Networking
CM30078/CM50123

Russell Bradford

2023

Introduction

These days we are all networked

Whether on a PC, tablet, phone or other device we spend our time surfing the web, reading emails, streaming media

We find it hard to do anything when were are not connected

We feel cut off when we can't communicate

This Unit is about how the technology that allows this to happen works, in particular, the Internet

Unit Outline

Structure of this unit: starting with 3 hours lectures per week

- Tuesday 13:15
- Wednesday 12:15
- Friday 13:15

The aim is to cover the necessary material early in the semester which will leave the last few weeks free for revision and problems classes

Unit Outline

Assessment

For the undergraduate CM30078:

- End of unit exam 100%

Unit Outline

Assessment

For CM50123

- Coursework 25%
- End of unit exam 75%

Coursework timelines (subject to change):

1. set Thu 26 Oct
   due Wed 15 Nov
2. set Thu 16 Nov
   due Fri 8 Dec

Feedback on coursework will be provided via Moodle

Unit Outline

Week 6 (starting 6th Nov) will be a “consolidation week”

No lectures for the whole of Computer Science (CM Units)

Presumably other Departments will carry on as usual

Unit Outline

Aims To understand the Internet, and associated background and theory, to a level sufficient for a competent domain manager.

Unit Outline

Learning Outcomes Students will be able to:

- Explain the acronyms and concepts of the Internet and how they relate;
- State and apply the steps required to connect a domain to the Internet and explain the issues involved to both technical and nontechnical audiences;
- Discuss the ethical issues involved with the internet, and have an “intelligent layman’s” grasp of the legal issues and uncertainties.
- Be aware of the fundamental security issues;
- Be able to advise on the configuration issues surrounding a firewall.

Unit Outline

Syllabus:

- The ISO 7-layer model. The Internet: its history and evolution - Predictions for the future.
- The TCP/IP stack: IP, ICMP, TCP, UDP, DNS, XDR, NFS and SMTP. Berkeley. Introduction to packet layout: source routing etc.
- Various link levels: SLIP, 802.5 and Ethernet, satellites, the “fat pipe”, ATM. versus carrying. Security and firewalls. Performance issues: bandwidth, MSS and RTT; caching at various layers.

Unit Outline

- Who ’owns’ the Internet and who ’manages’ it: RFCs, service Providers, domain managers, IANA, Jisc/UKERNA, MANs, commercial British activities. Routing protocols and default routers. HTML and Electronic publishing.
- Legal and ethical issues: slander/libel, copyright, pornography, Publishing

Unit Outline

We won't be covering the material in the above order, though, but in a more coherent fashion instead!

Also note that this is a Final Year/Masters Unit and so is a lot more stretching than previous years

It contains a lot of material as networking is a big subject

It is how the Internet works, **not** how to write networking programs or how to write Web pages

Unit Outline

Resources

Networking is now a mature subject (though still under development and change!) so there are many books available

I recommend

- “TCP/IP Illustrated Volume 1” W R Stevens, Addison-Wesley
- “Computer Networks, 5th Ed” A Tanenbaum, Pearson (4th Ed OK)
- “The Art of Computer Networking” R Bradford, Pearson (Polish Edition: “Podstawy Sieci Komputerowych”, WKŁ)

Stevens is available as an e-book in the library

Unit Outline

Resources

You don’t need me to tell you that there is a large amount of material out there on the Web?

Wikipedia is fairly accurate in this area: but, as usual with Wikipedia, you should check with other sources

Unit Outline

Resources

There is a Unit Moodle page, but as Moodle is so horrible I tend to use my own Web pages:

`https://people.bath.ac.uk/masrjb/CourseNotes/cm30078.html`

Unit Outline

Content

We will revisit and expand on what you (may have) seen in CM10195 or other units

But is much greater breadth and some detail

Unit Outline

Networking is a large subject with a lot of complicated detail

And there are very many acronyms

You'll need to remember the main acronyms, but a lot are less important

We shall cover much material in lectures

It is very techy material: if you are not a techy person you should think very carefully about taking this Unit!

Unit Outline

But it is the big picture that is important for this Unit

For example, there are many packet headers that contain lots of flags and fields

You should have a general idea of what the important fields are and what their purpose is, but precisely *where* they appear in the header is generally less important

Standard Introductory Slides

Remember:

You are expected to do some work outside of lectures

Lectures are the *start* of the learning process, not the end!

These slides are reminders to me on what to say in lectures

They are often abbreviated in style, and so are not the whole story and would not be suitable to be quoted verbatim in an exam

Standard Introductory Slides

Don’t try to copy everything down from the slides in lectures—the slides will be available after each lecture

Instead, make a note of what is important and use that later—in conjunction with the slides—to guide your further reading and study

Standard Introductory Slides

Do not rely purely on my notes for your revision

People who do this live to regret it

Like every Unit, you are expected to read around the subject for yourself

You need to take your own notes, read, and *participate*

You don't expect to get fit simply by paying to joining a gym. . .

"If you have college courses in CS, buy the books and spend day and night the few days before class going through the books and taking notes and answering questions and programming examples before the first class even starts. If you really want to do this in your life, that's what you should do, not just wait for the education to be handed you. Those who finish at the top will always be in high demand. You can learn outside of school too but you have to put a lot of time into it. It doesn't come easily. Small steps, each improving on the other, is what to expect, not instant understanding and expertise."

Steve Wozniak, co-founder of Apple

Standard Introductory Slides

Computer Science is not a spectator sport

Anon

Networks

Networks form a central role in the way computers are used today: these days it is very hard to do anything that is not networked

As commerce and big money have taken over the Internet the nature of networking has changed from a way of linking together some CS departments to a multi-billion (trillion?) pound enterprise

Thus a good knowledge of what networks are and how they work is essential to any good Computer Scientist

Networks

And also to anyone who uses networks as part of their everyday activities

If more people realised quite how open, fragile and subvertable the Internet is, they would be a lot more circumspect in what they do on it!

Networks

The Internet is familiar to everyone here

But networks have been around for a long time

A network is any means to connect entities together so they can communicate

Networks

Reasons to network include:

- Resource sharing
- Communication and collaboration
- Information gathering
- Reliability through replication
- Entertainment

Networks

Existing networks include:

- The telephone system
- The mobile phone system
- TV and radio
- System control networks, e.g., Controller Area Network (CAN bus) in cars (and bicycles!)
- Sensor control networks, e.g., Bluetooth and ANT
- Cable (TV) networks
- The Internet

Networks

Metcalfe's Law

> *The value of a network expands exponentially as the number of users increases*

The bigger the network, the more links it has

Networks

There are many different kinds of network, thus meaning we need classifications to put things into easy boxes

But there are many classifications to choose from

Networks

Classification by size

- LAN Local Area Network
- MAN Metropolitan Area Network
- WAN Wide Area Network
- PAN Personal Area Network, WPAN (wireless PAN)
- and so on

Networks

Classification by speed~~speed~~ technology

- Narrowband
- Broadband

Actually these technical terms do not denote speed: their real meanings have been distorted by marketing

Optical fibre, while very fast, is actually technically narrowband

Exercise Find the technical meanings for narrowband and broadband

Networks

Marketing terms include:

- Broadband (xDSL)
- Fibre Broadband (xDSL)
- Full Fibre (fibre)
- Fast (?)
- Superfast (above 30Mbs)
- Ultrafast (above 300Mbs)

Networks

Classification by technology

- Voiceband modem (V series of standards, V.92)
- Local Wired (Ethernet)
- Medium distance wired. ADSL (ADSL2, ADSL2+, . . .)
- Optical Fibre (FTTP)
- Hybrid (VDSL with FFTC, G.fast with FTTdp, . . .)
- Cable Data Over Cable Service Interface Specification (DOCSIS)
- Local Wireless (Wi-Fi, Bluetooth, . . .)
- Longer distance wireless (3G, 4G/LTE, 5G, WiMAX, . . .)
- Very long distance wireless: satellite
- Power line
- etc.

Networks

Continuing Exercise Find the meanings for the various acronyms

Exercise Read some adverts for Internet connectivity products and determine what they actually are offering (e.g., “Superfast broadband fibre”)

Exercise And read about the controversies about how they advertise speeds

Networks

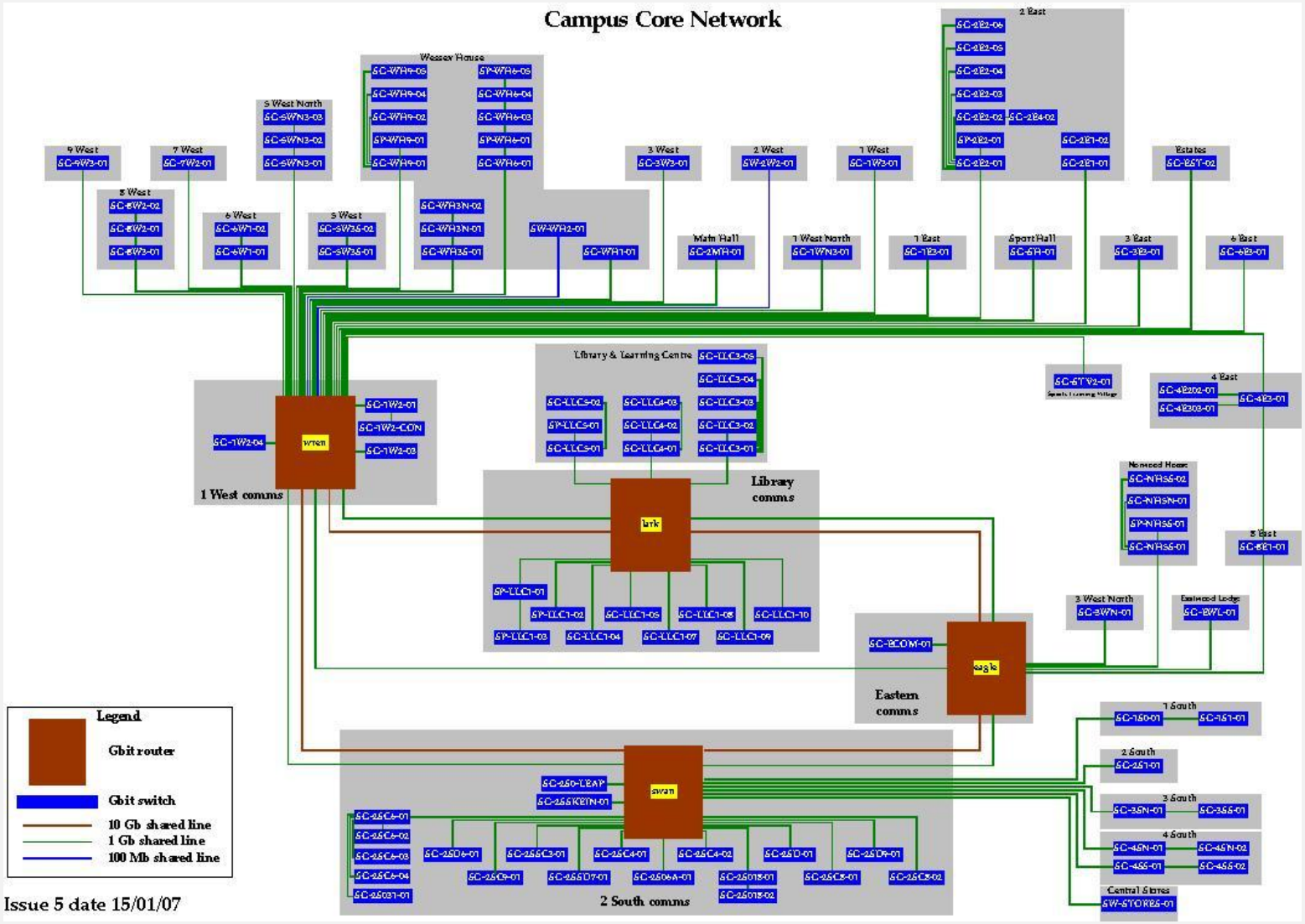
Exercise We use NFC to make contactless payments. Would you regard that as a network?

Exercise And what about Interplanetary networks?

Networks

So what does a typical network look like?

U of Bath Campus Network

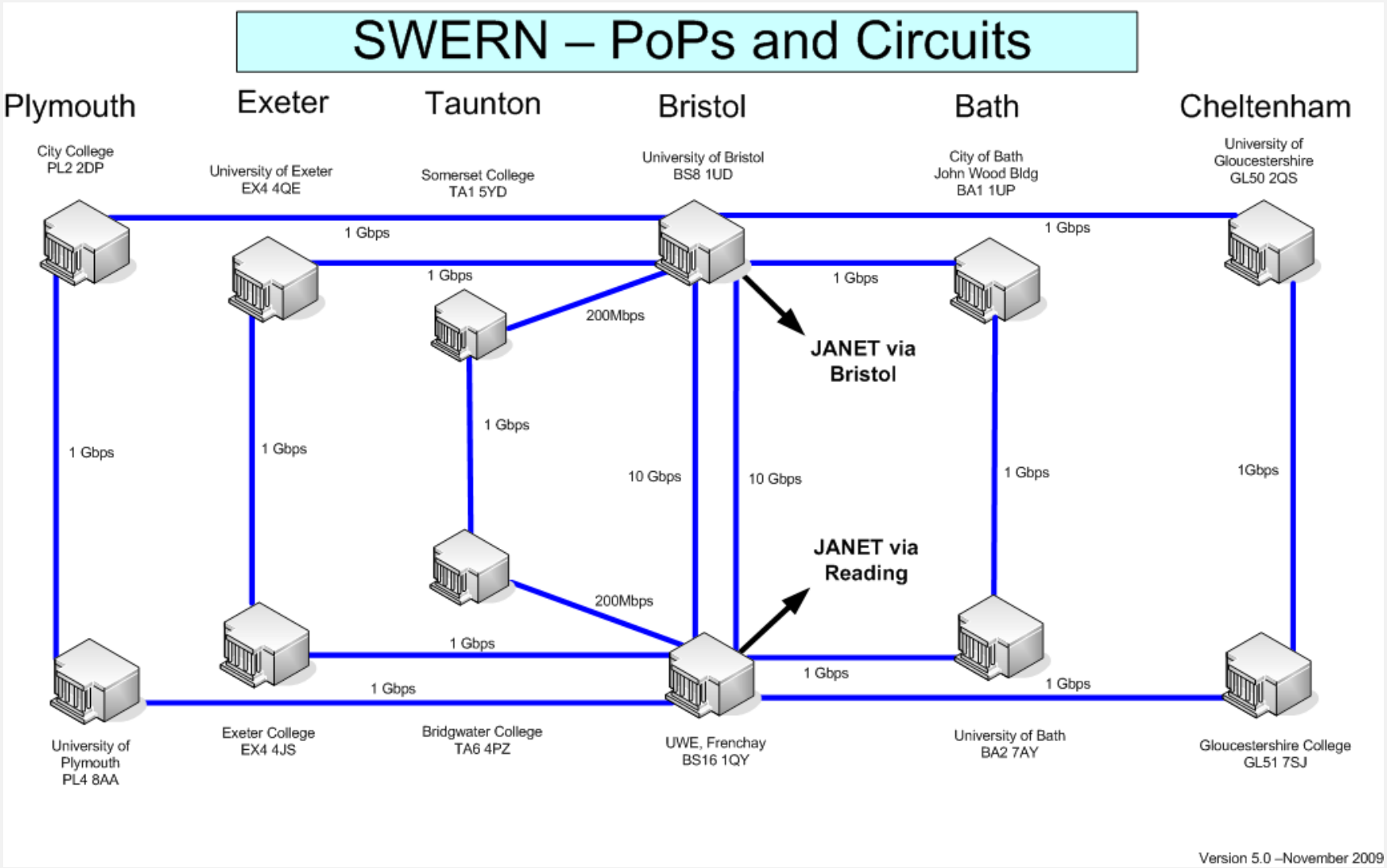


U of Bath Campus network

Networks

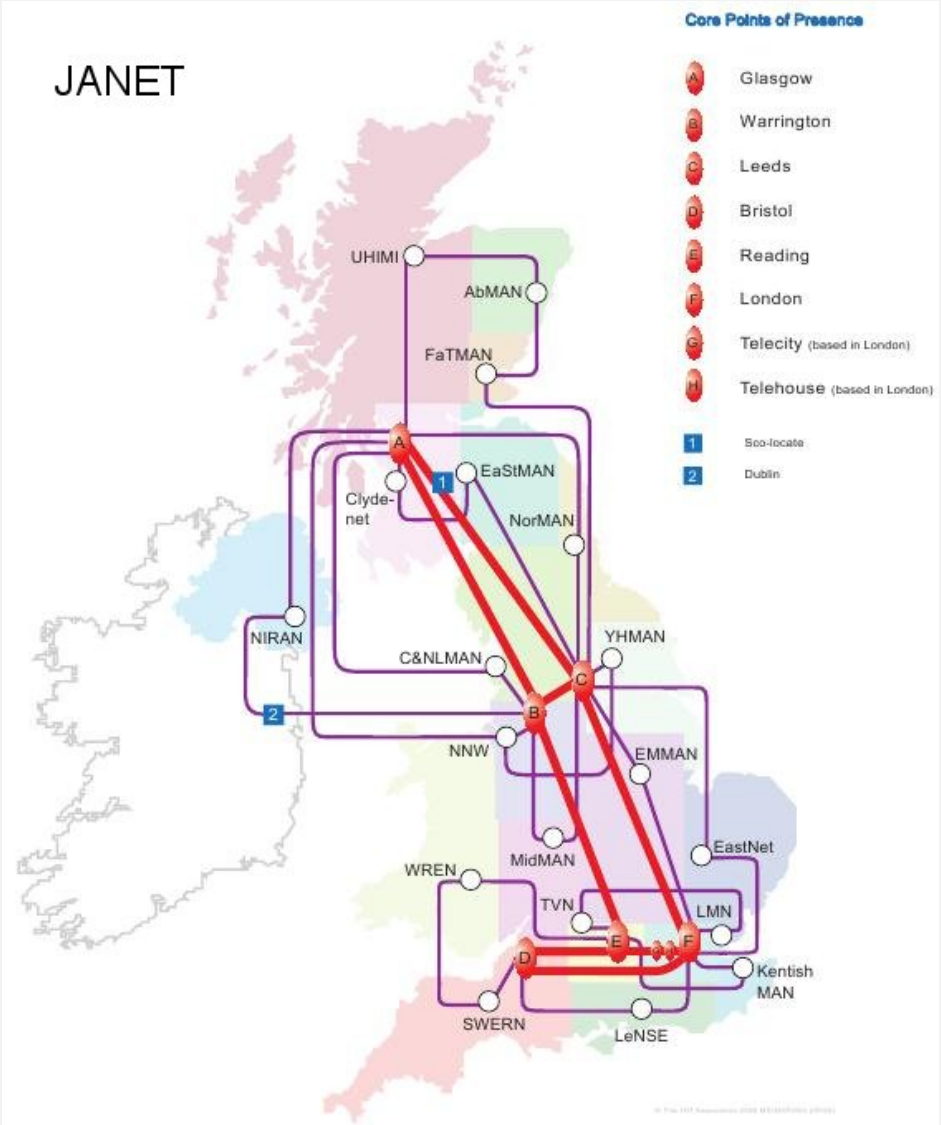
- No hosts shown: this is just the connectivity
- Multiple paths between points
- Gigabit and 10Gb links
- Other big networks, e.g., in CS, are not shown
- Connection to rest of world not shown

South West Regional Network



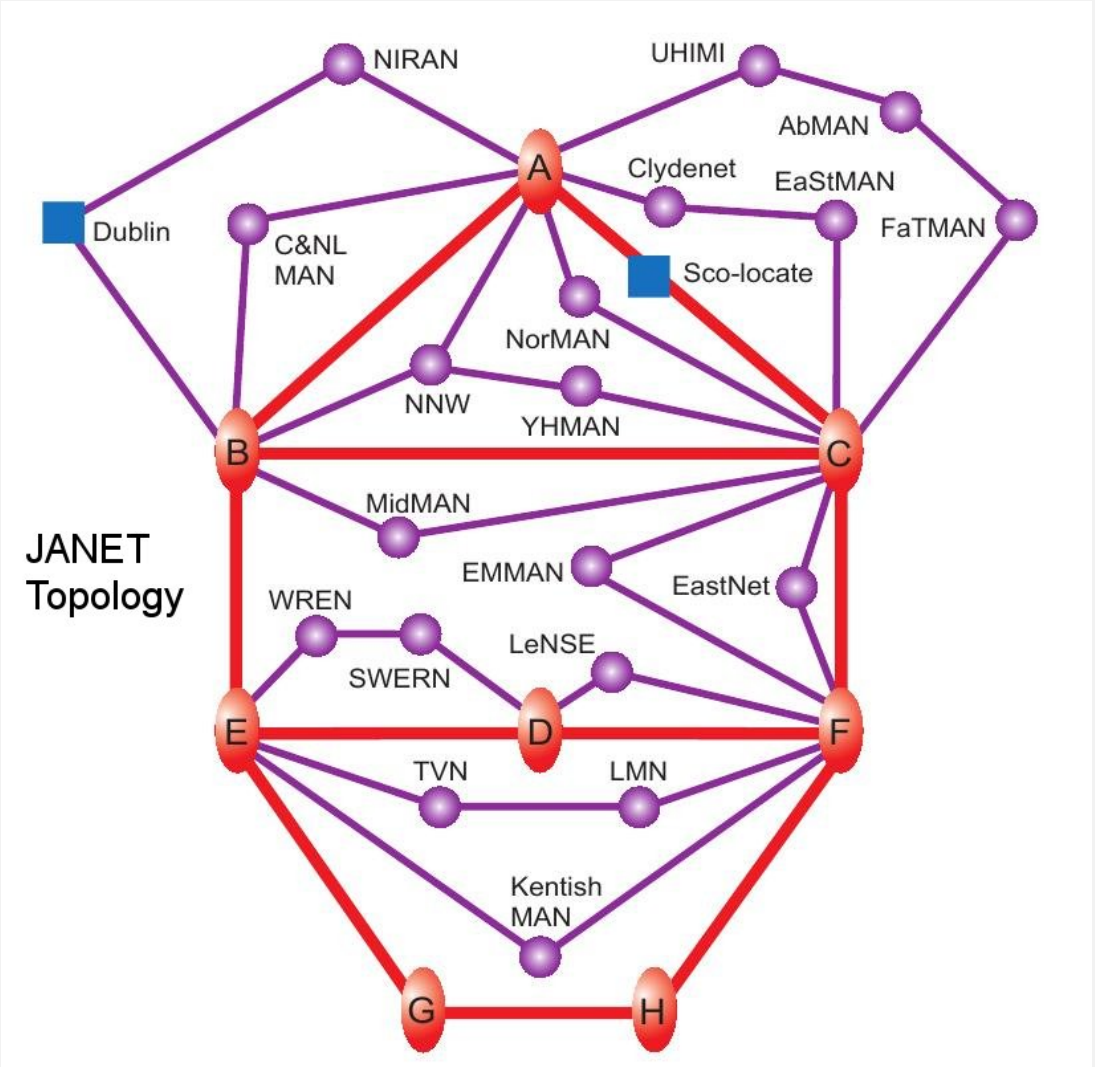
South West Regional Network (SWERN)

