Sending data across a network

Data needs to be **sent across a network** all the time:

- an e-mail or SMS sent from one device to another
- a file being opened on a student machine
- music data sent to bluetooth headphones
- webpages delivered to browsers
- security updates sent to a device

You need to know the basics of **how** this is done

1101011010001010111110010100010100011110101010
0111110000111001111100101011101001011100110000
1111001011100101011010100010011100001110001011
0011010100100010001110011101010010111011110010
1011011001011000111010111100011000010011101111
0101011000011001000011101 1

```
1101011010001010111110010100010100011110101010
0111110000111001111001010111010010111001100000
1111001011100101011010100010011100001110001011
0011010100100010001110011110101001011101111010
1011011001011000111010111100011000010011101111
0101011000011001000011011
```

Data to be sent across a network

There are **256 bits** of data in this example

Many sets of data to be sent would be much bigger than this

The network could be a LAN, WAN (including the internet) or PAN - data is all transmitted the same way

1101011010001010111110010101000101

0001111010101001111110000111001111

1001010111010010111001100000111111

0010111001010110101001001110000

1110001011001111

0111010100101111

0101100011101011

1011110101011011

The data is broken down into smaller **packets** before it is sent

This makes the sending **more efficient** and **increases security** in case a packet gets intercepted or delivered to the wrong address

In this case each packet is 32 bits

(real packets are much bigger than this)

0110111000111010110100010101111100101000101001

0110111001000011110101010011111000011100111**001**

0110111001110010101110100101110011000001111001

01101110100001

01101110101111

01101110110011

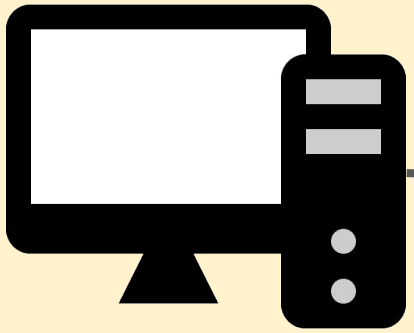
01101110111010

Headers and **trailers** are added to each packet. These do a number of things

- identify the **type of data**
- identify the sender and **address** to be delivered to
- include **error checking** bits in case some data is lost
- **number the packets** so they can be put back together in the right order
- include the total number of packets in the message

0110111100010111101010110000110010000111011001


0110111**0001**1101011010001010111110010100010100 1
0110111**0010**0001111010101001111100001110011100 1
0110111**0011**1001010111010010111001100001111001
0110111**0100**001
0110111**0101**111
What specifically are these four bits doing?
Real headers and trailers are much longer than this
0110111**0110**0111010100101110111100101011011000 1
0110111**0111**0101100011101011110001100010011001
0110111**1000**1011110101011000011001000011101100 1

