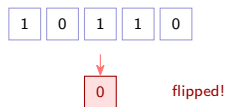
Error detection

Why check for errors?

Interference 干扰 in transit can cause:

- data loss – bits do not arrive
- data gain – extra bits appear
- data change – bits are flipped

So the receiver checks the data arrived correctly, using one of several methods.



Parity check

data: 1 0 1 1 0 0 1

count the 1s: four (even)

parity bit = 0 (keep it even)

sent: 1 0 1 1 0 0 1 **0**

A parity bit is added to make the number of 1s even (or odd)

A parity block finds *which* bit is wrong

The problem

A single **parity bit** 奇偶校验位 tells you *that* an error happened, never *where*.

Parity block

Send several bytes together: each byte carries a row parity bit, and one extra **parity byte** at the end carries the column parities.

Finding it

The row that fails and the column that fails cross at the flipped bit – change it back and the data is repaired.

Row parity plus a **parity byte** turns error *detection* into error *correction* for a single flipped bit.

ARQ: decide the next message

Automatic Repeat reQuest 自动重传请求 (ARQ) uses **acknowledgements** 确认 and a **timeout** 超时.

A sender sends packet 4 and starts its timer. Consider three separate cases:

Case	What happens at the receiver?
A	Packet 4 arrives; its error check passes.
B	Packet 4 arrives; its error check detects an error.
C	Packet 4 never arrives.

For each case, choose the receiver's response and the sender's next action. When does the sender need its timer?

ARQ: trace the three outcomes

Case	Receiver	Sender
A	Sends a positive acknowledgement: the check passed.	Receives it before timeout; stops this timer and can send the next packet.
B	Sends a negative acknowledgement: an error was detected.	Receives it; sends packet 4 again and restarts the timer.
C	Has no packet to check; sends no acknowledgement.	Timer expires without a positive acknowledgement; resends packet 4 and restarts the timer.

A negative reply can trigger a repeat *before* timeout. The timer handles silence, including a lost packet or a lost reply.

ARQ: the packet arrived, but the reply did not

Trace a fourth case. Events run from top to bottom.

- 1 Sender sends packet 4 and starts the timer.
- 2 Receiver gets packet 4; its error check passes.
- 3 Receiver sends a positive acknowledgement, but the reply is lost.
- 4 The sender's timer expires.

What does the sender do now? Does timeout prove that the packet was damaged?

Explain why the receiver's knowledge and the sender's knowledge differ.

Timeout means no confirmation

The sender resends **packet 4** and restarts its timer because it has received no positive acknowledgement.

The receiver already received the packet successfully. The sender cannot know that: the acknowledgement was lost on its return journey.

Timeout means **no positive confirmation arrived in time**; it does not identify whether the packet or the reply was lost or delayed.

ARQ requests another transmission; it does not itself repair the damaged bits. It uses an error check to decide whether to accept a received packet.

Worked example – the parity block

Even parity. One bit flipped in transit – find it.

	b1	b2	b3	b4	b5		par	
by1	1	0	1	1	0		1	
by2	0	1	1	0	1		1	
by3	1	1	1	1	0		1	row odd
by4	0	0	1	1	0		0	
par	0	0	1	1	1			col b3 odd

One parity bit only shows **that** an error happened. A parity **block** pinpoints it: the bad row $\times$ the bad column = **byte 3, bit b3**.

Four more checks

checksum 校验和

a value from the data; re-computed and compared

echo check 回送校验

receiver sends the data back to compare

ARQ 自动重传

ack/timeout; re-send if none arrives

check digit 校验码

an extra digit on a typed number (barcodes, ISBN)

A **check digit** helps detect entry errors, including some swapped (**transposition**), missing or extra digits. It does not catch every error.

What a check digit catches

An incorrect digit	One digit typed wrongly – 7 for 1.
A transposition error	Two digits swapped: 21 typed as 12.
A missing or extra digit	A digit left out, or one typed twice.
A phonetic error	A digit that <i>sounds</i> similar – 13 heard and typed as 30.

The check digit is recalculated from the other digits when the number is entered and compared – which is why **barcodes** and ISBNs carry one.

4

Encryption

Encryption

Symmetric: one shared secret key



Asymmetric: a public key and a private key



- **symmetric** 对称: one **shared** key encrypts and decrypts
- **asymmetric** 非对称: a **public key** 公钥 encrypts, a **private key** 私钥 decrypts

Encryption does **not** stop interception – it stops the data being **understood**. Asymmetric never shares a secret key.

Plaintext, ciphertext, and what encryption really promises

Plaintext

Plaintext 明文 is the readable data before encryption – what a thief wants.

Ciphertext

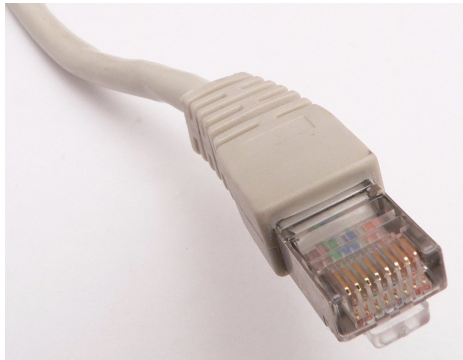
After encryption it becomes **ciphertext** 密文: still readable as characters, but meaningless without the key.

Intercepted

Encryption does **not** stop data being **intercepted** 拦截; it stops the intercepted data being *understood*.

State the promise precisely in an exam: confidentiality, not delivery. The packet can still be caught – it just cannot be read.

An Ethernet cable with an RJ45 plug



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

Exam tips

- Data is sent in **packets** (a header with the addresses and packet number, a payload, and a trailer). Packets can take different routes and arrive out of order, then are reordered by their packet numbers.
- Match the transmission types: **serial** (one bit at a time, good over long distances) vs **parallel** (several bits at once, short distances); **simplex** / **half-duplex** / **full-duplex** describe the direction.
- A single **parity bit** only shows *that* an error happened; a **parity block** locates the exact wrong bit – the row and the column that both fail.
- Learn the error-detection methods and what each does: parity check, checksum, echo check, ARQ, and the check digit (for typed-in numbers).
- **Symmetric** encryption uses one shared key; **asymmetric** uses a public key to encrypt and a private key to decrypt, so no secret key is ever shared.

5

Recap

Topic 2 – key takeaways

1 Packets

header (addresses, number) +
payload + trailer

2 Serial/parallel

serial one bit, long distance; par-
allel many, short

3 Errors

parity block locates the bit;
checksum, echo, ARQ

4 Encryption

symmetric one key; asymmetric
public + private

Check yourself

- 1 What puts out-of-order packets back in order?
- 2 Walkie-talkie: which duplex?
- 3 What does a parity block locate that one bit cannot?
- 4 Which key decrypts in asymmetric encryption?



Answers 1. the packet numbers 2. half-duplex 3. the exact wrong bit 4. the private key