Joint Academic Network



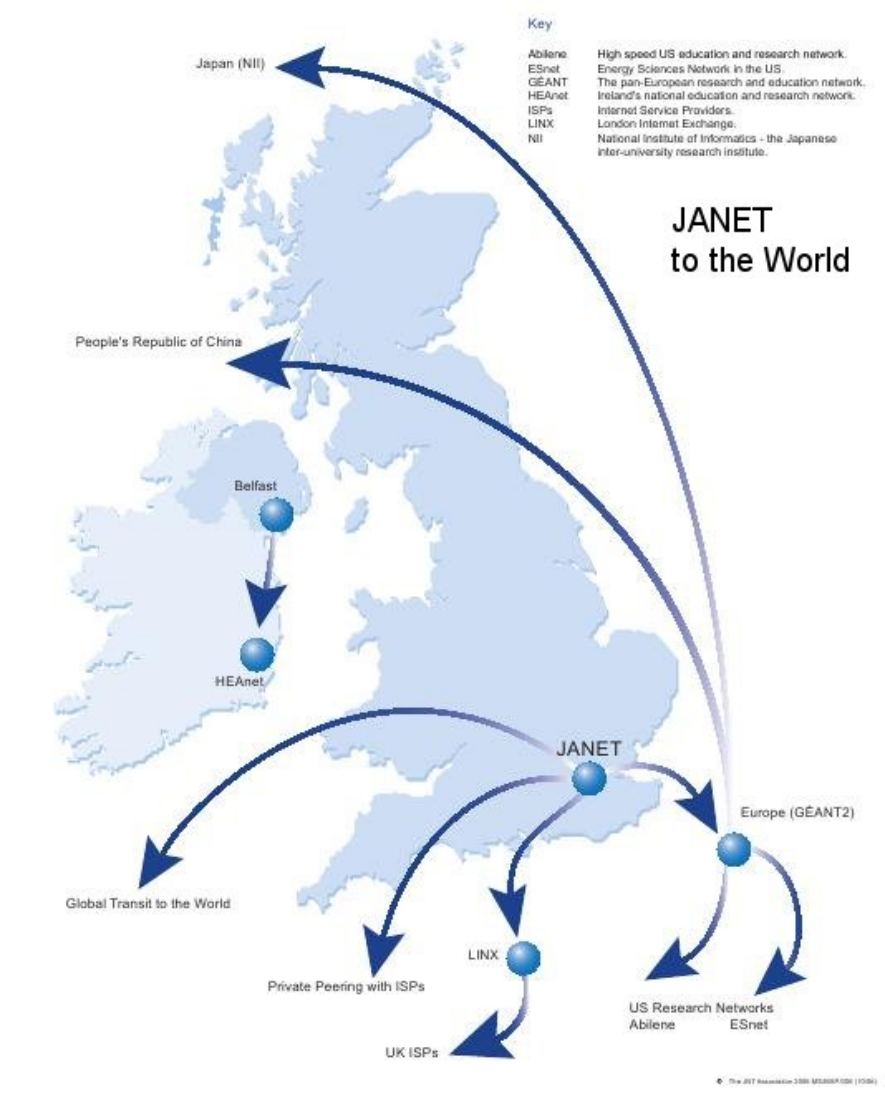
Joint Academic Network (JANET)

JANET



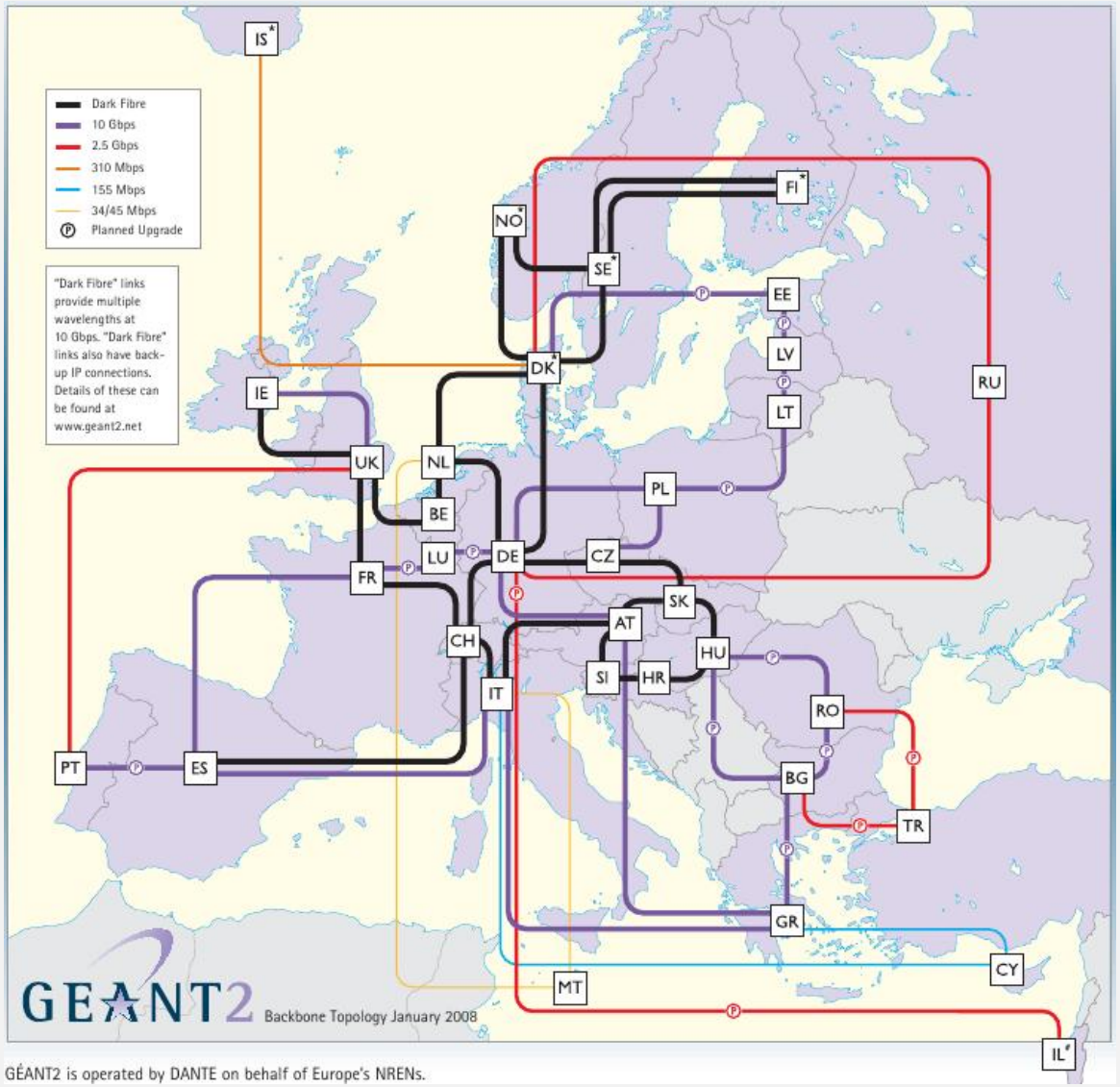
JANET Topology

Networks



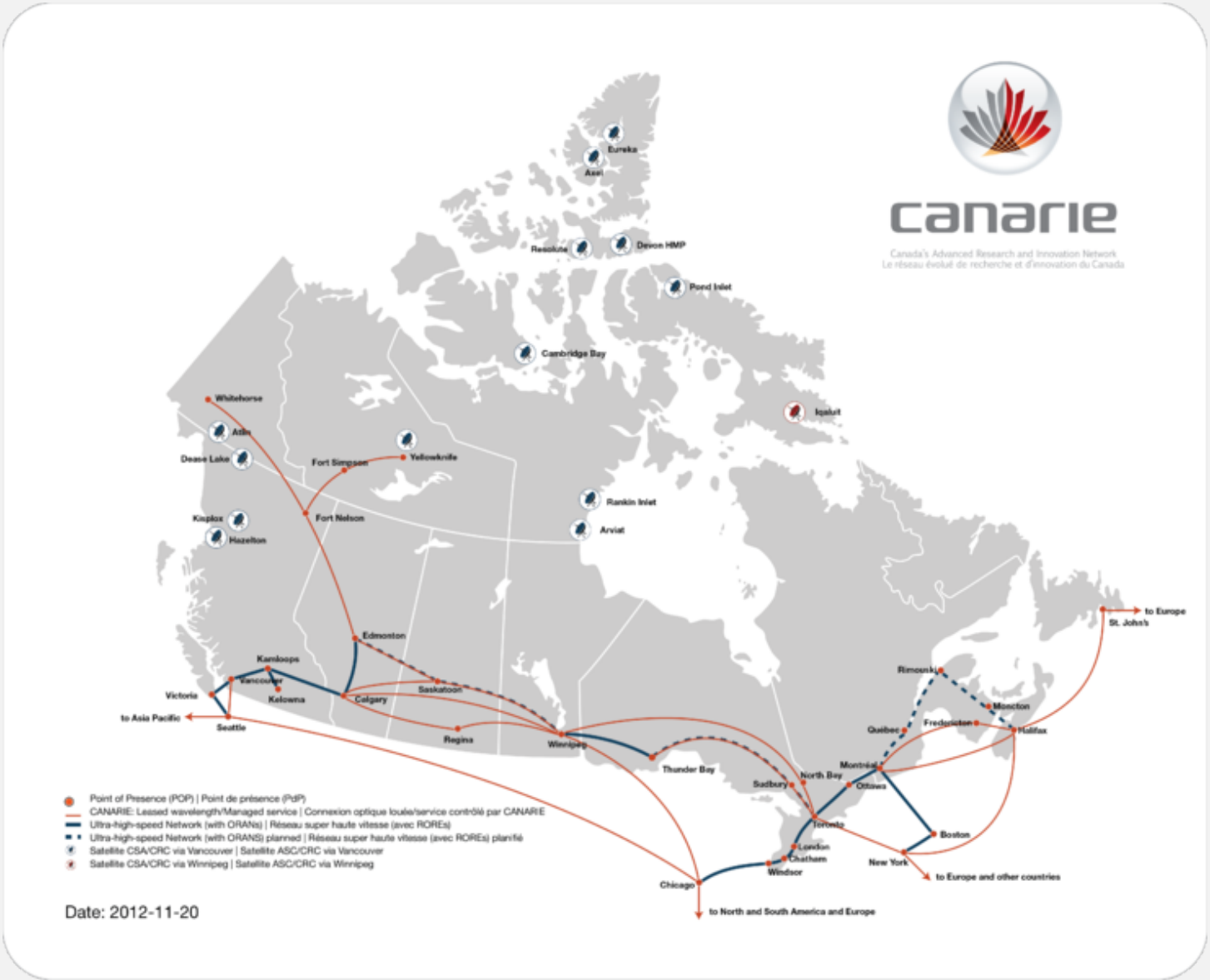
JANET connections to Internet

GÉANT



GÉANT European Network

CANARIE



CANARIE network in Canada

Hierarchy

We can see the Internet is a hierarchy of networks, managed by different groups

- department
- university
- region
- country
- world

for example

And this delegation of control is essential to the way the Internet works

Networks

Important Points

The “Internet” (capital “I”) is the world-wide collection of networks

An “internet” (lower “i”), an abbreviation of “internetwork”, is just some collection of networks

An “intranet” (with an “a”) is some collection of networks belonging to a single organisation

The Web is not the Internet

Anyone caught saying so will be laughed at and will lose marks in the exam

Networks

Tim Berners-Lee and Vint Cerf Front (photos from W3C)

Networks

Tim Berners-Lee and Vint Cerf Back (photos from W3C)

Networks

The basis of the Internet is *collaboration* between its member networks

Data travels from source to destination by being passed from machine to machine; from network to network

Networks

```
traceroute to www.youtube.com (208.65.153.238), 30 hops max, 40
byte packets
 1 fire.cs.bath.ac.uk (172.16.0.1) 0.166 ms 0.171 ms 0.216 ms
 2 gw.cs.bath.ac.uk (138.38.108.254) 0.570 ms 0.448 ms 0.337 ms
 3 swan-wren-10g1.bath.ac.uk (138.38.255.1) 0.430 ms 0.470 ms 0.352 ms
 4 7200-bath.bath.ac.uk (138.38.1.1) 1.190 ms 1.431 ms 1.356 ms
 5 fren-bath-ph.swern.net.uk (194.83.94.65) 3.198 ms 2.548 ms 2.515 ms
 6 so-1-3-0.read-sbr1.ja.net (146.97.42.157) 7.978 ms 7.859 ms 8.305 ms
 7 so-1-0-0.lond-sbr3.ja.net (146.97.33.142) 9.287 ms 9.468 ms 9.207 ms
 8 195.219.100.13 (195.219.100.13) 9.320 ms 9.553 ms 9.760 ms
 9 195.219.195.21 (195.219.195.21) 9.458 ms 9.401 ms 9.407 ms
10 ge4-1-0-1000M.ar3.LON2.gblx.net (64.208.110.81) 14.544 ms 17.433 ms
   13.969 ms
11 te1-1-10G.ar2.SJC2.gblx.net (67.17.109.102) 165.984 ms 167.465 ms
   169.402 ms
12 YOUTUBE-LLC.po1.401.ar2.SJC2.gblx.net (64.212.108.162) 165.040 ms
   167.189 ms 165.938 ms
13 youtube.com.hk (208.65.153.238) 165.972 ms 165.825 ms 165.815 ms
```

`gblx`: Global Crossing; `SJC`: San José, California

`youtube.com.hk` is in San José

Networks

This was done just after a major problem with the route to Youtube (a mistake in Pakistan lead to chaos, February 2008)

Allegedly, the Pakistan government was trying to censor a Youtube video by blocking all routes to Youtube in that country, but the block escaped to the whole Internet

A little later things settled down again. . .

Networks

```
traceroute to www.youtube.com (208.65.153.238), 30 hops max, 40
byte packets
 1 fire.cs.bath.ac.uk (172.16.0.1) 0.205 ms 0.210 ms 0.091 ms
 2 gw.cs.bath.ac.uk (138.38.108.254) 0.446 ms 0.431 ms 0.341 ms
 3 swan-wren-10g1.bath.ac.uk (138.38.255.1) 1.185 ms 0.841 ms 0.648 ms
 4 7200-bath.bath.ac.uk (138.38.1.1) 1.247 ms 1.062 ms 1.214 ms
 5 fren-bath-ph.swern.net.uk (194.83.94.65) 2.808 ms 2.438 ms 2.653 ms
 6 so-1-3-0.read-sbr1.ja.net (146.97.42.157) 7.839 ms 8.265 ms 7.798 ms
 7 so-1-0-0.lond-sbr3.ja.net (146.97.33.142) 9.526 ms 9.520 ms 9.726 ms
 8 po1-0.lond-gw-ixp2.ja.net (146.97.35.250) 9.672 ms 9.338 ms 9.089 ms
 9 195.66.226.185 (195.66.226.185) 9.804 ms 9.840 ms 9.926 ms
10 te7-3.mpd02.lon01.atlas.cogentco.com (130.117.2.26) 9.823 ms
   te2-1.3493.mpd02.lon01.atlas.cogentco.com (130.117.2.18) 10.223 ms
   te7-3.mpd02.lon01.atlas.cogentco.com (130.117.2.26) 9.685 ms
11 <snip>
19 * * *
20 youtube.com (208.65.153.238) 154.886 ms 156.732 ms 156.480 ms
```

Step 10: multiple probes go different routes

Step 19: a machine that refuses to respond to the probes

Host `208.65.153.238` is now named `youtube.com`

Networks

And again on 25 Sept 2017:

```
traceroute to www.youtube.com (216.58.204.14), 30 hops max, 60 byte packets
 1  fire-private.cs.bath.ac.uk (172.16.0.1)  0.109 ms  0.097 ms  0.088 ms
 2  gw-palo.cs.bath.ac.uk (138.38.108.254)  1.055 ms  1.048 ms  1.041 ms
 3  bath-gw-1-palo.bath.ac.uk (193.63.64.174)  1.608 ms  1.800 ms  1.703 ms
 4  xe-1-2-0.bathub-rbr1.ja.net (146.97.144.33)  1.287 ms  1.332 ms  1.330 ms
 5  xe-1-2-0.briswe-rbr1.ja.net (146.97.67.65)  2.286 ms  2.720 ms  2.707 ms
 6  ae22.londpg-sbr2.ja.net (146.97.37.201)  5.189 ms  4.652 ms  4.648 ms
 7  ae29.londhx-sbr1.ja.net (146.97.33.1)  5.089 ms  5.037 ms  5.012 ms
 8  193.62.157.22 (193.62.157.22)  5.270 ms  5.263 ms  5.246 ms
 9  108.170.246.225 (108.170.246.225)  5.938 ms  5.928 ms  5.869 ms
10  108.170.238.145 (108.170.238.145)  5.907 ms
    108.170.238.147 (108.170.238.147  6.141 ms  6.129 ms
11  lhr35s07-in-f14.1e100.net (216.58.204.14)  5.818 ms  5.820 ms  5.798 ms
```

Google are using a local server, probably in London

Networks

And again on 3 October 2019:

```
traceroute to www.youtube.com (216.58.198.174), 30 hops max, 60 byte packets
 1  fire-private.cs.bath.ac.uk (172.16.0.1)  0.197 ms  0.174 ms  0.149 ms
 2  gw-palo.cs.bath.ac.uk (138.38.108.254)  0.708 ms  0.682 ms  0.661 ms
 3  bath-gw-1-palo.bath.ac.uk (193.63.64.174)  1.776 ms  1.531 ms  1.856 ms
 4  xe-1-2-0.bathub-rbr1.ja.net (146.97.144.33)  1.074 ms  1.061 ms  1.047 ms
 5  xe-1-2-0.briswe-rbr1.ja.net (146.97.67.65)  2.113 ms  2.103 ms  2.092 ms
 6  ae22.londpg-sbr2.ja.net (146.97.37.201)  4.314 ms  4.329 ms  4.274 ms
 7  ae29.londhx-sbr1.ja.net (146.97.33.1)  5.163 ms  5.878 ms  5.854 ms
 8  193.62.157.22 (193.62.157.22)  5.587 ms  5.586 ms  5.544 ms
 9  * * *
10  172.253.71.200 (172.253.71.200)  7.069 ms
    108.170.238.118 (108.170.238.118)  6.627 ms
    172.253.68.210 (172.253.68.210)  6.284 ms
11  74.125.242.114 (74.125.242.114)  8.502 ms
    108.170.232.99 (108.170.232.99)  4.818 ms
    74.125.242.82 (74.125.242.82)  5.622 ms
12  lhr25s10-in-f14.1e100.net (216.58.198.174)  4.574 ms
    216.239.57.207 (216.239.57.207)  6.150 ms
    209.85.250.185 (209.85.250.185)  7.028 ms
```

Now much more variation in routes and multiple servers!

Networks

Mistakes in routing are not just ancient history: 4th October 2021 Facebook dropped off the Internet for 6 hours

A “misconfiguration” meant that its name servers (converting names like `facebook.com` to addresses) were not accessible

This took out Facebook, Whatsapp, Instgram, Oculus, Messenger, etc., and any site that uses Facebook to login

To the extent that the keycards on the doors to the machine rooms that Facebook engineers needed to get into to fix the problem were also not working

And the engineers couldn’t message the security guards with the backup keys, either!