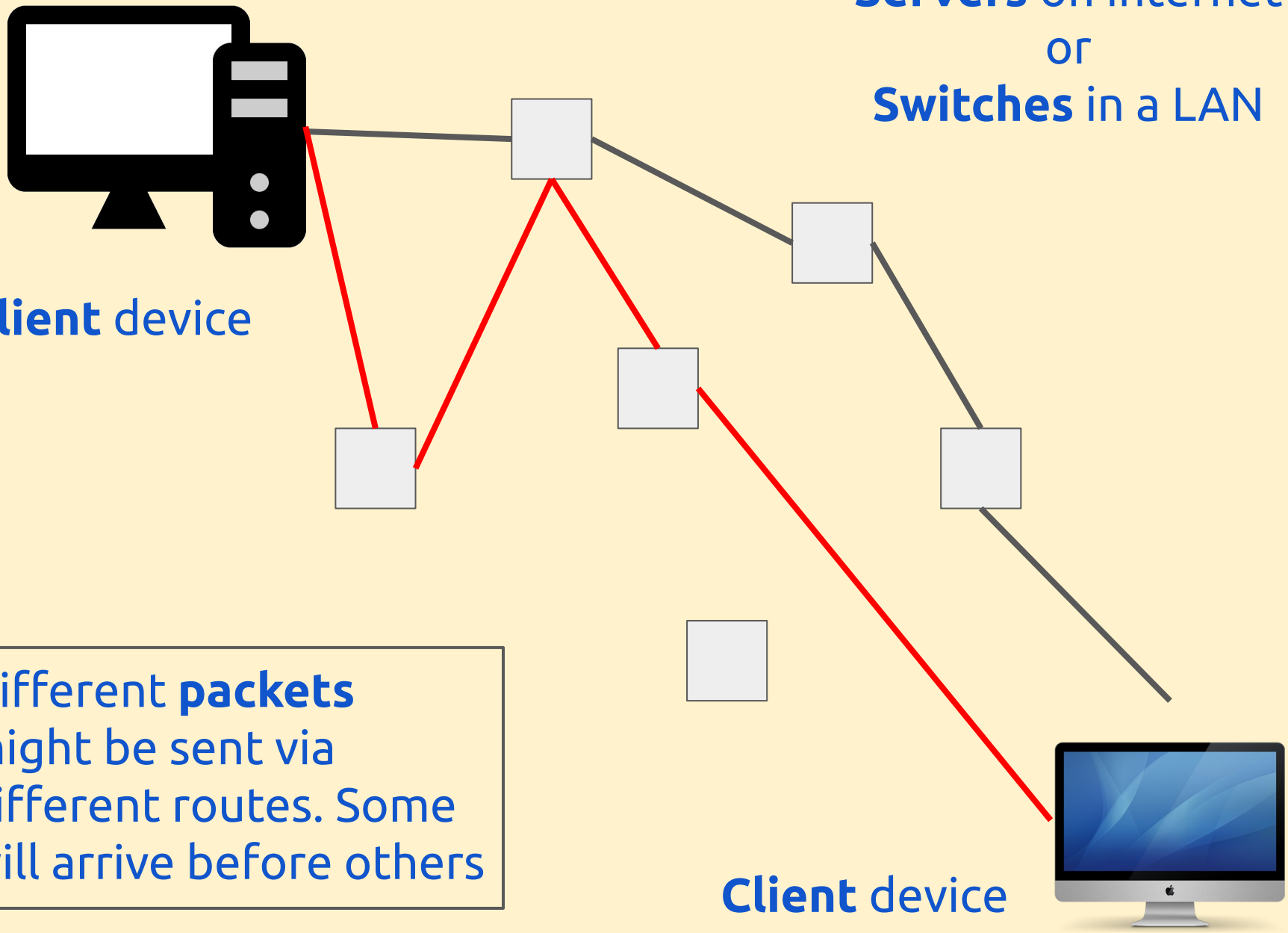


Servers on internet
or
Switches in a LAN

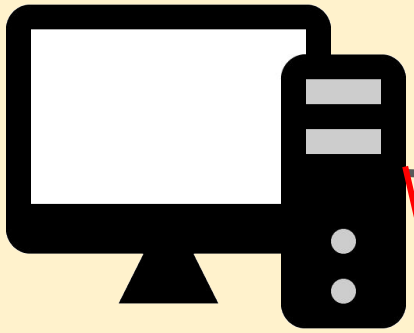
Client device

Data **packets** are sent across the network, routed by whatever route is most efficient right now

Client device


Servers on internet
or
Switches in a LAN
Client device
Different **packets** might be sent via different routes. Some will arrive before others
Client device

Servers on internet
or
Switches in a LAN

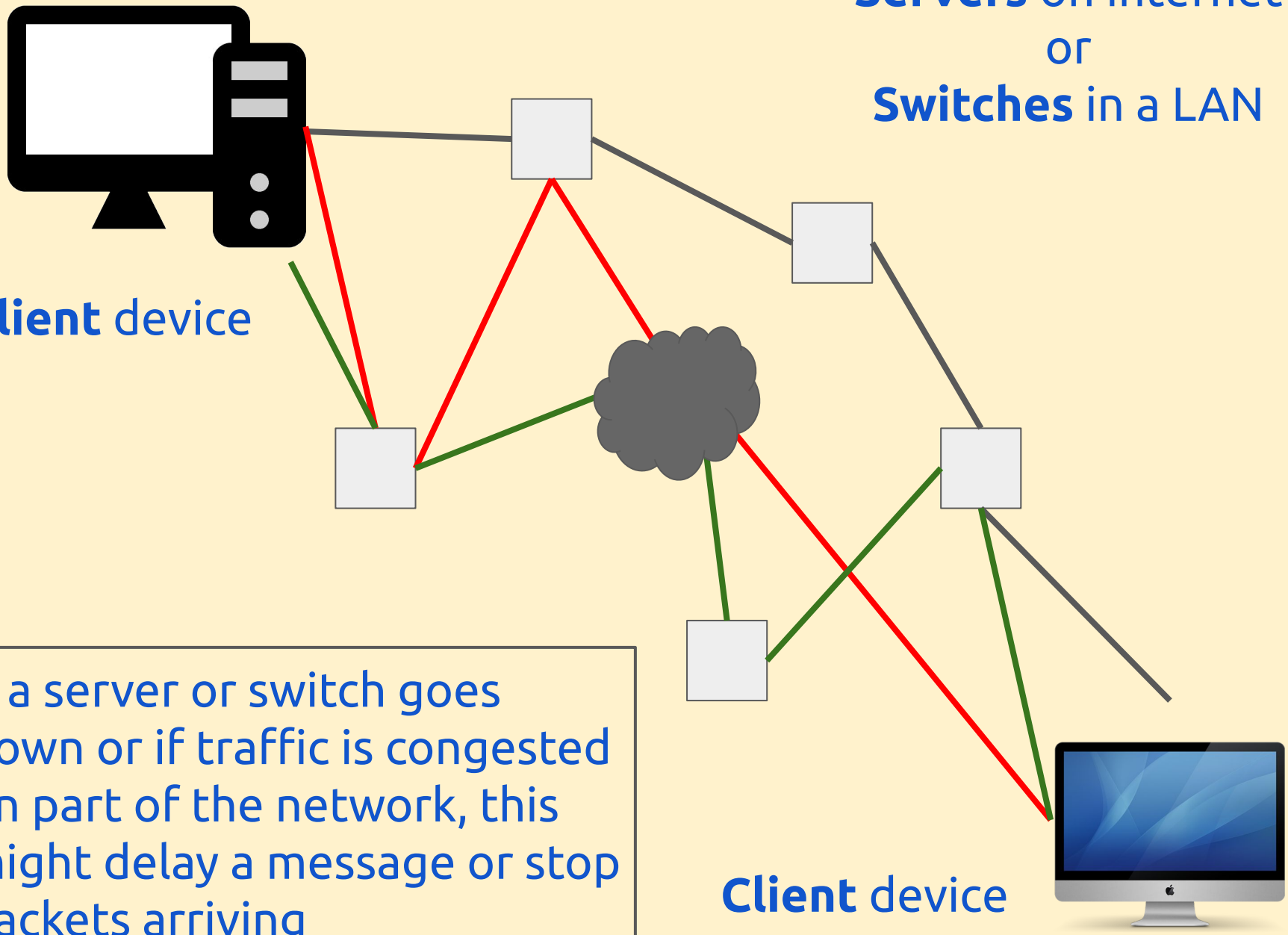


Client device

Client device

When packets arrive the data is put back together in the right order. Any **missing packets** can be identified and a message sent to re-send the data