Networks

History

The reason for this cooperative design comes from the history of the Internet

- 1957: The Soviet Union launches Sputnik
- mid 1960s: Advanced Research Projects Agency (ARPA) formed. A project to share expensive resources: namely their computers
- The network design was to be non-centralised to avoid single points of failure, particularly nuclear attacks
- So there is no single point of coordination or oversight of the network
- And there must be multiple paths between hosts

Networks

History

Using simple *circuits* (such as the telephone system used) between machines would be too vulnerable, so *packet switching* was devised

Data is chopped into small chunks, called *packets*, and each packet is sent individually, possibly over different paths

Individual packets might get lost, but others will get through

The original data are reconstructed at the receiving host

Networks

Warning

The word is “packet”, *not* “package”

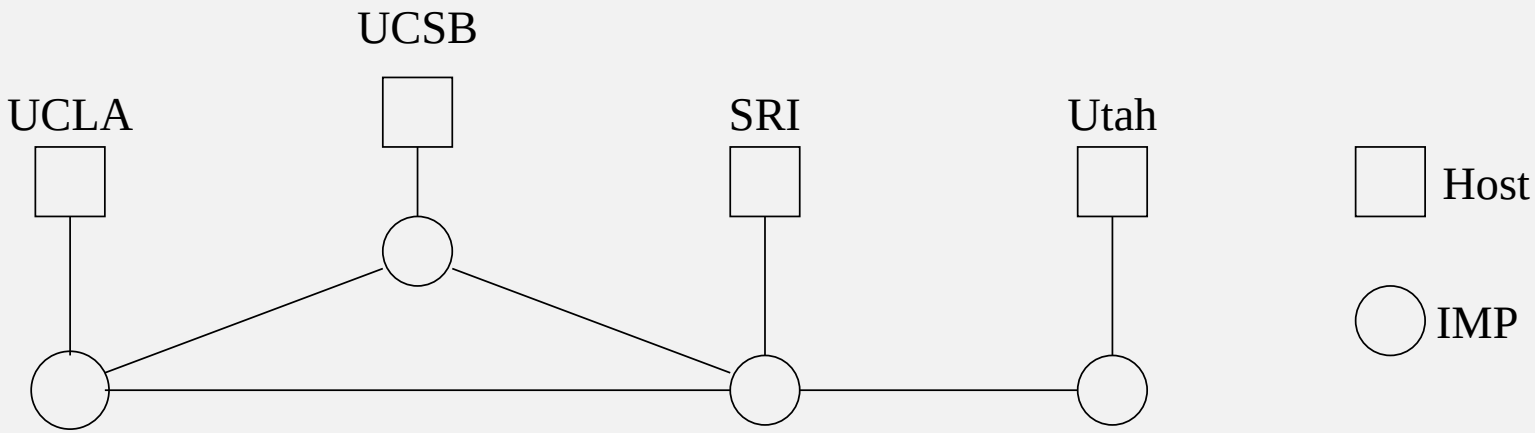
Take care never to use the word “package” in a technical context

Networks

- 1969 First Internet has just four nodes
- Runs NCP *Network Control Program*

Networks

History



The Original Arpanet, 1969; Separate *Interface Message Processors*

Networks

An Arpanet IMP (Wikipedia)

Networks

History

- Email and discussion groups are immediately popular
- 1973 Internet reaches London
- 1974 TCP/IP replaces NCP

Networks

History

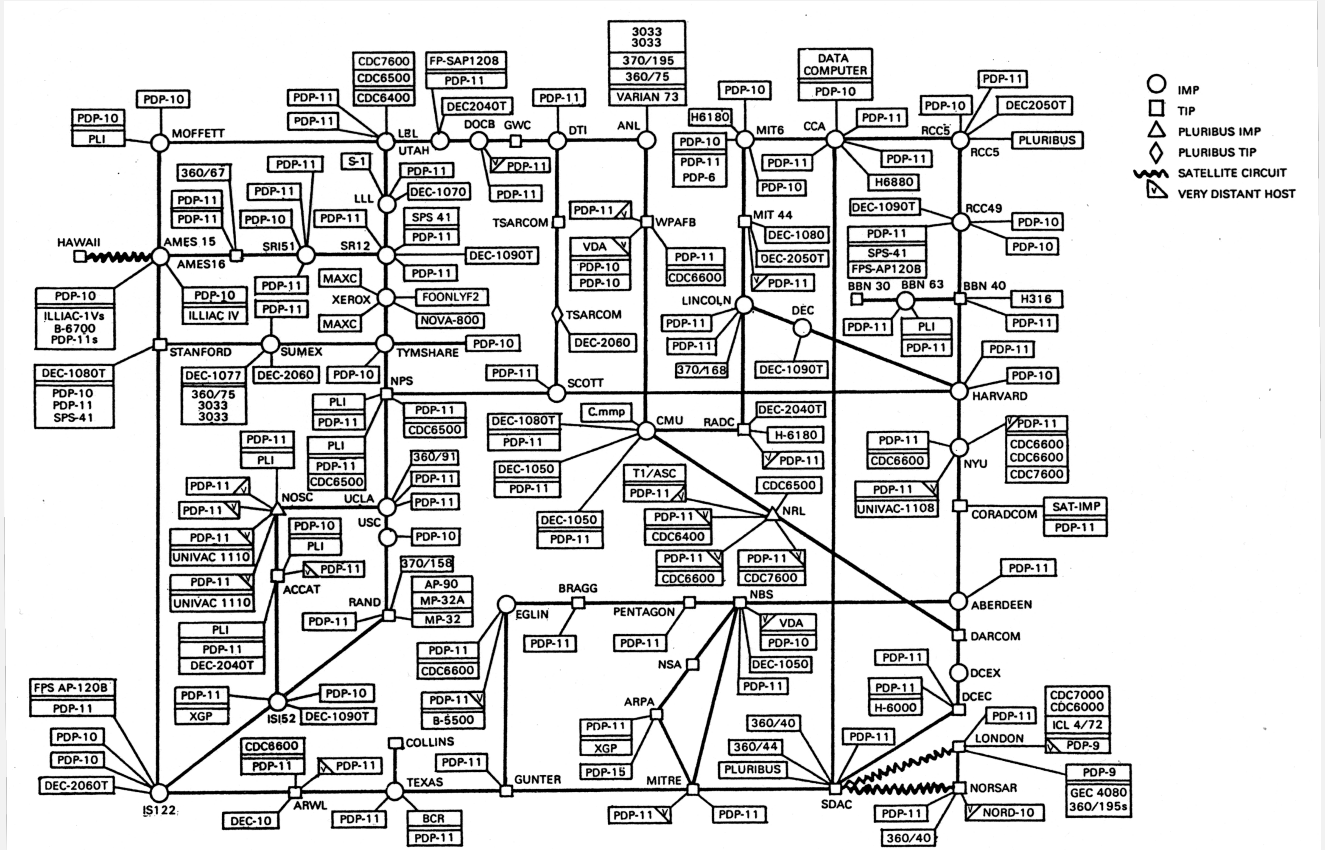


FIGURE 7.1: ARPANET LOGICAL MAP, JUNE 1979

Arpanet in 1979, from “Computer Networks, Fundamentals, Practice”; Bacon, Stokes, Bacon, 1984

Networks

History

- 1980s 1000s of machines on the Internet
- Domain Name System arrives
- 1980/90 Original ARPANET decommissioned and replaced
- Commerce arrives
- Other networks based on other protocols are replaced by the Internet
- 1992 1,000,000 hosts
- Gopher
- Tim Berners-Lee invents the Web

Networks

History

- The Internet starts to enter the home
- Microsoft gives up on its own network and falls into line
- The Dot Com boom
- The Dot Com crash
- Broadband to the home
- Large commerce over the Internet

Networks

History

- Mobile revolution
- “Social” media
- Internet of Things (IoT); blockchain; etc.
- . . . what next?

Networks

Already this decentralised and packet nature has implications on how the Internet must work

- how to chunk the data into packets?
- how are route(s) the packets use to get to their destination found?
- how do we reconstruct the original data as packets might be arriving in any order?

And higher level decisions like how we shall choose and build these multiple routes; what hardware to use, and so on

Networks

A packet doesn't know how to get to its destination

Even the source host doesn't generally know a route to the destination (only if the destination is on the local network)

A packet is like a postcard with the address written on it: it relies on the *routers* it passes though to make the right decisions

"Please forward me to `www.youtube.com`"

It is not like a car driver with their own map making their own decisions!

The question now becomes: how do the routers know what to do? More on this later

Networks

Protocols

To ensure maximum interoperability, the Internet relies on *standards* and *standardised protocols*

The use of standards means that machine A will be able to communicate with machine B even if A and B are made by completely different companies, are of completely different technologies and have never previously interacted

It is clear that you must have standards for interoperability in hardware: you can't plug a electrical plug into an optical socket

But you must have standards in the software too, as becomes clear when you try to use a Web page authoring tool that doesn't produce standard HTML

Thus we must have standards for the protocols

Networks

Protocols

This is somewhat akin to people agreeing all to use English (for example) to communicate

A pair of random people meeting can talk if they both know English

If not, the chances are that they share their native languages are quite small

Networks

Protocols

The Internet has a collection of specifications called *Request for Comments*, or *RFCs*

RFCs are freely available on the Internet for everyone to read and implement

The RFC philosophy:

> *Be as close to the RFC as possible in what you do yourself, but be as liberal as possible regarding what you accept from others*

Continuing Exercise When a topic is covered in lectures, read the relevant RFCs

Networks

Protocols

There are several bodies that oversee the structure and working of the Internet and the standards

- Internet Society (ISOC); oversees the Internet standard development processes
- Internet Architecture Board (IAB); ISOC committee that oversees the technical and engineering development of the Internet, particularly IETF and IRTF
- Internet Engineering Task Force (IETF); IAB committee that develops standards and publishes RFCs
- Internet Engineering Steering Group (IESG); executive sub-committee of IETF that has final say over RFCs

Networks

Protocols

- Internet Research Task Force (IRTF); IAB committee that does long-term research and development of Internet technology
- Internet Research Steering Group (IRSG); sub-committee of IRTF that manages the research groups
- Internet Corporation for Assigned Names and Numbers (ICANN); nonprofit internationally-organised organisation to oversee (sets policy) for global resources such as names and numbers or other identifiers
- Internet Assigned Numbers Authority (IANA); an affiliate body to ICANN that actually manages the domain names, IP addresses and other things, currently run by a company named “Public Technical Identifiers”

Networks

Protocols

IANA delegates management of various things to *Regional Internet Registries* (RIRs), e.g., domain names and addresses

Current RIRs:

- African Network Information Centre (AfriNIC); Africa
- American Registry for Internet Numbers (ARIN); North America and Antarctica
- Asia-Pacific Network Information Centre (APNIC); Asia, Australia, New Zealand
- Latin America and Caribbean Network Information Centre (LACNIC); South America
- Réseaux IP Européns Network Coordination Centre (RIPE); Europe, Russia, the Middle East, and Central Asia

Networks

Protocols

These are geographical, not political, regions

The RIRs further delegate things like management of domain names to commercial companies

E.g., 123-reg, GoDaddy and hundreds of others

Exercise Trace the movement of money up this hierarchy

Networks

Protocols

Outside ISOC, others important specification bodies include

- IEEE Institute for Electric and Electronic Engineers; hardware like Ethernet and Wi-Fi
- ISO International Standards Organisation; e.g., XML standards
- IEC International Electrotechnical Commission; e.g., Digital Living Network Alliance (DLNA)
- ITU-T Telecommunication Standardization Sector of the ITU (International Telecommunication Union); e.g., DSL standards
- lots more national and international institutions, such as the British Standards Institution (BSI)

Networks

Protocols

There is quite a lot of overlap in what these institutions cover

Sometimes one institution will take a standard from another institution and put a new cover sheet on it and give it a new name or number

For example, the JPEG standard, from the Joint Photographic Experts Group, is the same as ISO standard 10918-6:2013 and ITU-T T.872

Exercise Investigate these standards bodies

Networks

History

Anyway, we need a common “language” (the protocols) for a network

For the Internet, this common language is called the *Transmission Control Protocol/Internet Protocol* (TCP/IP)

This name is more historical than accurate, but to see what it means we need to think of *layers*

Networks

Layering Models

What do we need to make two computers communicate?

We need to connect them, so there must be some kind of physical (electrical, optical, radio or other) thing between them

So they must be compatible on voltages, how bits are represented as electrical or optical signals, etc.

And they must agree on how to represent data as bits: recall the different ways of representing signed and unsigned integers; similarly there are several ways of encoding alphabetic characters as bits

And the same problem for all other kinds of data: how to represent that sound or that shade of blue?

Networks

Layering Models

And technical requirements we have of the network

Such as do we make sure data arrived safely and didn't get lost or corrupted in transmission?

And a lot of other problems that only become clear when you try to build a network

Getting this right all at once is very difficult

Networks

Layering Models

So how should we implement a network system?

First we need a standard to follow

So how should we design a network standard?

The standard must address all the issues (and more) mentioned previously

Networks

Layering Models

This is too big a problem to be tackled all at once

How about chopping the design into chunks, each chunk having a well-defined functionality?

Note this is just the way we approach writing large programs

Except we are not writing a program here, we are designing a standard

So we slice the problem into nice, bite-size pieces, called *layers*

Networks

Layering Models

So what should the chunks be?

A *layering model* for a system is a suggestion on how you might want to slice up the problem of designing it all

An oft-misunderstood point is that a layering model is *not* a networking standard

It is a recommendation on how you *approach the design* of the standard

After you have written the standard, you can then make implementations

Networks

Layering Models

So:

- We pick a layering model
- We use this to guide us in making a standard
- We use the standard to direct the implementations
- We can then use the implementations

Note that there will likely be several differing implementations

But, if it is a comprehensive standard, and *if all the implementations follow the standard*, they will interoperate

Networks

Layering Models

For networks, there are two main layering models in use: the ISO Open Systems Interconnection (OSI) Seven-Layer Model; and the Internet Four-Layer Model

That is: two popular recommendations on how to design a networking standard

The OSI model is widely used while the Internet model is not, despite closely mirroring the Internet standard

Networks

The OSI Model

The seven OSI layers are

1. Physical
2. Data Link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

Networks

The OSI Model: Physical Layer

The *physical layer* (PHY) or *layer 1* is the hardware layer and deals with the transmission of bits over a channel

For example:

- what voltages to use or colours of light pulses or radio wavelengths to use
- what encoding for bits; how long (in time) a bit should be
- how many wires to use in a cable
- what plugs and sockets to use on the cable
- and many more

Generally, anything to do with choices regarding hardware

Networks

The OSI Model: Data Link Layer

The *data link layer*, also called the *media access layer* (MAC) or *layer 2*, takes the physical layer and tries to create a channel where there are no undetected errors of transmission

Note “undetected”: we know networks are not 100% reliable (e.g., wireless networks in particular) so we presumably want to take into account possible errors and deal with them: the ISO standard recommends you think about that here

A typical MAC layer sends the data as a sequence of *frames* (recall the packet nature of the Internet). A frame is a chunk of bytes, maybe tens or thousands of bytes long

Networks

The OSI Model: Data Link Layer

If a frame is corrupted, maybe the MAC layer can resend it; or send a message to the next layer indicating a problem

A popular choice in real standards is to do nothing at all: let a higher layer figure out what's gone wrong and choose a remedy

Again: it is up to the standard we are designing as to what actually happens. The layering model just says it is a good idea to consider this kind of thing here

In real implementations, this layer is often strongly intertwined with the physical layer and we tend to talk about both of them together

Networks

The OSI Model: Network Layer

The *network layer*, *layer 3*, controls the operation of the network, particularly the issue of routing data from source to destination

Also, it can deal with *congestion*: where there is too much data for a particular link it might route some data via another link, or use *flow control* to slow down the rate of transmission

Or speed up the rate if things are going well

Accounting might be managed in this layer: counting the number of bits so we can bill the user

And *quality of service*: e.g., ensuring there is always enough bandwidth to stream a video

Networks

The OSI Model: Transport Layer

The *transport layer*, *layer 4*, accepts data from the session layer (layer 5) and arranges it into packets suitable for the network layer: *packetisation*

Similarly, it takes packets from the network layer (layer 3) and reassembles them into the original data stream: *depacketisation*. This might need to deal with packets arriving out of order

You might want to think about reliability in this layer: ensuring the data received is the same as the data sent. No corruption or loss in the data

Curiously, reliability is not always a requirement of a network!

Networks

The OSI Model: Session Layer

The *session layer*, *layer 5*, manages *sessions* between source and destination.

- Establishing and terminating connections; e.g., a remote login session
- Restarting interrupted connections

Sessions can be quite short, e.g., just long enough for an email or Web page to be transmitted; or arbitrarily long

In general, a session is just some logically connected set of exchanges that have some unified identity

Networks

The OSI Model: Session Layer

For example, if the network crashes and reboots halfway through a big data transfer, the session can be picked up from where it left off, rather than starting again

You may already know that protocols like HTTP *don't* automatically pick up from where they left off

This tells us there is possibly a gap or omission somewhere in the relevant protocols: something they didn't address in the design

This may have been through deliberate choice; but it's equally likely they just didn't think about it

Networks

The OSI Model: Presentation Layer

The *presentation layer*, *layer 6* provides some things to help us retain the *meaning* of data

In particular, it decides on representations of data, such as characters, integers and floating point values, colours, sounds and so on so that the source and destination can agree on the data communicated

Networks

The OSI Model: Presentation Layer

So if the source wants to send the number 42, the presentation layer deals with encoding this in a suitable way as (say) some bits, which are then transmitted (passed to layer 5)

And the destination presentation layer can determine that this particular sequence of bits it has just received (from layer 4) represents the number 42

They can agree on “42” regardless of how each host chooses to represent integers internally

Networks

The OSI Model: Application Layer

The *application layer*, *layer 7*, is the layer application programmers use: ideally programmers would not have to worry about lower layers in their application

It contains protocols like HTTP for the Web, SMTP for email, and so on

Built on top of these protocols are the applications that the users see, e.g., Firefox or Chrome for the Web, Outlook or Thunderbird for email

Networks

Layering Models

Conceptually, data from an application is passed down through the layers until it reaches the hardware: i.e., through a sequence of pieces of software that perform the functions of each layer

As it passes from later to layer it is *encapsulated*: a transformation of the data in such a way that the layer below can cope with it transparently

And in a way that it can be untransformed back again

Networks

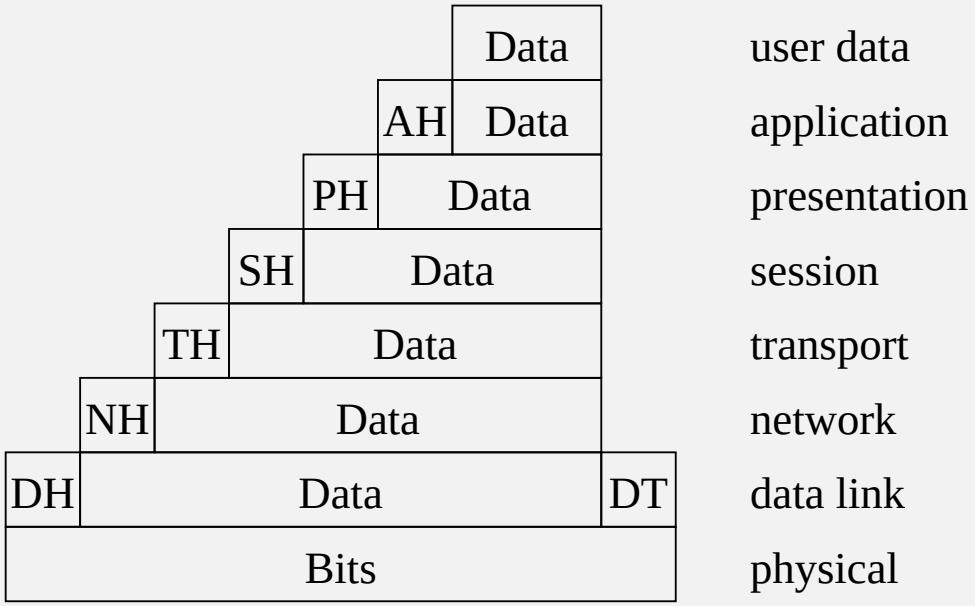
Layering Models

At each layer, the transformation might

- add an identifying header or trailer or both that is needed for the functionality of the layer
- encode any bit patterns that might be misinterpreted or mis-transmitted by the next layer
- put items in a standard form, e.g., integers into a well-known format
- do some arbitrarily complicated manipulation
- do nothing at all!

Networks

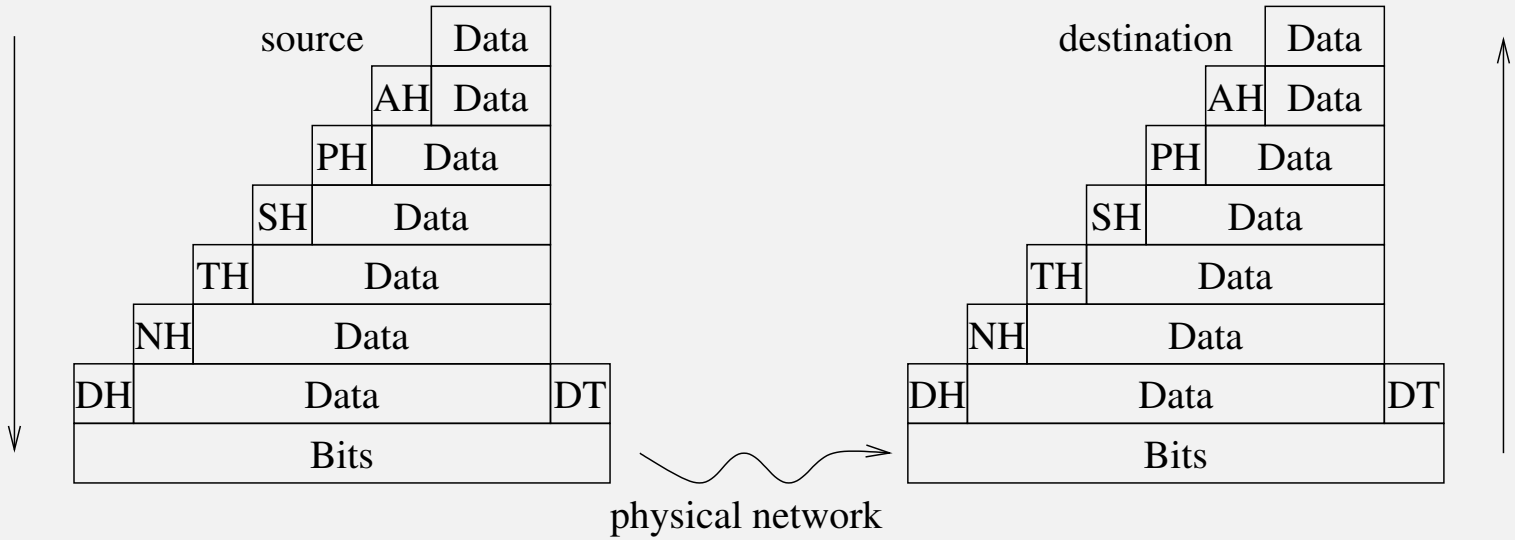
Layering Models



A possible (but unlikely) OSI encapsulation

Networks

Layering Models



Data is encoded and decoded

Networks

Encapsulation

An example. Some early modems treated byte values less than 32 as commands to the modem, not data to be transmitted

E.g., value 4 might mean “end of transmission” and the modem should drop the connection

What do you do if your data happens to contain the value 4?

You can’t just send it, as the modem would interpret the data as a command and end the connection

Networks

Encapsulation

So you need to transform the data somehow so that “4” is never seen by the modem in the datastream

And the transformation must be reversible, so the other end can reconstruct the 4

This is why encapsulation is necessary: so data can be transmitted accurately, even if you are using weird hardware

Networks

Byte Stuffing

In this case, the transformation often used was *byte stuffing*: the link layer could replace byte value “04” by, say, a pair of bytes “DB D4” (hexadecimal)

Both bytes will be transmitted unmolested by the modem

The link layer at the other end could recognise this pair and replace it by the single byte “04”

The “DB” is called an *escape character*, and its presence in the datastream means the next character is encoded, so special action must be taken

Networks

Byte Stuffing

Take a while to think of the issues this raises: what happens if our original data contained the pair of values “DB D4”?

We can’t just send “DB D4” as the other end will replace them by “04”

Not only do the bytes under 32 need to be stuffed, so does the escape character

For example, “DB” in the original data could be stuffed as “DB FF”

The datastream “DB D4” becomes “DB FF D4”

With byte stuffing, we exchange some expansion of the data for the correct transmission of that data

Networks

This kind of situation is why encapsulation exists

Of course, modern hardware doesn't act like early modems, but the principle remains

Networks

Layering Models

Say you want to send an email. In a strict implementation adhering to the layers the following might happen

- The email application might add a standard email header (`From`, `To`, etc.)
- This is passed to the presentation layer. As far as this layer is concerned it gets a chunk of text from the application layer
- It doesn't (or shouldn't) know that the first few characters are an email header
- It may transform the characters in some way, e.g., converting video into a transmissible format; it might prepend its own header to indicate what it has done

Networks

Layering Models

- This is passed to the session layer. As far as this layer is concerned it gets a bunch of bits from the previous layer
- It doesn't (or shouldn't) know that the first few bits are a layer header
- It may transform the bits in some way; it might prepend a header to help it manage sessions
- And so on down through the layers

Eventually, the physical layer transmits some bits

Networks

Layering Models

At the destination a bunch of bits is received by the hardware

We now proceed up the layers, unwrapping and untransforming as we go

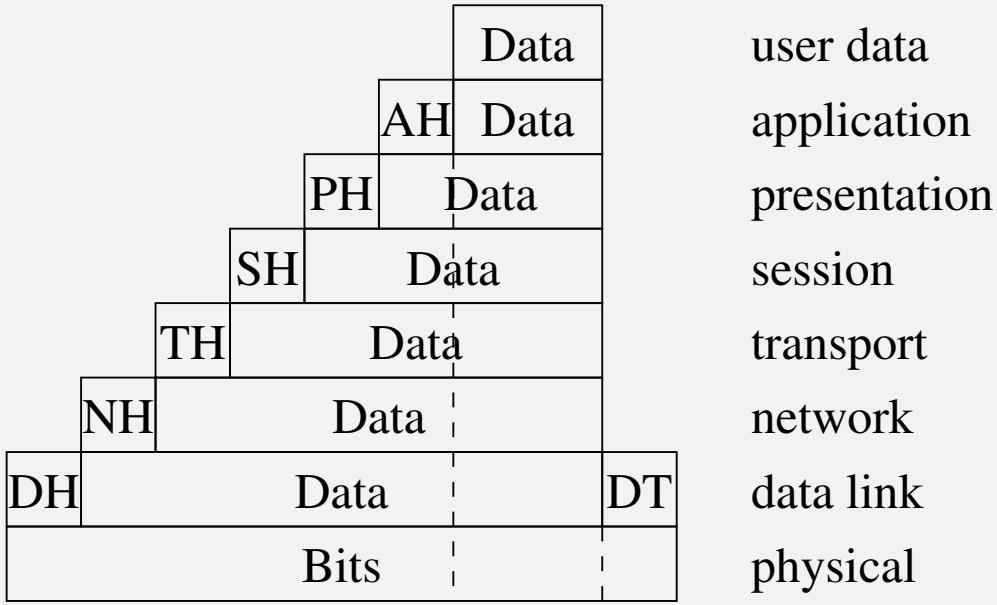
And, eventually, we get the original data arriving at the application (we hope)

So why do this as it seems so wasteful?

Networks

Layering Models

If the original data are small the data transmitted on the wire can be mostly headers from the various layers



Encapsulation overhead

Networks

Layering Models

Surely it is easier just to put the original data on the wire?

- Encapsulation adds complexity to the implementation
- It adds overhead (both space and time)
- thereby reducing effective throughput

But it turns out layering and encapsulation actually *reduces* overall complexity, just like breaking a large program into functions/objects/whatever does for programming

It also gives flexiblity

Networks

Layering Models

Suppose we have a 1Gb network card in our machine and someone comes along with a 10Gb card

Because the physical layer is (mostly) separate from the data link layer we can just write a new standard for the 10Gb physical layer and slot it in where the old 1Gb standard was

The upper layers needn't know anything has changed

And we can slot in the implementation for the new hardware in exactly the same way

We don't have to rewrite our email application (and Web browser, and all our other applications) because of the upgrade

Networks

Layering Models

Similarly for all the other layers: we can replace specifications in a layer and implementations of those specifications without affecting the rest of the stack

In principle, you could use carrier pigeons as the physical layer and your browser should work unchanged

Apart, perhaps, from speed

Someone really did this once!

Exercise Read RFC1149

Networks

Layering Models

And as each layer simply hands over to the next, it doesn't actually matter what the next layer "really" does

As long as it has the right behaviour, it doesn't matter how it is actually implemented

This enables useful tricks like *tunnelling*, which we shall look at later

Networks

Layering Models

Why seven layers in the ISO model?

History is a trifle vague on this, but it seems that IBM had a seven layer protocol and managed to persuade the ISO committee in charge of the model

Some people advocate more layers: e.g., splitting the hardware layer up

For example, a sublayer describing the physical medium, such as copper or fibre; and a sublayer describing the signals in that medium, such as various kinds of electrical signalling

Exercise Reality is complicated. Read IEEE 802 to see how the physical layer can be split into three sublayers; and the link layer can be split into two sublayers

Networks

The Internet Model

Others want fewer layers. A good example is the Internet Model

This is a four layer model, developed post-hoc after the Internet Protocol had gained prominence (RFC1122)

- Link Layer
- Network Layer
- Transport Layer
- Application Layer

Networks

The Internet Model

We shall describe this model, together with its primary instance TCP/IP

Take care to distinguish between the model and the instance

They often get confused as they seem so similar

It is possible, though unlikely, that there could be another network protocol, not TCP/IP, based on the four layer Internet model

Networks

The Internet Model: Link Layer

The Internet link layer corresponds to the OSI physical plus data link layers

The model does not say much about this layer, only that it has to be capable of sending and receiving the next layer packets

So what you do with your hardware is pretty much open

TCP/IP many realisations here, including Ethernet, VDSL and Wi-Fi

And pigeons

Networks

The Internet Model: Network Layer

Also known as the *Internet* layer, the network layer handles the movement of packets, particularly routing

This directly corresponds to the OSI network layer

In TCP/IP, the *Internet Protocol* (IP) is defined in this layer

IP is an *unreliable* protocol. This is a technical term that means that it does not guarantee delivery of packets

unreliable $=$ not guaranteed reliable

Networks

Aside: Reliability

Sometimes it is better to deal with an occasional lost packet than to hold up the system while the lost packet is re-requested and resent, e.g., video, where fast delivery is more important than accurate delivery

So it is quite useful to have a “unreliable” delivery sometimes

A lot of Internet hardware is actually fairly reliable (non-technical sense) these days

But wireless (Wi-Fi, etc.) and some wired (DSLs) are more unreliable than you might think

Networks

The Internet Model: Transport Layer

The transport layer corresponds to the OSI transport layer, providing a flow of packets between source and destination

In TCP/IP, two protocols are in this layer: the *transmission control protocol* (TCP) and the *user datagram protocol* (UDP)

Networks

The Internet Model: Transport Layer

TCP is a *reliable* (guarantees delivery) protocol

It makes a reliable layer out of a potentially unreliable IP underneath by a complex mechanism of packet acknowledgements

We don't always want to pay the non-trivial cost of that mechanism, so the other protocol, UDP, is not reliable

Actually, it is reliable as the underlying layer, IP, is reliable

And IP is as reliable as its underlying physical/datalink layer is reliable

Networks

The Internet Model: Transport Layer

UDP was devised long after TCP when it was realised how useful unreliable protocols can be: this is why the protocol set is called “TCP/IP”, as that was the entire protocol set for a fair while

Networks

The Internet Model: Transport Layer

We shall see packets have a header field indicating what the protocol of the data in the packet is

TCP has protocol number 6

UDP has protocol number 17

Exercise Have a look at “Protocol Numbers” at
`https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml`

Networks

The Internet Model: Application Layer

The application layer covers (roughly) the OSI session, presentation and application layers

This means, in particular, Internet *applications* must take care over presentation issues if they want to be completely interoparable

Many forget this, e.g., many programmers forget that not all machines represent integers in the same way and so the bit pattern they use for the number they want to send is (mis)interpreted as a different number by the receiver

Networks

The Internet Model: Application Layer

In terms of implementation, typically an OS kernel will implement everything below the application layer (TCP, UDP, IP, Ethernet, Wi-Fi, etc.)

This is because they use system resources that must be shared fairly amongst applications

Anything *above* the transport layer must be done by the application programmer in their application code

Networks

The Internet Model: Application Layer

So a typical email application will need to apply a presentation encapsulation and add application layer headers (To, From, etc.)

The Multipurpose Internet Mail Extensions (MIME) standard is a way to encode data (e.g., text, sound, pictures, video) in a safe way

Originally developed in the context of email, it is now used in other areas like Web page delivery where there are mixed kinds of data to transmit

Networks

The Internet Model: Application Layer

Similarly for the session layer

If a persistent session is needed, the application must code it

Many applications, like Web browsers using HTTP, don't

Note: if the TCP/IP had session management, applications would get this "for free"

The counter-argument is that many applications do not want session management, and should not have to pay the overhead of supporting it

Networks

The Internet Model: Application Layer

In the real world, each application (running over TCP/IP) that needs session management has to re-implement it for itself

Of course, libraries of code exist to do these “missing” things (sessions, presentation and so on), but the programmer must write the code to incorporate them

Networks

Example of layering in practice: how an email might be transmitted over an Ethernet

- We start with the text of the email
- Application: the email application transforms the text using a MIME encoding (presentation)
- Application: the email application adds an *envelope* header (From, To, etc.)
- Transport: TCP adds its header (reliability)
- Network: IP adds its header (routing)
- Datalink: Ethernet adds a header (local routing) and a trailer (checksum)
- Physical: The bits are transformed using a 4B/5B encoding to smooth the bit patterns and are sent using a three-level electrical coding MLT-3 (physical)

Networks

Going through all these in detail is the content of this Unit

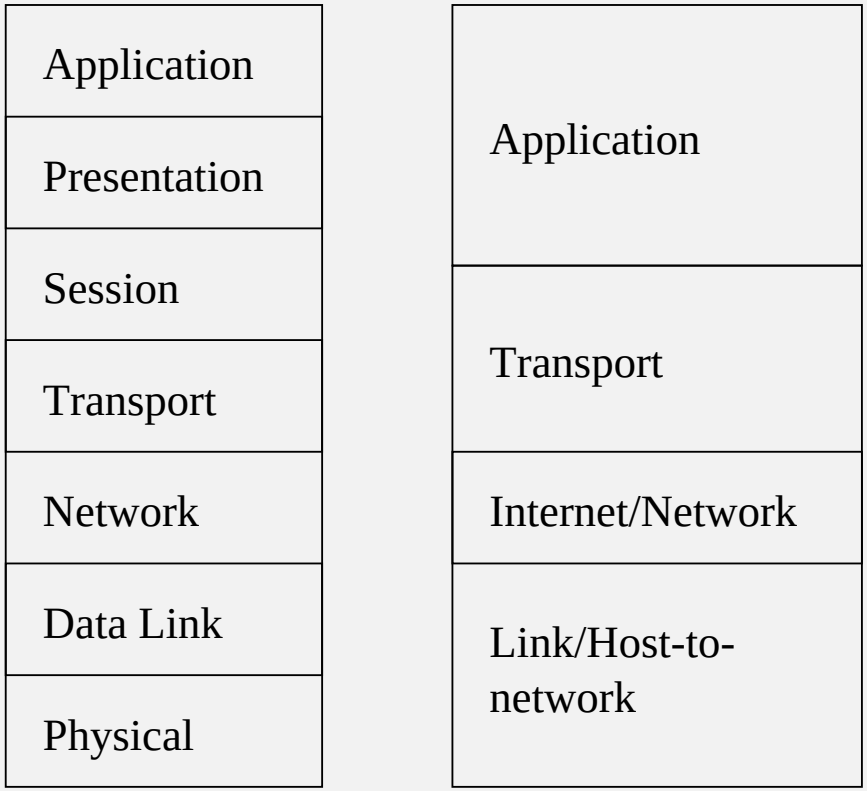
Networks

We have two layering models, ISO and Internet, two approaches to designing a standard

How do they compare?

Networks

Layering Models



OSI vs. Internet Models

Networks

Layering Models

Comparing the two models:

- OSI was developed before an implementation; the Internet Model was created *after* TCP/IP
- OSI make a clear distinction between model and implementation; Internet is fuzzy
- OSI is general and can apply to many systems; Internet is specific, namely to TCP/IP
- Implementations following standards following the OSI model were dire; TCP/IP is wildly successful

Networks

Layering Models

Problems with the Internet Model (*not* TCP/IP) include

- it is only good for describing TCP/IP
- the physical and data link layers are merged; this makes it difficult to talk about, say, copper vs. optical fibre installations

Networks

Layering Models

Non-problems include

> "OSI is slower as data has to go through more layers"

This is confusing the model with the implementation and ignoring the standard in between them

An implementation need not have 7 separate modules: it only needs to *behave as if it did*

Early implementations of a standard derived from OSI made this mistake

There are good CS reasons why we should do this separation, but practically we have to make tradeoffs between maintainability and speed

Networks

Layering Models

"OSI has larger encapsulation overhead as data has to go through more layers"

As above

And you don't *have* to add a header at every layer: it depends on what the standard requires

The model doesn't *require* anything

Networks

Layering Models

"There are no decent implementations of OSI"

Again, confusing a model with a standard

And TCP/IP can be regarded a standard that fits the OSI model, anyway

If you squint a bit

Networks

TCP/IP

The OSI model is widely used; the OSI protocols never

The Internet model is rarely used; the TCP/IP protocols are everywhere

The main reason that TCP/IP is so successful is that its standards (RFCs) are open and freely available: anyone can join in

Furthermore, the code was also free and widely available

Not brilliant quality, but at least it worked. . .

Networks

TCP/IP

Networks before the Internet tended to be closed and proprietary, where you had to pay to get in

But all these failed to get critical mass: even Microsoft failed to get their own alternative to the Internet off the ground and they had (grudgingly) to join with the rest of the world in using TCP/IP

Networks

Layering

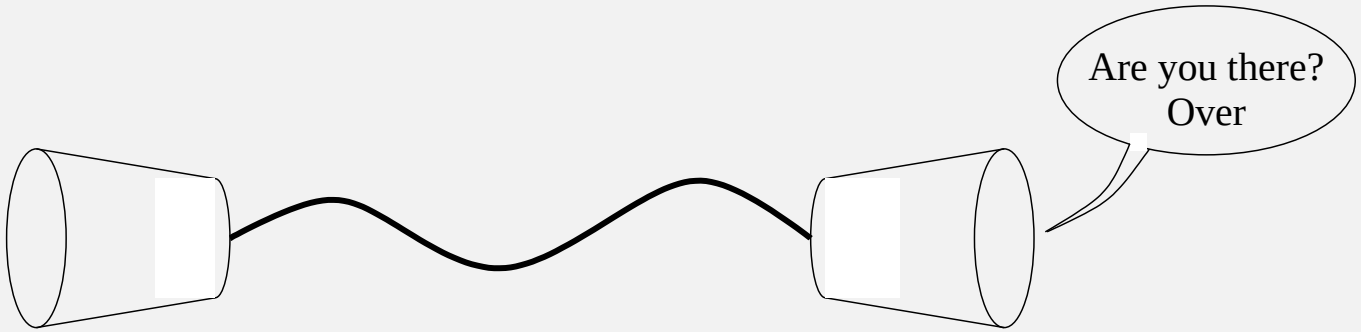
Other layering models exist, e.g., *Tanenbaum's Five Layer Model*

- Physical
- Data link
- Network
- Transport
- Application

Still missing the essential presentation layer, but a lot more useful in a world where the physical layer is often changed, e.g., 1Gb Ethernet to Wi-Fi

Networks

Layering Models



Cup and string network

Exercise identify the OSI and Internet layers as they apply to a cup-and-string network

Exercise Read section 3 of RFC3439

Networks

Security in the IP

Despite being initiated by the Military (ARPA), the Internet protocols were mostly designed(?) and developed in Academia

This has had a great effect on the *security* of the Internet

The Internet was developed in a “safe” academic environment where little regard was given to issues of privacy or authentication

And the models are also weaker on security than they ought to be

Networks

Security in the IP

OSI says “security should be involved at all layers”. Not particularly helpful

The Internet Model says even less

Compounding the issue of lack of support for security in the Internet protocols, early TCP/IP implementations were woefully poor: many exploitable bugs

Networks

Security in the IP

By default:

- Data in transit is east to read and modify as it is passed through the various machines on the path to the destination
- Many protocols used are not resistant to malicious interference
- Authentication mechanisms are weak to non-existent

And the implementations were very fragile and easily hacked

Networks

Security in the IP

Note the two separate issues here:

- the protocols are fragile and easily breakable
- the implementations of those protocols were often poor

A good implementation of a bad protocol is bad

A bad implementation of a good protocol is bad

Networks

Security in the IP

Many of these issues have since been tackled (not always successfully), particularly when commerce got involved

But there are still several areas that could be improved: see the routing to Youtube problem earlier; and that wasn't even maliciously intended

New protocols and secure (we hope) extensions to existing protocols are now available: e.g., HTTPS for the Web, SMTPS for email

Management and use of cryptography has an overhead. This is an extra workload on servers: some people are unwilling to pay this price

More on security later

Long term plan

We shall now work our way up the layers, looking in detail at what TCP/IP does for each

This is going to be a long journey!

Networks

Hardware

First, hardware

There are several popular hardware implementations. Some you should have come across are

- Ethernet: a wired network
- ADSL and VDSL: telephone networks
- Wi-Fi: a short range wireless network
- Cellular: mobile phones

We shall look at some of these

Networks

Hardware

Exercise How many different radio/wireless systems does your mobile phone support?

Networks

Ethernet

Ethernet arose in 1982, from DEC, Xerox and Intel, based on the earlier *Aloha* protocol

The original Ethernet supported 10Mb/s

Note: Mb/s = megabit/sec; MB/s = megabyte/sec

In comparison, current consumer Ethernet runs at 1Gb/s, while typical top-end Ethernet runs at 100Gb/s, with 400Gb/s starting to be used in datacentres and plans for 800Gb/s and 1.6Tb/s

Networks

Ethernet

To be a bit more precise, the original Ethernet had a 10Mb/s *signalling rate* (also known as *line rate*)

The signalling rate is the rate of delivery of bits across the physical network

Due to layering encapsulation and other physical overheads, this is overwhelmingly *not* the rate of delivery of bits to the application you are running

For example, there is always a gap between packets where data is not being transmitted!

Networks

Ethernet

However, the signalling rate is the number marketers like to use

The rate actually realised can be much lower; e.g., a 54Mb/s Wi-Fi 3 (802.11g) network might only deliver half that figure to an application

Networks

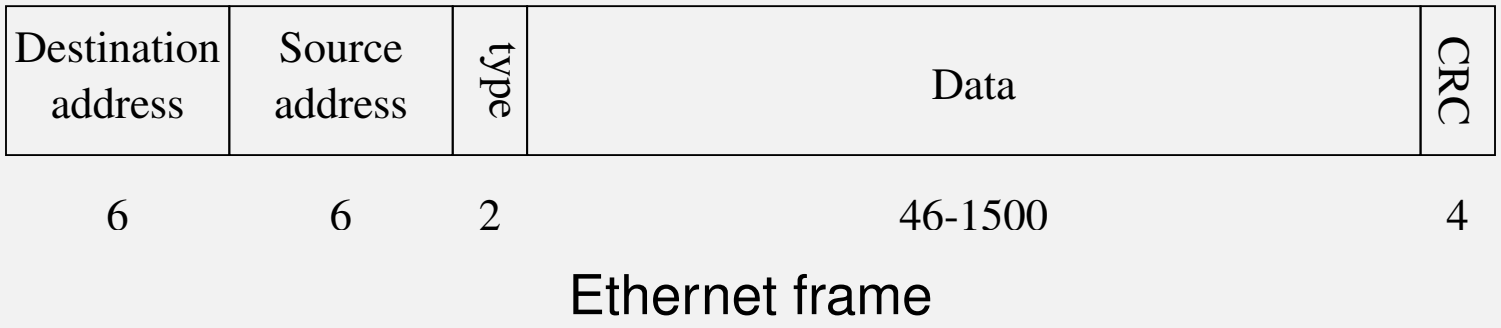
Ethernet

The Ethernet standard covers both the PHY and the MAC layers, so we shall look at them together

And we begin with the frame format

Networks

Ethernet



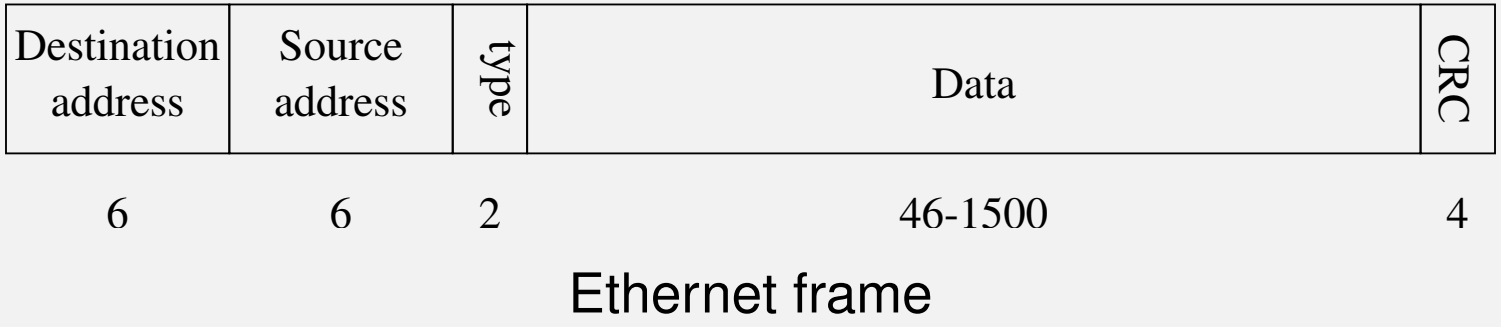
Ethernet frame

Numbers are byte counts: so, e.g., the destination address is 6 bytes long

- 2 byte type indicates what kind of network layer data follows, e.g., (hex) 0800 for an IP packet
- The data, maximum 1500 bytes
- **Minimum 46 bytes**. The data must be padded with extra bytes if fewer than 46 bytes are supplied

Networks

Ethernet



Ethernet frame

- A higher layer must detect and remove this padding when necessary
- 4 byte checksum, also called *cyclic redundancy check* (CRC)
- Use to check for corruption errors in the frame

Networks

Ethernet

The sending host fills in the other fields and computes and fills in the CRC

The receiving host computes the CRC on what it gets and compares with what is in the CRC field

If they differ, it is very likely the packet was corrupted

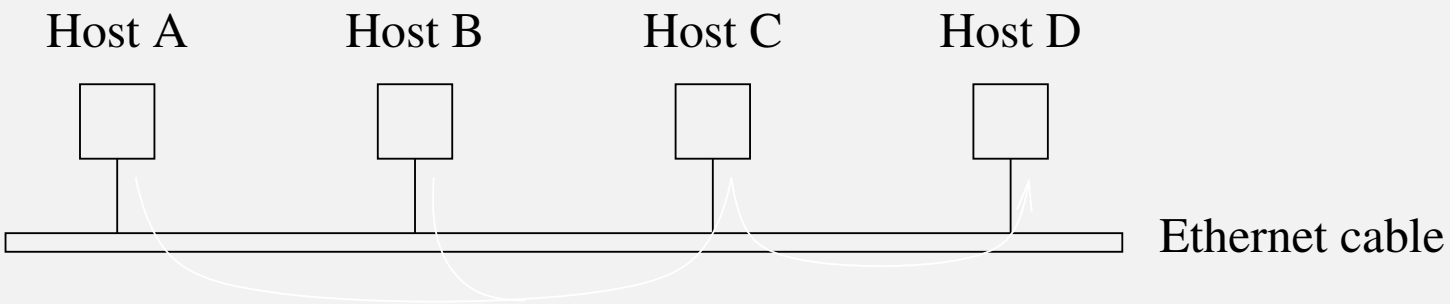
(There is a small chance that the CRC alone got corrupted and the other fields are good; or an even smaller chance the frame *and* the CRC both got corrupted in ways they still match)

Ethernet just drops corrupted frames; no more action is taken

Networks

Ethernet

(Original) Ethernet is *shared*, so every host sees every frame on the local network



Original Ethernet

So how is a frame matched up to the intended destination host?