Servers on internet
or
Switches in a LAN

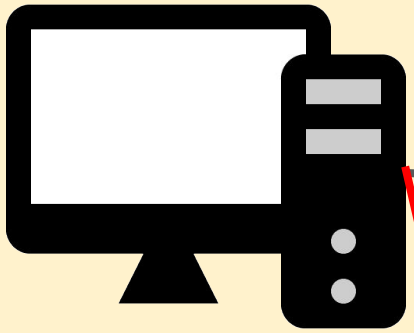


Client device

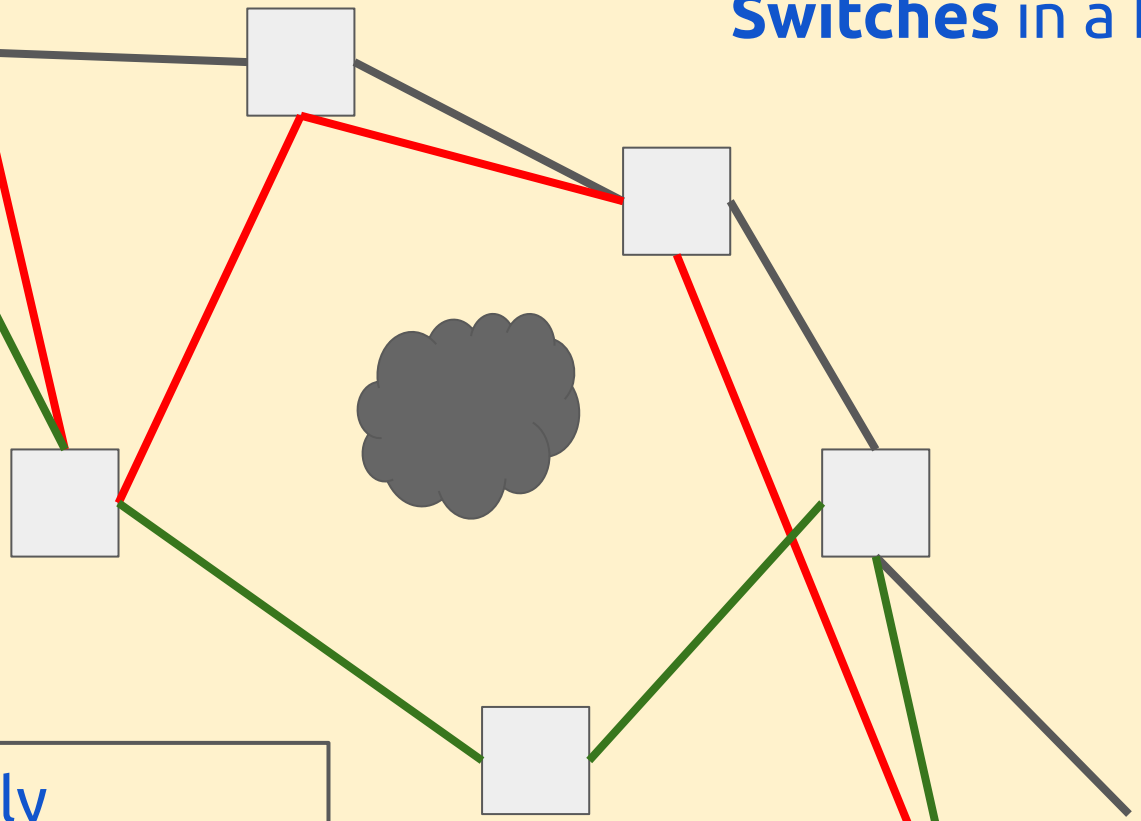
Client device

If a server or switch goes down or if traffic is congested on part of the network, this might delay a message or stop packets arriving

Servers on internet
or
Switches in a LAN



Client device



Client device

Packets are simply re-routed to avoid the problem server - if possible…

(this may not be possible on a LAN or in some cases where a single connection is used)

Sending data across a network

You need to know the basics:

- that data is broken down into **packets**
- that **headers** and **trailers** are added
  - that these include **addresses**
  - and identify the **order** of packets
  - and include **error checking bits** and that error checking can take place
- that data is **routed efficiently** and can be re-sent if necessary

Sending data across a network

Come back to this slide once you know about protocols

- **TCP** or **UDP** defines **how** data is split into packets. They also **reassemble packets** in the right order once data is received and do **error checking**
  - **TCP** can ask for packets to be re-sent and fully checks for errors
  - **UDP** can't - it's more basic and designed to be quicker and does less error checking
- **IP** defines **how to address packets** and **route them**. It relays data across network boundaries (i.e. from the client to the network and vice versa)