Networks

Ethernet

Every Ethernet card has a unique address built into it

(Not the full story, but true enough for now)

So the destination address on the frame allows an Ethernet card in a host to recognise that a frame is for it and so can read and process it

There is a security issue here. . .

Networks

Ethernet

The source address on the frame allows a host to determine who sent the frame and so it can reply if needed

`000010000000000001000001001101000110100111011101` is an example Ethernet address, a 48-bit value

For convenience we write this as `08:00:20:9a:34:dd`, six hexadecimal numbers

Networks

Ethernet

This address is enough for when the destination is on the local Ethernet network: we have to work harder if the destination is non-local

And the destination might not be on an Ethernet, so how can we specify such a destination?

This is the job of the next layer, IP, which we look at later

Ethernet is purely a local area network technology

Networks

Ethernet

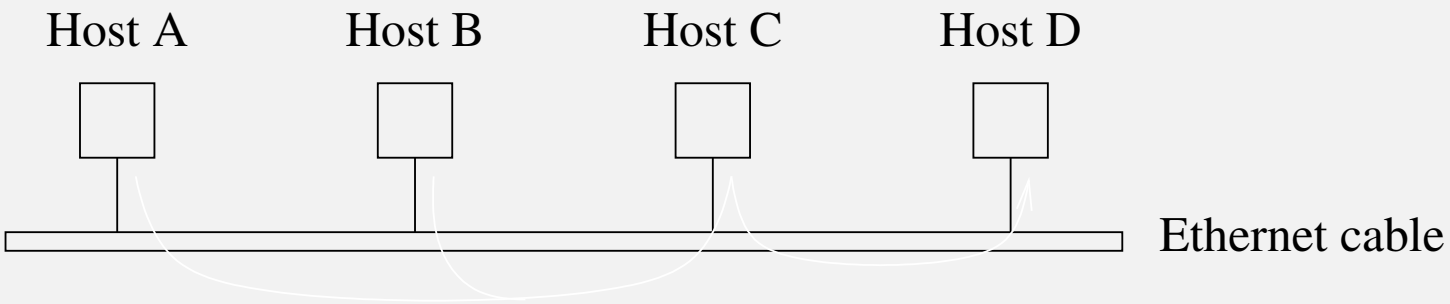
What of the signalling on the wire?

Ethernet uses *carrier sense, multiple access with collision detection* (CSMA/CD)

Networks

Ethernet CSMA/CD

Ethernet is a *multiple access* (shared) medium, meaning that several hosts use the same piece of wire to send data to one another



Original Ethernet

Networks

Ethernet CSMA/CD

If two hosts try to send simultaneously, there will be a *collision*

This is an actual physical condition where the electrical signals from the two hosts get mixed and thus corrupted

So before they send data, a host *listens* to the Ethernet to see if anyone else is using it at the moment: *carrier sense*

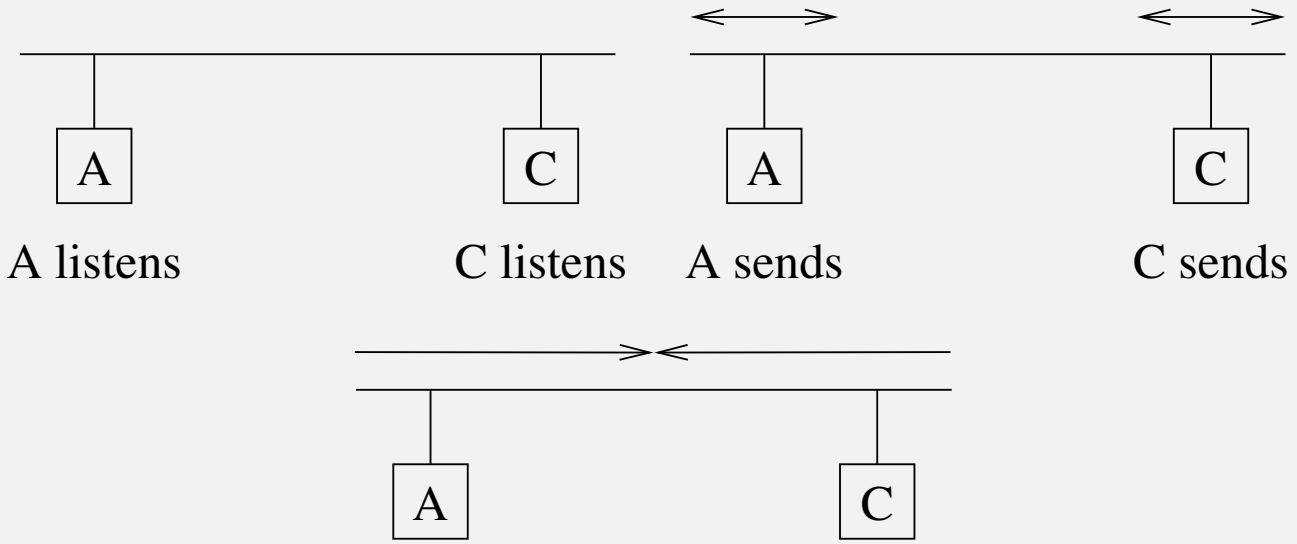
If not, it sends the data

Otherwise it must wait, listening until the carrier is free

Networks

Ethernet CSMA/CD

This still isn't quite enough



Frame collision

So each host **continues to listen while transmitting** to make sure there are no collisions: *collision detection*

Networks

Ethernet CSMA/CD

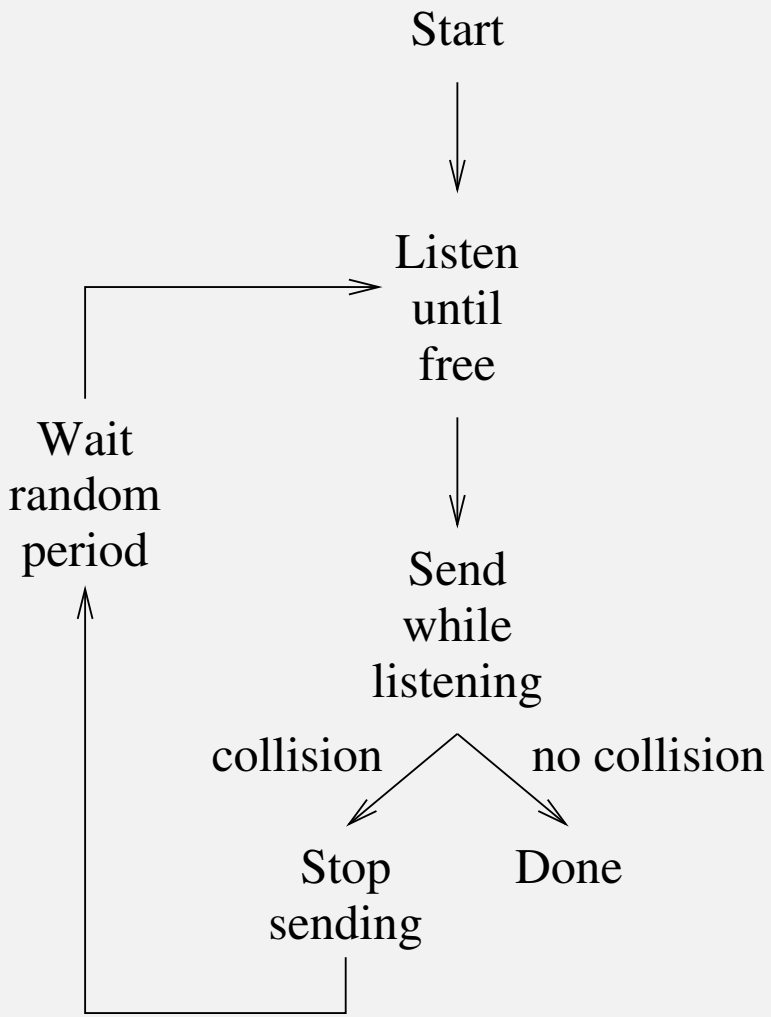
If a collision is detected, each host stops transmitting, waits a (small) **random** period of time and retries with the carrier sense

The random wait means that a further collision is less likely as one host will come in slightly later than the other and see its signal while it is carrier sensing

Detecting collisions on an Ethernet is simple: if the signal you are seeing on the network is not the same as the signal you are putting on the network, that means someone else is transmitting, too

Networks

Ethernet CSMA/CD



CSMA/CD flowchart

Networks

Ethernet CSMA/CD

Exercise Explain why we need to go back to carrier sense after the random pause

Exercise Read further about jamming signals and what to do if the transmission repeatedly fails

Networks

Ethernet CSMA/CD

Collision detection is why there is a minimum frame size

The frames must be on the wire long enough that the hardware can detect a collision

The speed of the signal in the wire is the problem here!

(The speed of a signal in a cable is approx 2/3 c; 100m is 520 cpu cycles of a 1GHz cpu)

And this is made worse with later faster Ethernets

Exercise Find out how CSMA/CD differs from Aloha

Networks

Physical Ethernet

There have been many Ethernet physical layers

Standard	cable	max len	rate
10Base5	Thick coax	500m	10Mb/s
10Base2	Thin coax	200m	10Mb/s
10BaseT	Twisted pair	100m	10Mb/s
10BaseF	Fibre optic	2000m	10Mb/s

Base means *baseband*, namely using a single chunk of frequencies from 0 (the base) up to a single cut-off point

Networks

Physical Ethernet

And these evolved (just a selection here):

Standard	cable	max len	rate
100BaseT4	Twisted pair	100m	100Mb/s
100BaseT	Twisted pair	100m	100Mb/s
100BaseF	Fibre optic	2000m	100Mb/s
1000BaseT	Twisted pair	100m	1Gb/s
2.5GBaseT	Twisted pair	100m	2.5Gb/s
5GBaseT	Twisted pair	100m	5Gb/s
10GBaseT	Twisted pair	100m	10Gb/s

Networks

Physical Ethernet

The cables used in these PHYs change over time. Unshielded Twisted Pair (UTP) comes in various qualities:

- Category 1: No performance criteria
- Category 2: Rated to 1 MHz (used for telephone wiring)
- Category 3: Rated to 16 MHz (used for Ethernet 10BaseT)
- Category 4: Rated to 20 MHz (used for Token-Ring, 10BaseT)
- Category 5/5e: Rated to 100 MHz (used for 1000BaseT, 100BaseT, 10BaseT)

Category 5 has been replaced by Category 5e which has slightly better construction specifications

Networks

Physical Ethernet

All the twisted pair cables are bundles of 4 pairs of wires with an RJ45 plug on the end

Then we have shielded cables, where each pair has a metal foil wrapper:

- Category 6: Rated to 250 MHz
- Category 6a: Rated to 500 MHz
- Category 8.1: Rated to 2000 MHz
- Category 8.2: Rated to 2000 MHz, special end plugs

Plus extra rules on how the plugs on the end are joined on

Networks

Physical Ethernet

You will see “Category 7” cable being sold

It is not standardised, and does not use the usual RJ45 plugs

Even worse, you will see it being sold with RJ45 plugs on, to be compatible with most current consumer networks. This actually reduces its performance to something like Cat6, but at an increased cost

Currently (2023) the best cable to buy is Cat6a as it supports any speed your home network is likely to have and is fairly cheap

Networks

Physical Ethernet

Amusingly, you find reviews of Cat 8 cables on Amazon along the lines of “I installed Cat 8 instead of WiFi and now my home network is super-fast”

They forget using *any* kind of wired instead of WiFi is likely to be faster, less latency and more stable than WiFi

And they would very probably get the same benefit from the much cheaper Cat 5e or 6a

A connection cannot be faster than the slowest component: the device interface, the cable and the switch connecting them

Currently very few home users will have anything faster than 1 Gb interfaces and switches

Networks

Physical Ethernet

Cat 5e and Cat 6/6a is what you will find most widely used today

Cat 6a is roughly the same price as Cat 5e and gives some future-proofing

In particular, Cat 5e is at the edge of supporting 1Gb and bad installs can easily cause problems, dropping the speed to 100Mb. Cat6 has more “headroom”

Networks

Physical Ethernet

There is no testing body to ensure the Category standards are met, so anyone can call any cable anything they like

Reports say that 80% of Cat 6 (and higher) cables (even expensive ones) on sale do not meet the relevant standard; many even fail the Cat 5e test

Networks

Physical Ethernet

The NBASE-T Alliance claims “an estimated 70 billion meters of cabling, which is more than 10 trips to Pluto” has been installed

So people are trying hard to make new Ethernet standards that don’t require ripping out the old cabling and installing new

Thus we have intermediate curiosities like 2.5GBaseT and 5GBaseT (standards developed *after* 10GBaseT), that run on lower-spec cables

The higher speeds and more expensive cabling is usually found only in specialist installations like data centres, HPC and Internet exchanges

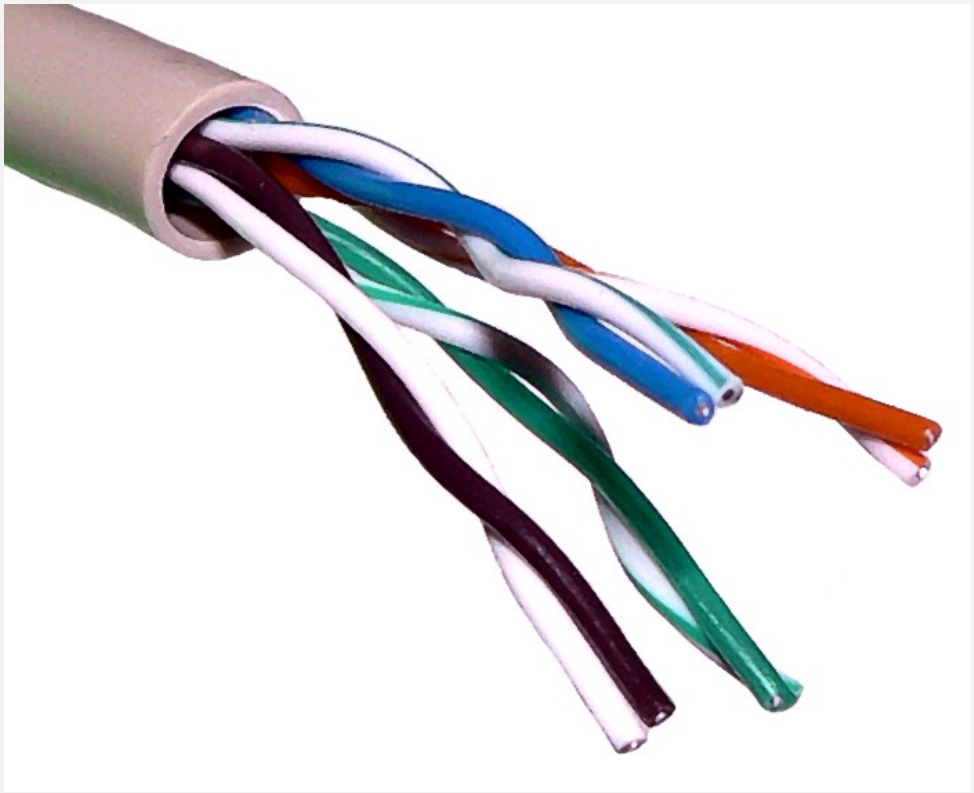
Networks

10Base5 Transceivers

By Robert.Harker at English Wikipedia, CC BY-SA 2.5, `https://commons.wikimedia.org/w/index.php?curid=9891521`

Networks

Ethernet



UTP cable (Wikipedia)

Networks

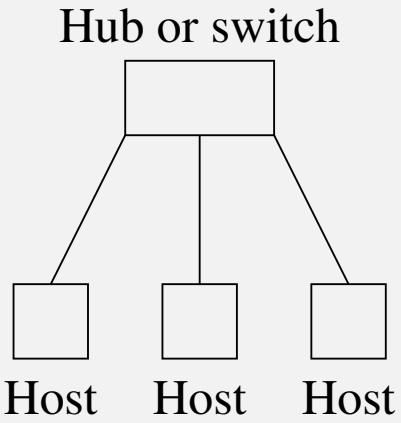
Physical Ethernet

Screened Shielded Cat 6a (Kenable)

Networks

Ethernet

Twisted pair differs from coaxial Ethernet in that it uses *hubs* or (these days) *switches* to connect multiple hosts together



Hosts connected using a hub or switch

Networks

Ethernet

Hubs were simple electrical repeaters. An incoming signal is sent out on all outputs

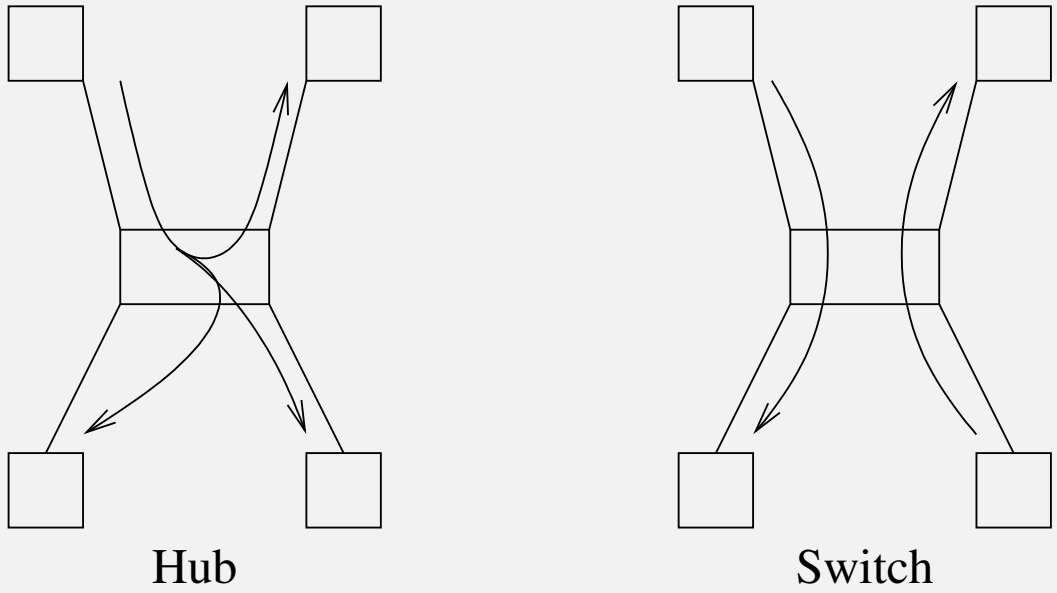
There is a single *collision domain* as all hosts see all signals: any pair of signals between any hosts will collide

The available bandwidth is shared amongst all the hosts

Networks

Ethernet

A switch understands the link layer and can track where a destination host is. It only sends the signal out on the single output that has the destination host



Hub vs Switch

Networks

Ethernet

This requires the switch to read and understand the MAC addresses in the frames and to track the socket where each host is plugged in

This is extra complexity in the switch hardware, but reduces the number of possible collisions, increasing throughput

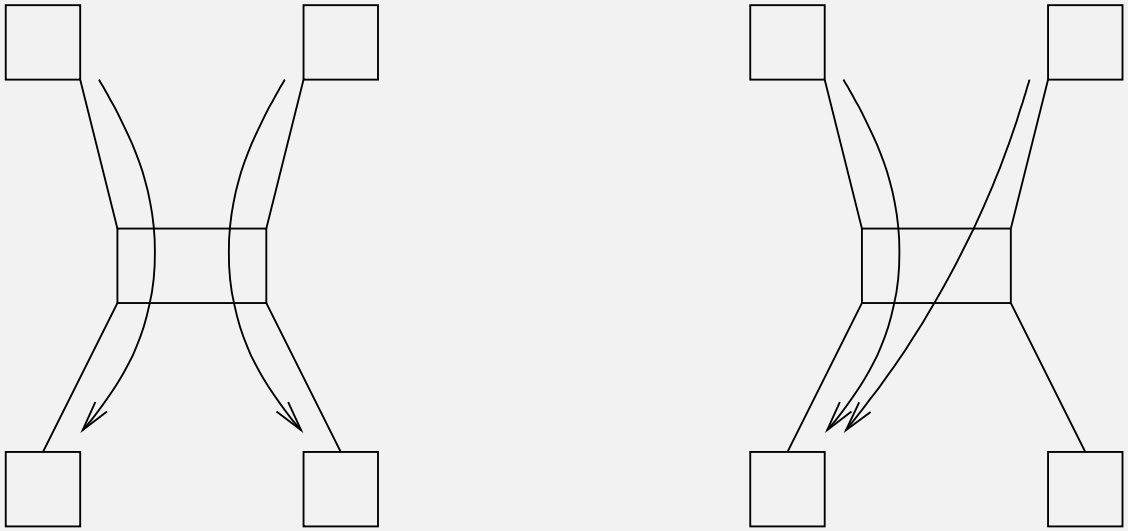
Each output cable is now a separate collision domain

The full bandwidth is available on *each* output, simultaneously

Collisions only if two hosts send to the same destination simultaneously

Networks

Ethernet



Collisions in switches

Networks

Ethernet

If an output is busy, rather than have a collision, a switch may choose to *store and forward* a packet later when that output is free

Now there can be no collisions and we might think we can do away with CSMA/CD

But buffers in the switch can fill up and then packets would have to be dropped by the switch

Instead the switch can send a jamming signal on an input to get it to back off and resend later: thus still using CSMA/CD

Networks

Ethernet

Some switches can *cut through*, sending the start of the packet onwards before the tail has arrived: more complex, but less latency through the switch

Switches can run *full duplex*, with independent inward and outward traffic to each host

This gives *twice* the total bandwidth of previously

No collisions are possible between opposing traffic as inward and outward traffic runs over different twisted pairs (below 1Gb)

Networks

Ethernet

Ethernet is moving faster: 10Mb/s to 1Gb/s and more, all using the same basic CSMA/CD protocol, but using differing electrical signalling

Ethernet cards can autonegotiate to select optimum speed

But it's not just a case of increasing the frequency of the signal, there are other complications to get around the electrical limitations of the cables (discussed later, if we have time)

Networks

Ethernet

Ethernet with speeds above 100Gb/s are called *Terabit Ethernet*

200Gb/s and 400Gb/s Ethernet are available, while 800Gb/s and 1.6Tb/s are under development

Mostly optical fibre rather than copper twisted pair, but some support for very short (e.g., 2m) copper connections

Not likely to be seen in the home for many years!

Networks

Ethernet

Addendum: October 2023

Some ISPs have just announced they will sell $>$ 1Gb/s FTTH products, namely 1.6Gb/s (EE) and 2.2Gb/s (Vodafone in 2024), so providing a “True Gigabit” to the home

To take advantage of this your home network will need a 2.5Gb/s switch: these are available at a moderate price premium over 1Gb/s switches

Your PC or laptop will need a 2.5Gb/s port: you can get USB-C to 2.5Gb/s adaptors reasonably cheaply

Cat 6a will be fine (Cat 5e *should* work, too)

Wireless

The next physical medium we look at is wireless

Wireless networks have been around for a long time: for example cellular telephone systems

Everything wireless is overseen by national and international bodies: we can't have a free-for-all in a wide area shared resource

One wireless system can affect another hundreds or thousands of miles away: there must be some sort of cooperation

So some wireless systems are only allowed with very low power, e.g., Wi-Fi

Wireless

Europe has the *European Telecommunication Standards Institute* (ETSI)

USA has the *Federal Communication Commission* (FCC)

Collaborating with the International Telecommunication Union (ITU)

Such bodies manage the radio spectrum, allocating various frequencies to various purposes, ensuring minimal interference between the competing concerns for parts of the spectrum

Wi-Fi

The IEEE 802.11 group of standards deal with “wireless Ethernet”, more commonly known as *Wi-Fi*

In principle, it is an analogue of CSMA/CD over wireless, but with some extra problems unique to wireless

For example, the shared medium is now all around, not just within a wire

So signals from *multiple* networks can interfere; not just the hosts *within* one network

Wi-Fi

Wireless networks generally have fairly high error rates due to interference from electrically noisy environments, signal reflections, other wireless networks, etc.

So the bandwidth achievable is dependent on the circumstances of the environment

Conversely, wireless networks generate interference themselves which must be controlled so not to be annoying to other people

Wireless Problems

In 802.11, the allowed power of transmission is generally kept quite low by the standards bodies to minimise interference

E.g., a typical laptop will transmit at about 32mW; it can read a signal as low as 0.00000001mW

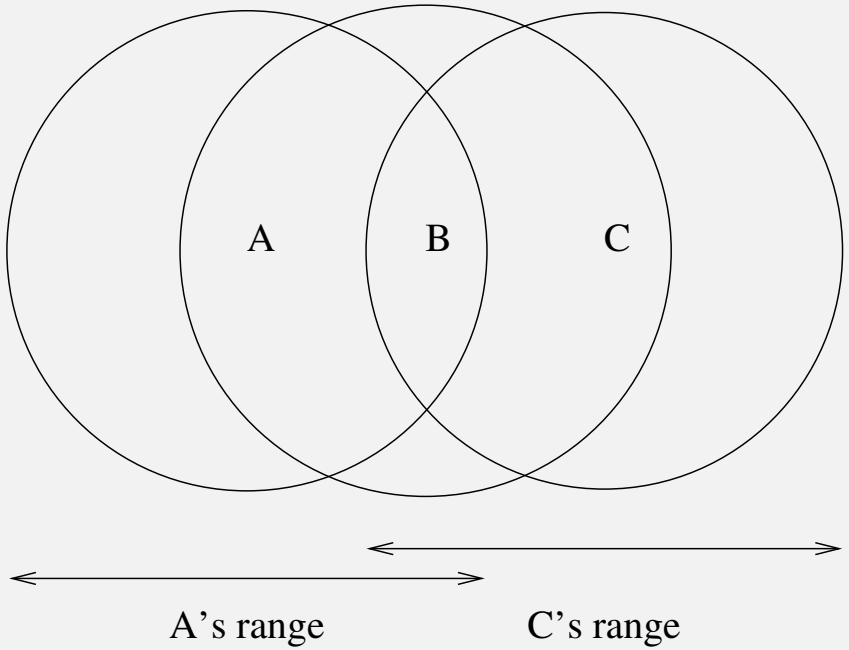
(A digital TV mast might transmit at 100kW)

Thus the range achievable by Wi-Fi is often quite limited — deliberately

But a limited range can cause complications

Wireless Problems

When we have wireless, we get the *hidden host* problem:



Hidden host

Hosts A can B can "see" each other; B and C can see each other, but A cannot see C, so A cannot tell if its packets to B are colliding with C's to B

Wireless Problems

In reality, the ranges will not be circular, but something rather complicated dictated by the environment

But the limited ranges mean that CSMA/CD will not work for wireless

CSMA/CD relies on everyone's signals being visible to everybody for CD to work

Wireless Problems

Next difference: as packets are broadcast, wireless networks are intrinsically insecure, so extra effort must be taken over security and authentication

War Driving is driving with your laptop around the neighbourhood until you find an unsecured wireless signal: then you have free access to the Internet!

This is illegal in the UK and elsewhere

These days, many fewer people forget to secure their networks than was common in the early days of Wi-Fi

Only use a Wi-Fi network if you have permission to do so

Wireless 802.11

There are several parts to the 802.11 standard, including 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, 802.11ax and more

You may now see them under the brandings:

Wi-Fi 1	802.11
Wi-Fi 2	802.11b
Wi-Fi 3	802.11g, 802.11a
Wi-Fi 4	802.11n
Wi-Fi 5	802.11ac
Wi-Fi 6, 6E	802.11ax

Wi-Fi 7 (802.11be) is due in 2024

Wireless 802.11

Other parts of 802.11, like 11c, 11d, 11e, 11f, 11h, 11i deal with things like power management, quality of service, security and authentication and so on

Wireless 802.11

The original standard specified signalling rates of up to 2Mb/s

Up to 100m (300 feet) indoors and 300m (1000 feet) outdoors

There was an infra-red mode as well as a radio mode, but this was not widely implemented

802.11b extended this to rates of 5.5Mb/s and 11Mb/s

Wireless 802.11

They use the unlicensed 2.4GHz waveband

That means you do not need to get a licence to use that frequency at low power

This was a frequency that was otherwise unusable commercially and is subject to interference from microwave ovens and other things

And the frequency fell within the capabilities of low-power chips that were buildable at the time

Wireless 802.11

		Freqs GHz	Signalling rate
WiFi 1	11	2.4	2Mb/s
WiFi 2	11b	2.4	11Mb/s
WiFi 3	11g	2.4	54Mb/s
WiFi 3	11a	5	54Mb/s
WiFi 4	11n	2.4,5	600Mb/s
WiFi 5	11ac	5	6.9Gb/s
WiFi 6	11ax	2.4,5	9.6Gb/s
WiFi 6E	11ax	6	9.6Gb/s
(WiFi 7	11be	2.4,5,6	46Gb/s)

Wireless 802.11

Improvements are achieved through more sophisticated signal encodings and using more wireless channels simultaneously

Each will fall back to previous standards to maintain compatability with earlier devices

For example, a 5GHz signal has problems going through walls, so 11a can fall back to 11b if you move to the next room

Exercise Look these up. Particularly the use of multiple aerials for *beamforming* and *spacial multiplexing*

Wireless 802.11

802.11 hardware is branded “Wi-Fi”, which is actually a certificate of interopability given to manufacturers whose equipment demonstrably works with other manufacturers’

Administered by the Wi-Fi Alliance, a consortium of interested companies

Wireless 802.11

The bits in 802.11 are not simply transmitted directly: there is a lot of environmental interference to overcome

Instead the signal is spread over many frequencies using variety of techniques collectively called *spread spectrum*

Exercise Read about *Direct Sequence Spread Spectrum* (DSSS)

Exercise And read about film actress Hedy Lamarr

Wireless 802.11

For Wi-Fi, the allocated frequency band (2.4–2.5GHz) is split into 14 overlapping 22MHz channels each centred on specified frequencies

The number of channels available depends on the country

- Most of Europe: 13
- North America: 11
- Japan: 14

Wireless 802.11

Channel	GHz
1	2.412
2	2.417
3	2.422
4	2.427
5	2.432
6	2.437
7	2.442
8	2.447
9	2.452
10	2.457
11	2.462
12	2.467
13	2.472
14	2.484

Wireless 802.11

Those are central frequencies, with each channel being 22MHz wide

So, for example, channel 1 is 2.401–2.423GHz and channel 2 is 2.406–2.428GHz

The channels are 5MHz apart, so neighbouring channels overlap (as they are 22MHz wide) and interfere. Therefore you need to take care which channels you use

There are recommendations on using channels

Wireless 802.11

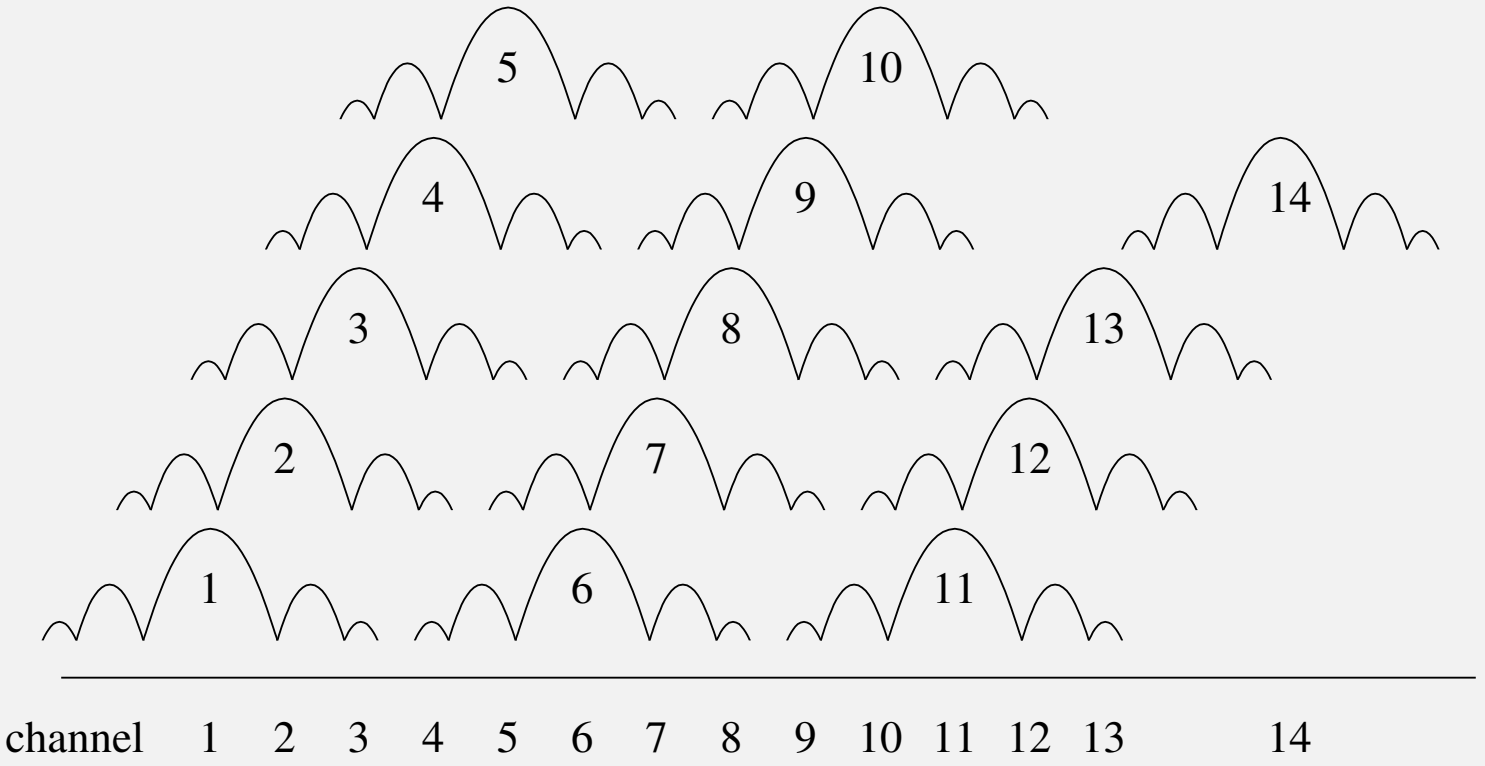
- Separate channels by at least 2 (e.g., use 1 and 4) to reduce interference
- Separate by 4 (e.g., use 1 and 6) to have no interference at all
- This means we can have three non-interfering co-located networks on channels 1, 6 and 11

Wireless 802.11

Separating networks physically gives more leeway:

- Separate by 1 (e.g., use 1 and 3) if the networks are more than 40m apart
- Adjacent channels (e.g., use 1 and 2) are OK over 100m
- Channels can be reused when the networks are sufficiently separated

Wireless 802.11



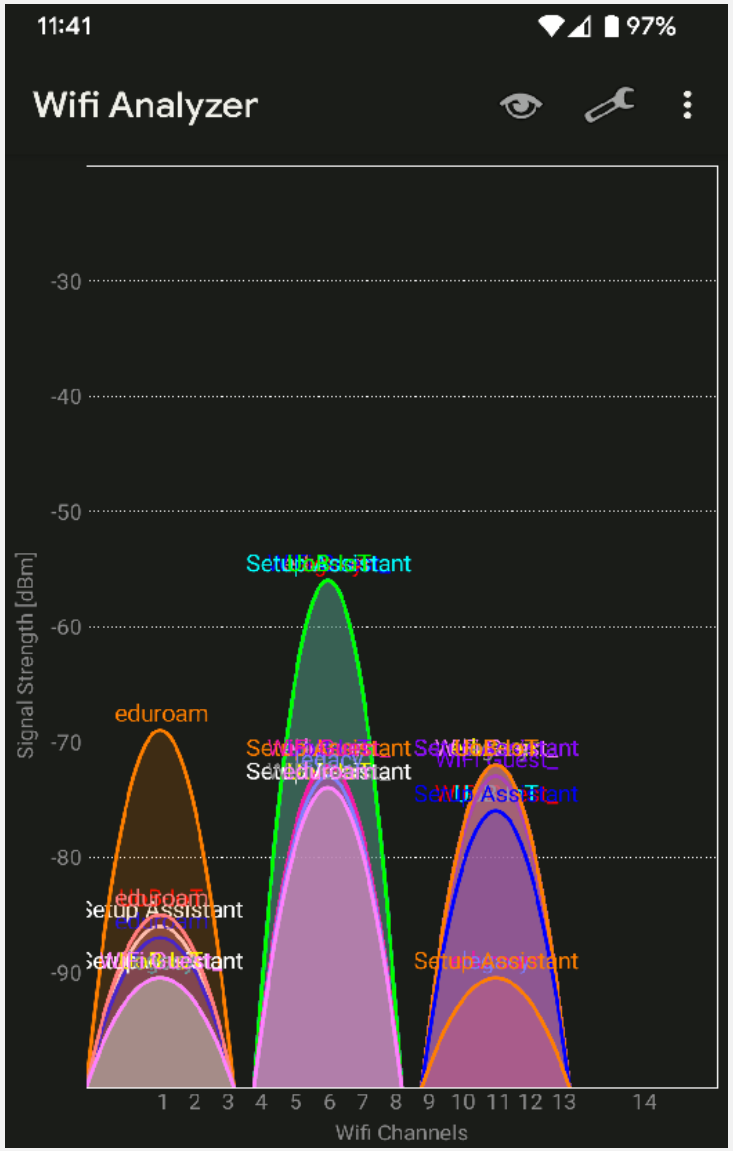
Overlapping WiFi channels at 2.4GHz

Wireless 802.11

More subtle channel allocations allow a little overlap (e.g., using channels 1 and 3) that have a little interference, but a greater overall aggregate bandwidth

Exercise Mobile phones have wireless apps that display the wireless environment. Walk around and see what it is like

Wireless 802.11



WiFi Analyzer app

CSMA/CA

The Hidden Host Problem means 802.11 can't use CSMA/CD, like wired Ethernet

Instead, it uses carrier sense, multiple access, *collision avoidance* (CSMA/CA)

This is similar to CSMA/CD, but with a big difference

- Carrier sense: to deal with the common case of non-hidden hosts, first listen for a signal
- If free, send a packet
- If busy, wait until the end of the transmission and then enter a *contention period*: wait a random period
- Go back to carrier sense

CSMA/CA

Waiting for the contention period is the collision avoidance

A random wait mean that several hosts wanting to transmit are unlikely to all start transmitting simultaneously

We are trying to avoid a collision in advance rather than detect one after the fact: we know that signal detection is problematic in Wi-Fi

But collision avoidance does not *guarantee* no collisions, particularly with hidden hosts, so we need more

CSMA/CA

Thus, on successful receipt of a packet, a destination host will broadcast an acknowledgement (ACK) packet

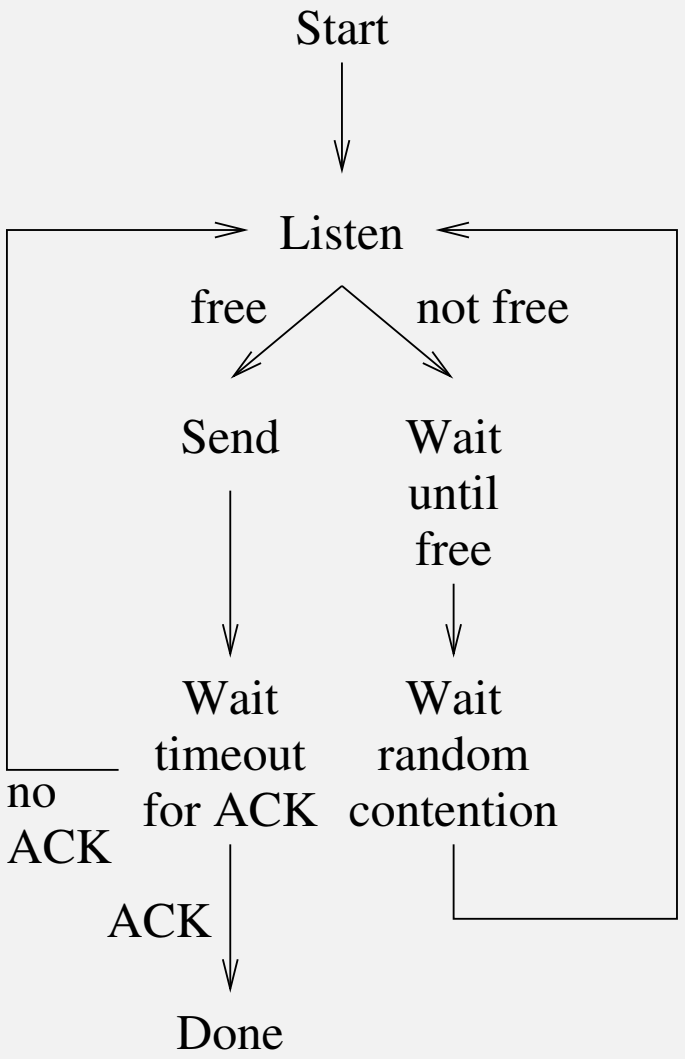
This is just a packet to inform the sender that everything worked well and there was, in fact, no collision or other loss

If the sender never gets the ACK, it will resend, starting from the CS again

This ACK is important, as measurements have found loss rates on the order of 30%

Note the ACK is also visible to everyone in range of the destination, giving extra indication to others when a transmission has finished

CSMA/CA



CSMA/CA flowchart

CSMA/CA

Exercise Compare and contrast the CSMA/CA flowchart with the CSMA/CD flowchart

CSMA/CA

Why use collision avoidance rather than collision detection?

Clearly, the contention period means more latency in transmission

We do it because with wireless, collisions can be very hard to detect

With Ethernet, detecting another host's signal on a wire is easy as the power of its signal is roughly the same as yours

CSMA/CA

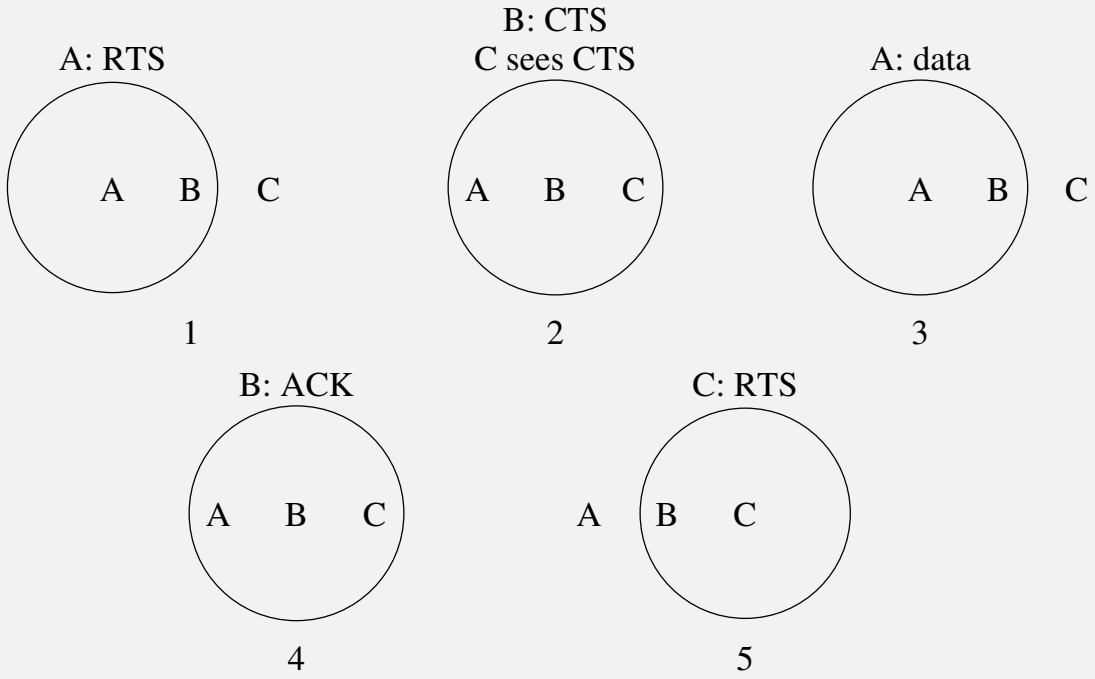
In contrast, detecting another host's radio signal can be very difficult as it can be a tiny fraction of the power of yours, and your signal will drown out the colliding signal and make it undetectable

Recall the wide range of power that Wi-Fi signals encompass: another destination might be transmitting quite powerfully, but its signal can be very small by the time it reaches you

Wi-Fi

To help further with the visibility problem, there is optional *RTS/CTS handshaking*, which can improve performance in certain circumstances

RTS/CTS



RTS/CTS handshaking

1. Before sending a data packet the source A can send a *request to send* (RTS) packet to B; 2. If the destination B is happy (it is not already receiving from another host that A cannot see) it responds with a *clear to send* (CTS) packet; 2. Every other host within the range of the destination will see the

RTS/CTS

This means there is even more latency overhead before data starts to be transmitted, so RTS/CTS can be switched off or on as required:

RTS/CTS always on: good for large or busy networks

RTS/CTS never on: good for small or lightly loaded networks where every host can see all other hosts

RTS/CTS for large packets only: a compromise that reduces the relatively large overhead for small packets

Wireless Rates

Although 802.11b is nominally 11Mb/s and 802.11g is nominally 54Mb/s remember these are the signalling rates, not the data rates

The signalling rate is the raw bit rate over the airwaves: a lot of that is consumed in overheads

Realistically, 802.11b gives about 3 to 4Mb/s and 802.11g about 20Mb/s

Some of the later 802.11 standard improve speeds by reducing overheads (as well as using better encodings)

802.11

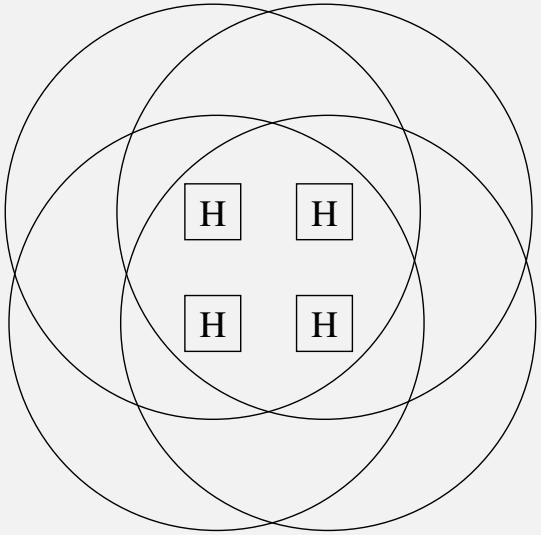
Exercise 802.11ac (branded “Wi-Fi 5”) is common and 11ax (“Wi-Fi 6”) hardware becoming more common. Read up on what they promise and what they deliver

Wireless Networks

While the use of access points is common, this is not the only way to set up a wireless network

802.11 can be arranged in point-to-point networks called *Ad-Hoc* or *Independent Basic Service Set* (IBSS)

Wireless Networks



Point−to−point connections
IBSS

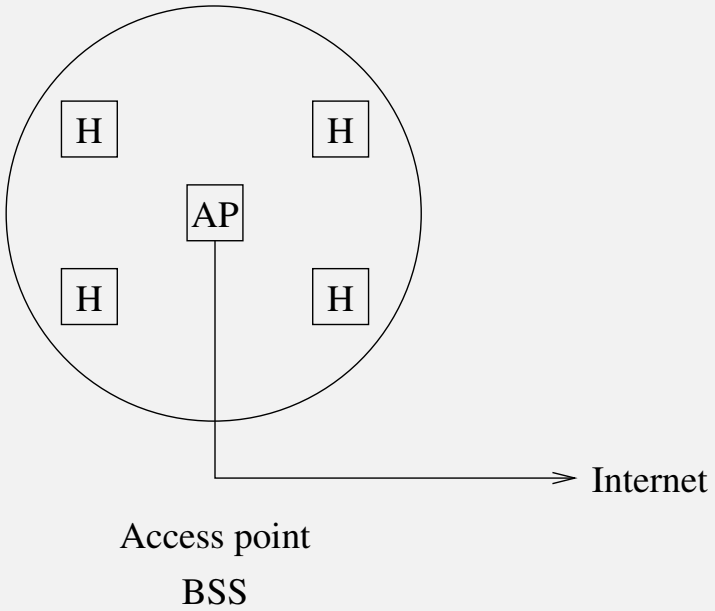
Ad-Hoc network

Each host communicates directly with each other without an access point

Clearly all hosts need to be sufficiently close to each other

Wireless Networks

But the usual Wi-Fi network is a *Infrastructure* or *Basic Service Set* (BSS), where a central hub (*access point*) relays traffic between hosts



Usual access point setup

Wireless Networks

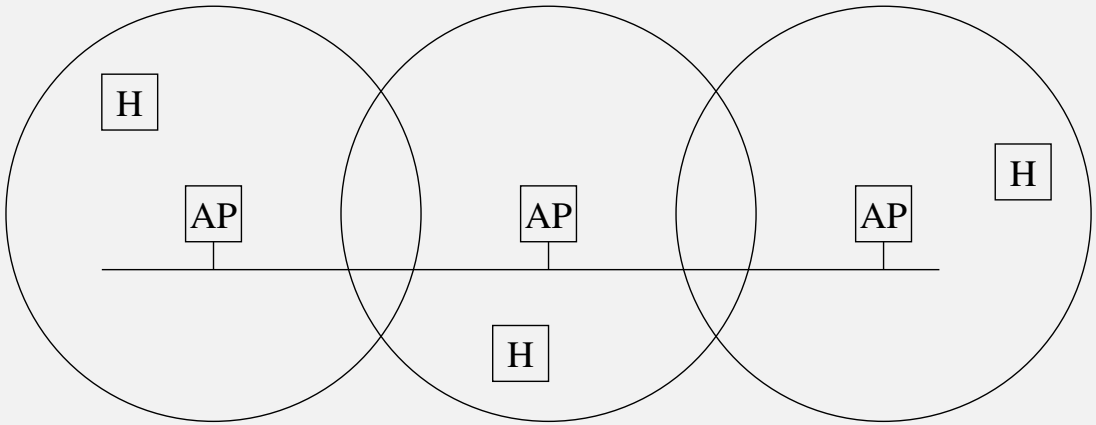
This is more expensive to set up (as you have to buy an AP), but covers a larger area

And is easier to manage by non-technical users

Also the AP can connect into a wired network and so the rest of the Internet

Wireless Networks

Extended Service Set (ESS) connects several APs by a wired network



ESS

Extended network

This allows hosts to roam and things can be configured to handoff automatically between APs if the required authentication infrastructure is set up in the APs

An ESS can cover an area as large as you like

Wireless Networks

Exercise Read about *Wi-Fi Direct*, another peer-to-peer wireless connection between hosts, often used as a device setup mechanism. Compare with Ad-Hoc mode

Exercise Read about Mesh networks

Wireless Security

While we are talking about authentication. . .

Wireless packets are readable by anybody in the neighbourhood, so security is essential in a wireless network

We have two issues:

- is this machine allowed to connect to this network: authentication
- ensure data in transit is kept secret: privacy

On Ethernet, being plugged into the network is the “authentication”, while the physical security of the network is the “privacy”

But only private from people not on the network!

Wireless Security

Original 802.11 employed the *Wired Equivalent Privacy* (WEP) encryption scheme

Both ends of a communication share a secret key that is used to encrypt the traffic between them

WEP is now easily breakable: after collecting a modest amount of traffic the system can be broken

As can its successor, *Wi-Fi Protected Access* (WPA)

Wireless Security

Currently we use mostly WPA2, (IEEE 802.11i-2004)

Exercise Read about the break of the WPA2 protocol (Oct 2017)

Exercise Read about the new WPA3

Wireless Security

Two major ways to set up authentication are

- WPA-Personal: also called WPA-PSK (pre-shared key), where an access point has a secret key, and a host authenticates directly with the AP using the secret key
- WPA-Enterprise (802.11X): requires a separate authentication server (typically a RADIUS server) that the AP will contact. Much more fiddly to manage, but allows roaming across an ESS. Also roaming across institutions using hierarchical RADIUS servers

We usually find BSS using WPA-PSK and ESS using WPA-Enterprise, but either can use either

Wireless Security

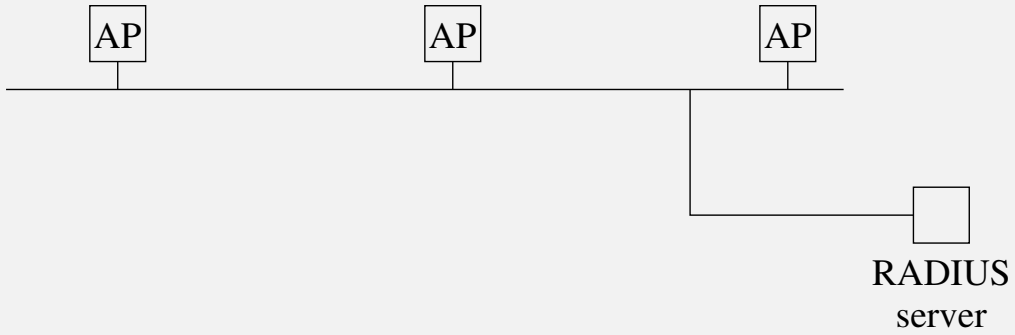
For WPA-PSK the secret key is usually derived from a password for ease of use

The password is communicated off-line, e.g., written down somewhere

Everybody on the network shares the same key/password; authentication is done in the AP

WPA-Enterprise is more complex

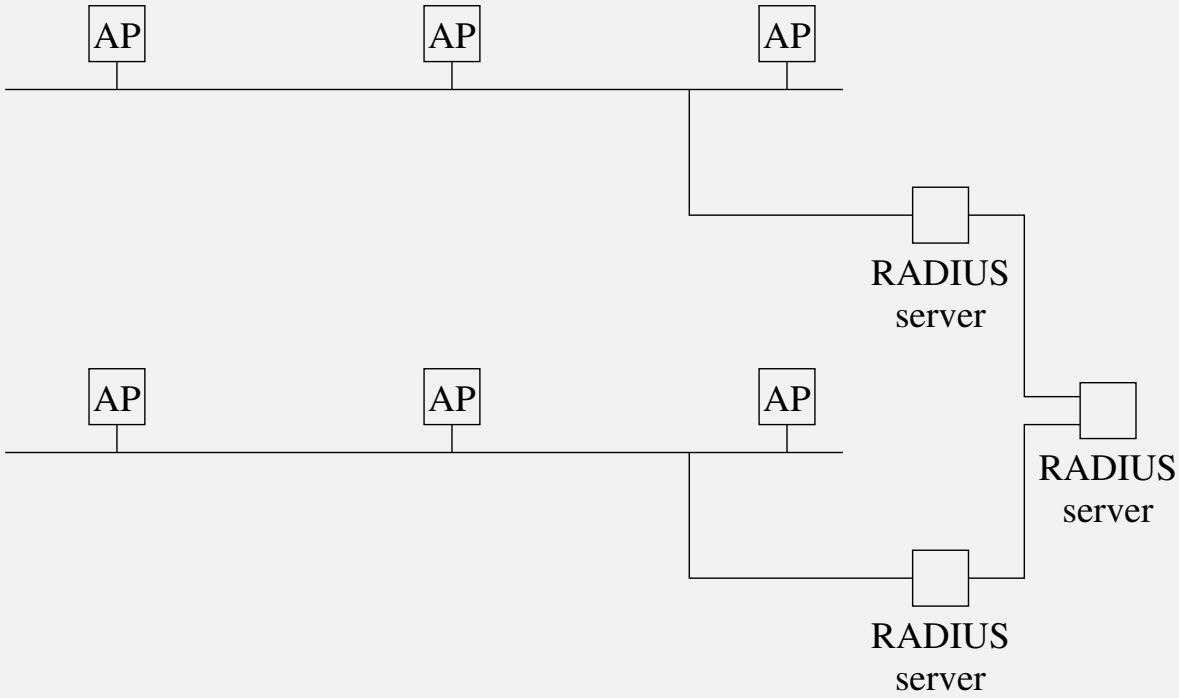
Wireless Security



RADIUS authentication

Access points do not authenticate, but ask a RADIUS server

Wireless Security



Multi-institution

Wireless Security

For WPA-Enterprise each user has their own key/password

Authentication is done in the RADIUS server on both the username and the password

Wireless Security

Exercise Read about how Eduroam uses WPA-Enterprise

Exercise Read about RADIUS: *Remote Authentication Dial In User Service*

Wireless Security

Some APs have *Wi-Fi Protected Setup* (WPS), a simplified way of setting up WPA/WPA2 security

Designed for those people who find typing in a password too challenging

It is seriously broken and should be disabled on your AP

Exercise A common system we see on public Wi-Fi is a redirect to a login web page: sometimes called a *captive portal*. What kind of security (privacy and authentication) does this provide? Note this is *not* WPA-Enterprise

Wireless 802.11

The frame layout for Wi-Fi is the same as Ethernet

In particular it has the same format MAC addresses, e.g., `00:04:ed:f1:ef:8a`

This allows the transparent mixing of Wi-Fi and Ethernet in a single network

An AP can pass on a Wi-Fi frame unchanged to an Ethernet; and vice versa

Exercise What implication does this have for Ethernet collision domains?

PHY Sublayers

This is a good argument for sub-dividing the physical layer!

Exercise For hardware hackers: read about the IEEE layers:

- *Physical Medium Attachment* (PMA) for things like frames
- *Physical Coding Sublayer* (PCS) for things like 4B/5B
- *Physical Medium Dependent* (PMD) for the hardware

But it does mean we don't have to discuss Wi-Fi any further!

Other Wireless

Many other wireless networks exist, from local to wide-area

Other Wireless

Bluetooth gives short range, point-to-point communication

Point-to-point: just two hosts in the network

A range of 10m

Also uses 2.4GHz band

Not really designed to run IP, but can by layering a suitable protocol (see PPP, later)

Bluetooth Low Energy (BLE), is a non-backwards-compatible evolution designed to reduce power consumption

Other Wireless

Exercise Read about *Adaptive Network Topology* (ANT and ANT+) for short range low power wireless, similar to BLE, but for use with fitness (and other) sensors (by Garmin)

Exercise Read about *Zigbee* for short range low data rate, low power wireless, for use in home automation and control

Networks

We now move to look at a different environment: longer distance networks, in particular to the home

These have quite different requirements from, say, commodity Ethernet: in particular you don't always get to choose the physical layer. Sometimes you have to make do with whatever hardware is available, e.g., buried in the street

Networks

Analogue

Before digital networks were common, the physical layer of choice was an acoustic modem, using the existing analogue telephone network

This used **MO**dulation and **DEM**odulation to convert bits into acoustic symbols, i.e., sounds

The early Internet (Arpanet) ran over the existing analogue telephone network

Networks

Analogue

Exercise Read about the V series of modem standards

Exercise Read about *amplitude modulation*, *frequency modulation* and *phase modulation* and *Quadrature Amplitude Modulation* (QAM) *constellations*

Networks

Digital

After analogue, public telephone systems started to support purely digital networks

Exercise Read about Integrated Services Digital Network (ISDN)

ISDN was the precursor to ADSL

ADSL

Asymmetric Digital Subscriber Line (ADSL) was an important method of delivery, and drove the first big increase of the Internet in the home

Analogue modems are limited to 56Kb/s, the maximum speed available from a standard analogue telephone line where all frequencies apart from a 3KHz chunk centred on the human voice are filtered out and thrown away

The telephone wire — while only originally specified to be capable of sending voice — is capable of more, ADSL tries to take advantage of this

It uses many blocks of frequencies simultaneously — broadband — that avoid areas of the spectrum that have interference, and to make best use of areas of the spectrum that don't

ADSL

The data rate you get depends on the quality and length of the copper loop connecting you to the telephone exchange: the longer it is the harder it is to get a clean signal down it

And the amount of interference there is along the route of the copper loop

ADSL2+ tops out at 24Mb/s, dropping to 2Mb/s at its longest reach (about 4km, maybe up to 8km if you are really lucky)

ADSL

It is *asymmetric* in that is divides the available bandwidth unequally into (say) 24Mb/s downstream (towards the user) and 2Mb/s upstream (towards the Internet)

Which is what most home users want: a few clicks on a Web link (low bandwidth) resulting in a large page download (high bandwidth)

ADSL

But 24Mb/s is not enough for today's video streaming, multi-occupant houses: we need to go faster and ADSL and updates to ADSL (e.g., VDSL) can't keep up

Exercise ADSL is just one in a series of DSL standards, collectively called xDSL. Read about these

The Last Mile Problem

This is part of *last mile problem*: how to bridge the gap between the local telephone exchange and the final user

Also called the *first mile problem*

Fibre Hybrid

Currently, the most popular solution is to have *Fibre to the street cabinet* (FTTC) and then use a DSL over the existing copper wire (the copper *loop*) to the home

VDSL2 is used on the copper from the cabinet to the home: with an asymmetric up to 80Mb/s downlink, 20Mb/s uplink

(The actual limits are more complex; and made even harder by legal restrictions on advertisements of speeds in marketing)

Fibre Hybrid

VDSL (and VDSL2 etc.), another DSL standard, gives higher data rates than ADSL over short distances

But the rates drop off rapidly with distance, and after about 1.6km its performance drops below that of ADSL

Exercise Read about the various distances, performances and frequencies used by these standards

Fibre Hybrid

With VDSL, the distance you live from the street cabinet governs what speed you actually get

In contrast, ADSL uses the old copper loop all the way from the exchange to the home: this can be many kms

The FTTC “Fibre broadband” hybrid represents the current cost-effective way of getting decent bandwidth to the home

Fibre

Ideally we would each have a high-bandwidth optical fibre to our home

Optical fibre is not subject to electrical interference like copper wires, and can carry huge (terabits is possible) datarates

We would like *Fibre to the building/business* (FTTB) or *Fibre to the premises* (FTTP), where fibre comes to a building (business or multiple occupancy building); or *Fibre to the home* (FTTH) where fibres come to individual houses

Fibre

It will be very expensive to provide everybody with a fibre connection: a lot of digging up the road is needed

The legacy copper telephone network was put into place over many decades

Though progress on laying fibre is continuing and there are plans to decommissioning the copper network at some point in the future

In 2019 the UK government announced that there will be 100% coverage of gigabit “broadband” by fibre or 5G cellular by 2025

Previously it had an “aspiration” to have 100% optical fibre by 2033 — much more realistic

Exercise Find out what the current Government target is

Fibre

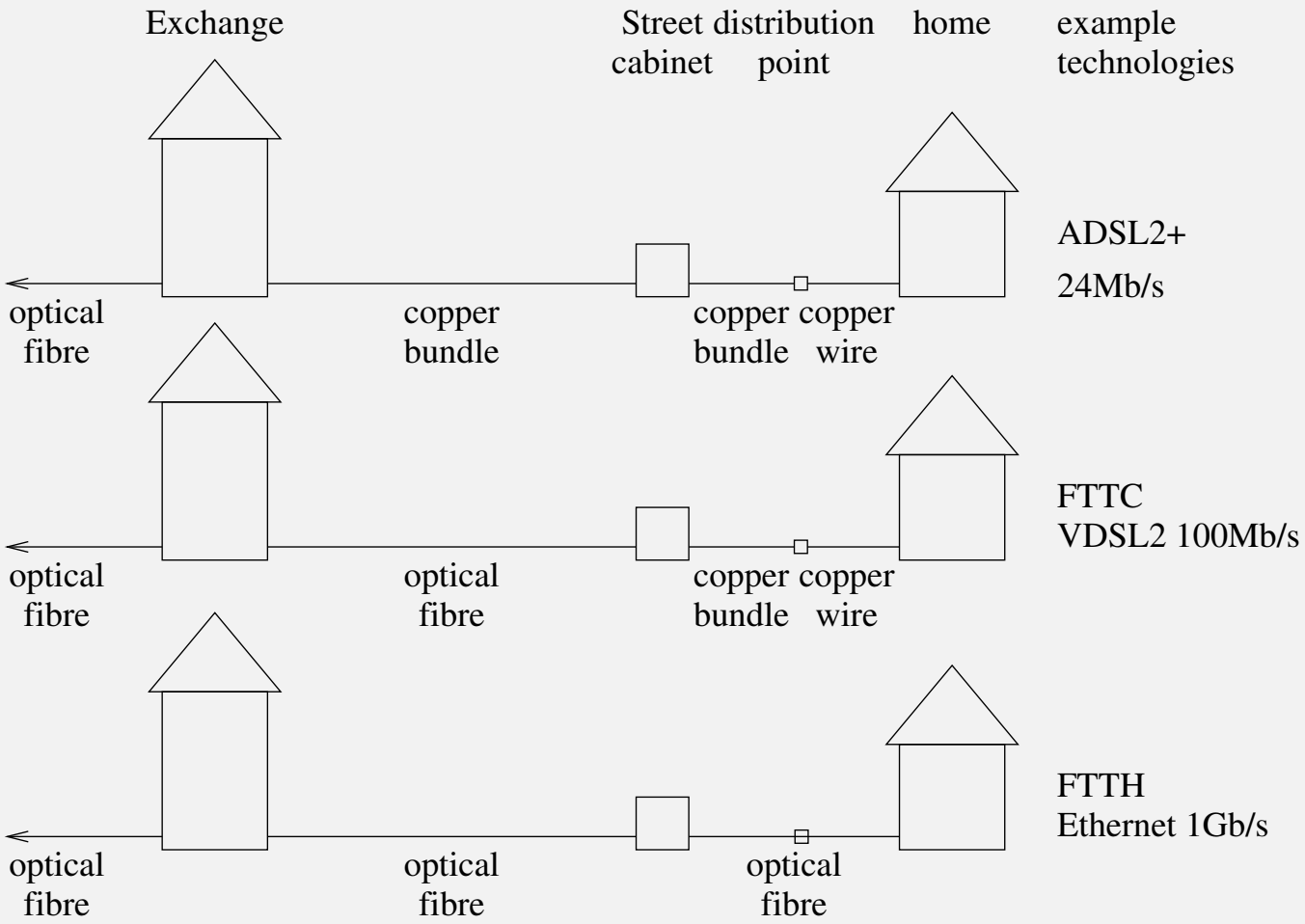
The current big push is to install FTTP and there is a lot of engineering works across the country digging up streets to install it

Big companies like Openreach (BT) and Virgin as well as smaller *altnets*, like Truespeed and City Fibre currently digging up Bath

Offering up to gigabit speeds: usually advertised as 900Mb/s (rules on advertising, again)

With an eye to the future, some altnets are installing hardware that can provide up to 10Gb/s

FTTH/P is being marketed as “full fibre” to distinguish it from FTTC



Current common connections in the last mile

FTTx

A street cabinet is where a bundle of wires or fibres for several streets (say) split into smaller bundles going to the respective streets

A distribution point is a place (another cabinet or underground beneath a manhole or on a telegraph pole) where the bundles split into the individual connections to their destinations

For example, a FTTH might have multiple fibres to a street cabinet; a single shared fibre to the DP where the signal is split optically on to individual fibres to the homes

Exercise Read about (the now defunct) FTTdp that took fibre all the way to the DP

The Last Mile

In the UK we have:

5500 exchanges	ADSL	copper	24Mb
10,000s street cabinets	VDSL FTTC	fibre + copper	80Mb
1,000,000s distribution points	G.fast FTTdp	fibre + copper	300Mb
30,000,000 premises	Ethernet FTTP	fibre	1GB

The Last Mile

In the UK there is a legal requirement: the *universal service obligation* (USO) of a connection of at least 10Mb/s down, 1Mb/s up

BT (or other large provider, like KCOM in Hull) have to provide this at a reasonable cost

It can be wired/fibre or even wireless

End of Analogue

BT wants to turn off all analogue (*Public Switched Telephone Network* (PSTN)) networks by 31 Dec 2025

Meaning the voice network, i.e., telephone

Voice will be replaced by digital (*Voice Over IP* (VOIP)) over copper or fibre to a modem in your home; your phone plugs into the modem

In the long run they want to remove the copper, but this means building fibre (or wireless) everywhere first

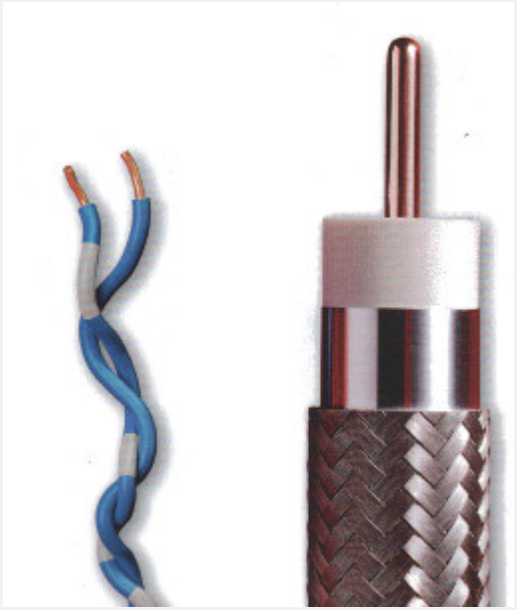
Cable TV

The cable TV system, where available, is another solution to the Last Mile

Newer installations are full fibre, but there is also a lot of another fibre/copper hybrid, with fibre to cabinets and then copper to the home

However, the copper wires used is good(ish) quality coaxial cable that is well screened against interference and crosstalk, and so the data rates it supports are much higher

Cable TV



Telephone wire and coaxial cable

Picture from Virgin Media

Exercise *Read up on Data Over Cable Service Interface Specification* (DOCSIS)

Wireless

Next: cellular networks, as used by mobile phones for the Last Mile

The first important digital system was *Global System for Mobile Communications* (GSM)

Retrospectively named *2G*

(1G was the preceding analogue system)

Rates of 9.6Kb/s to 14.4Kb/s (similar to old analogue wired modems)

High Speed Circuit Switched Data (HSCSD) takes this up to 57.6Kb/s

Wireless

General Packet Radio Service (GPRS), packet based, up to 171.2Kb/s

Uses several GSM channels

Enhanced Data rates for GSM Evolution (EDGE) uses better encodings to get up to 384Kb/s, again using several channels

EDGE used by Third Generation (3G) systems

High-Speed Downlink Packet Access (HSDPA) ups this to 42.2Mb/s

Evolved High-Speed Packet Access (HSPA+) will do 168Mb/s

Wireless

4G is well established

Using *Long Term Evolution* (LTE) with the promise of 300Mb/s

LTE, marketed as “4G”, originally did not meet the proposed 4G standard as it did not satisfy the proposed technical specifications of a 4G system

In particular, a 4G network should support 1Gb/s for a stationary host

The ITU (who say what “4G” is supposed to mean) actually gave in to commerce and retroactively changed the definition of 4G to allow for LTE

Wireless

LTE is data traffic only, and does not have a voice channel

Currently on many LTE systems if you want to make a voice call it has to drop back to 3G (or even 2G)

Some phones and systems support *voice over LTE* (VoLTE) using a suitable digital encoding of sound over the data channel

Exercise Some systems support “Wi-Fi calling”, which is using your Wi-Fi (rather than the cellular network) to connect to the telephone system. Read about this

Wireless

5G is currently being deployed

It uses the available spectrum much more efficiently than 4G, and employs frequencies up to 86GHz (LTE uses up to 6GHz)

Projections indicate users connected to a base-station will share 20Gb/s download and 10Gb/s upload rates

And base-stations will support “millions” of devices per square mile (enabling the *Internet of Things*)

A device will be able to connect even if it is moving at 500km/h (e.g., in a plane); latencies will be 1ms, compared to the current 20ms on LTE

Wireless

Current sticking points over the adoption of 5G are:

- lack of support in “old” mobile phones
- phone 5G chipsets currently suck a lot of power
- the need to build a lot more base stations (using higher radio frequencies means the range of a cell is smaller)
- or upgrading old ones and re-purposing existing frequencies used by 3G

Wireless

6G? A new “G” appears roughly every 10 years, so maybe 2030, but this is uncertain as 5G has significant improvements planned. Maybe a standard will be published in 2025

With targets of 100Gb/s to 1Tb/s using 100GHz to 1THz (terahertz) frequencies

THz is between microwaves and infrared, not ionizing; current mobile is MHz and GHz. The tech to generate THz waves is still very new

(The base-stations will need really good onward connectivity!)

Wireless

2G and 3G signals are due to be phased out by 2033 so their frequencies can be reused by 4G and 5G

Probably 3G will go first, as 2G is widely used in things like Smart Meters, and as a low-power, long-range fallback, particularly in rural areas

Many companies are looking at dates like 2025 for 3G removal

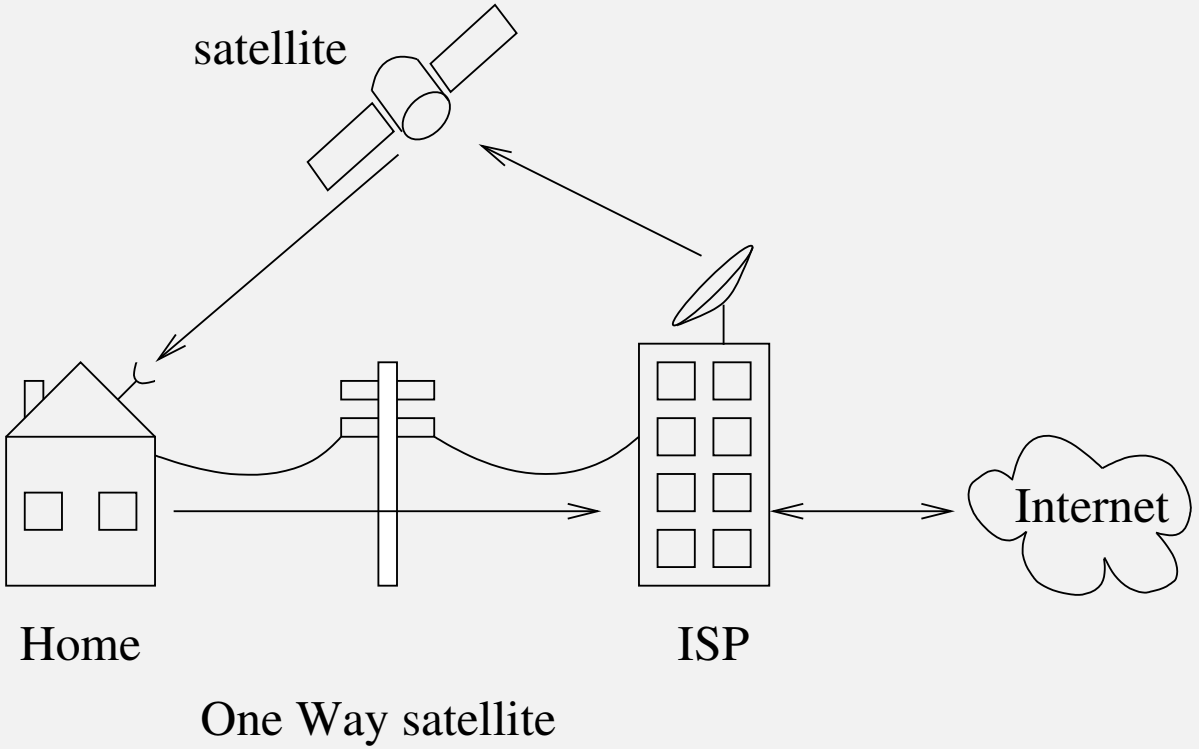
Exercise Read about how this conflicts with the limited support for VoLTE in 4G

Wireless

Satellite networks can be used outside of well-connected urban areas for the Last Mile

There are two main variants

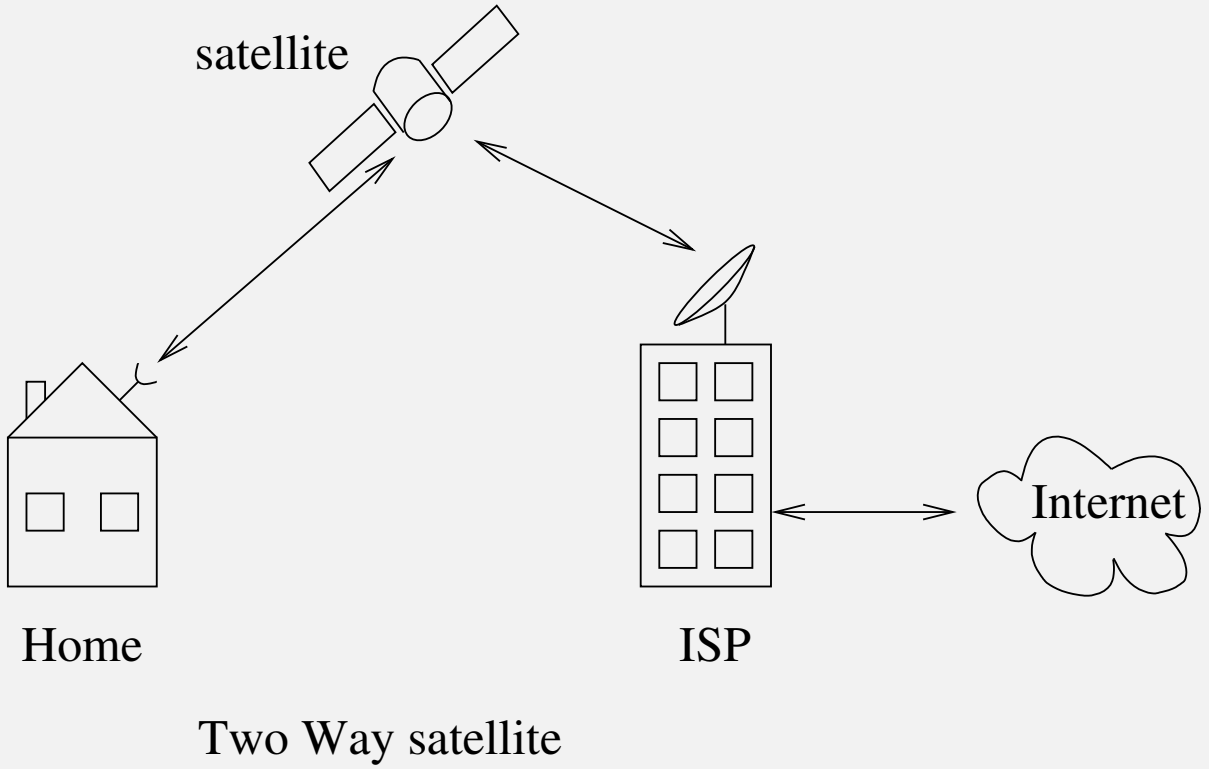
Wireless



One way satellite

One way satellite: this employs the usual asymmetry. Data away from the home travels by telephone wire; data towards the home travels through a satellite connection

Wireless



Two way satellite

Two way satellite: satellite connections both ways. More expensive in equipment in the home, but not reliant on a telephone network

Wireless

Satellites are very expensive to put up and to run

They cover a large area with a reasonably good bandwidth

They are good for remote and undeveloped areas with no other local infrastructure

Wireless

Geostationary satellites have a large latency: about 1/10 sec, which can be very noticeable in highly interactive applications (games)

So lately providers are putting satellites into low orbits, but this means they are forever moving (from the perspective of someone at ground level)

This is fixed by having a large number of satellites so there is always one overhead

Lower latency $\Longrightarrow$ lower orbit $\Longrightarrow$ faster moving satellites $\Longrightarrow$ more satellites needed to maintain coverage

Wireless

Starlink (amongst others) are currently building a low orbit satellite network

Targets are 300 Mb/s at 20ms latency

But this will need 10s of thousands of satellites (a problem for astronomers!)

Due to the cost, this may turn out to be a “top up” service for the hard to get at places; not a general connection for all

The Physical Layer

We have seen some implementations of the physical layer

There are very many more

There are many implementations as there are many physical requirements of networks (distance, speed, power, etc.)

Fortunately, as we go up the layers, the amount of variety decreases!

Link Layer Protocols

We now turn to some other link layer protocols

Serial Line Internet Protocol (SLIP) is an early protocol used on modems to encapsulate IP traffic over serial (telephone) lines

It is a *point-to-point* protocol, meaning it links just two machines to each other: the normal requirement in early dial-up systems

SLIP

SLIP frame

A very simple frame encapsulation with a terminating byte (hex) `c0`; also often a starting `c0` byte, too

SLIP

So how to send data that contains the byte `c0`?

Use *byte stuffing*

To send `c0` actually send two bytes `db dc`

The pair `db dc` is reconstructed as `c0` at the other end

Stuff `db` as the pair `db dd`, which the other end reconstructs as `db`

A minor expansion of data, but it enables transparent transmission of data

SLIP

There is no frame size limit, but it was suggested you should support at least 1006 bytes

296 bytes was common (40 bytes of TCP/IP headers plus 256 bytes of data)

Larger frames have relatively less overhead, but at 9600b/s (a typical early modem speed) 1006 bytes takes roughly 1 sec to transmit

If there is a bulk transfer of full-sized frames at the same time as an interactive session, the interactive session's frames would have to wait 0.5 sec on average to get through, much too slow

An interactive response of over 100-200ms is felt to be slow

SLIP

296 is a compromise: not too slow for interactive, not too small for bulk transfer, but not particularly good for either

On more modern modems (56Kb/s) it was increased to 1500 bytes

Exercise Compute the average latency for 296 byte frames on 9600b/s; and 1500 byte frames on 56Kb/s

Exercise And how big a frame could we have on a 10Mb/s Ethernet for the same latency?

SLIP

SLIP has several problems

Only IP in the next layer is supported (no type field in frame)

The ends must have pre-agreed IP addresses: no mechanism for agreeing addresses

No checksum: even though telephone lines were noisy and created data corruption

No authentication: no way to check *who* is connecting

PPP

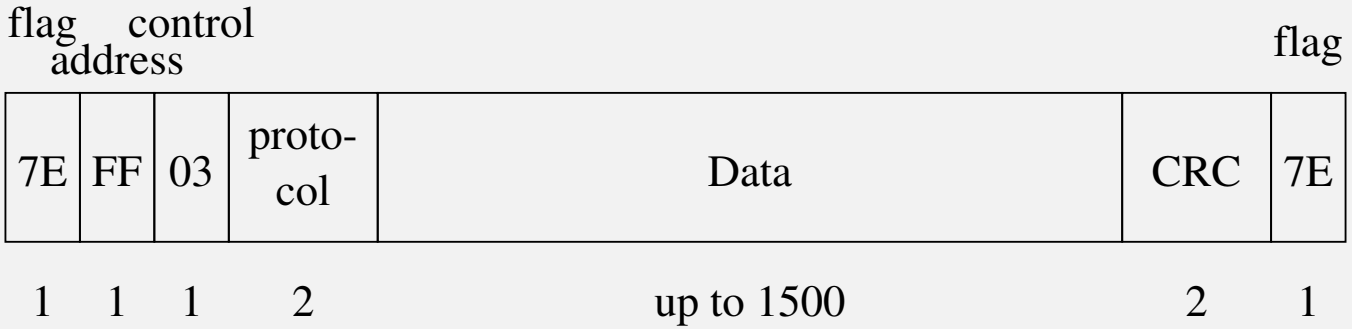
Thus the *Point-to-Point Protocol* (PPP) was developed

Like SLIP it is a point to point protocol

It has three parts

- A framing layout for packets
- A link control protocol (LCP) for managing and configuring links
- A set of network control protocols (NCP) to manage network layer specific options

PPP



PPP frame

- Frame delimiters `7E`, start and end
- Address (always `ff`), Control (always `03`)
- Protocol: tells us what the next layer is, e.g., IP is `0021`, LCP is `C021` and NCPs are `80xy`
- Cyclic redundancy check to spot corruption
- But no address fields (**Exercise** why not?)

PPP

- Up to 1500 bytes of data (but can be negotiated)
- Values are escaped (like SLIP) by `7d`

- `7e` $\rightarrow$ `7d 5e`
- `7d` $\rightarrow$ `7d 5d`
- x, where $x < 20_{16}$ $\rightarrow$`7d` [x+`20`], so, e.g., `01`$\rightarrow$`7d 21`

PPP

NCPs can negotiate extras, like compression, frame size, etc.

And authentication, e.g., passwords

While it was devised to be used over telephone modems, PPP is still actively used, e.g., in PPP over Ethernet (PPPoE) as it allows authentication of a connection

Current FTTC products use (IP over) PPPoE over VDSL to pass authentication to the ISP

Exercise Read about this

Exercise Look at the configuration of your home ADSL or VDSL modem

Link Layers

Several other link layers exist: too many to talk about in any detail

We have already seen the Ethernet frame for a local area network

There are many link layers for carrying data over long distances, at high data rates, both electrical and optical

Link Layers

For example, *Asynchronous Transfer Mode* (ATM) was popular for a while

Designed by telephone engineers, it was really a *connection oriented* digital voice network into which you could squeeze data packets

It has fixed-length frames of 53 bytes (48 data plus 5 header) — good for voice, not so good for data

Exercise Read about ATM and PPPoA, that layers (IP over) PPP over ATM, as used in ADSL and DOCSIS

Link Layers

Multiprotocol Label Switching (MPLS) was designed post-ATM when the technology decisions that drove the design of ATM were deemed no longer applicable

Designed by network engineers to be a general long-distance network, it is much better suited to modern data networks

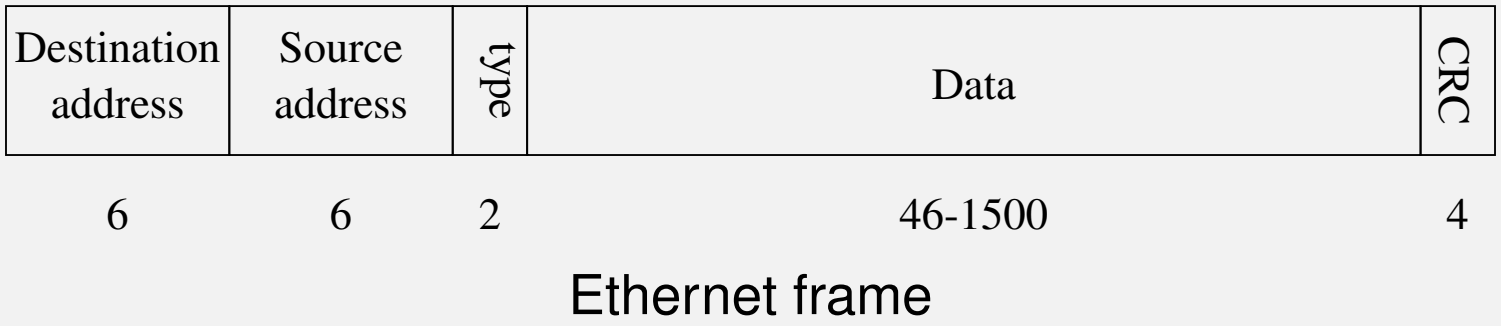
Exercise Read about MPLS and how BT uses it in its 21C Network

Networks

Ethernet Link Layer

We want to move up to the network layer: but before doing so there is one more remark on the link layer

Recall Ethernet. The data on the wire:



Ethernet frame

Networks

Ethernet Link Layer

The interesting fields here are the *addresses*

The addresses allow a frame to identify the intended destination (and source)

This works well enough when the destination is on the local Ethernet network

Which is shared (or switched), so the frame has no problem being seen by the destination host

Networks

Ethernet Link Layer

What to do when the destination is non-local?

We can't simply treat the world as a shared medium and broadcast the packet to everybody

And the network at the destination might not even be an Ethernet and will not have an Ethernet address

So we need *hardware independent addresses* to identify hosts that work independently of the physical network

In the Internet Protocol, these addresses live in the network layer

Networks Link Layer

IP

The network layer used in the Internet Protocol is called the *Internet Protocol* (IP)

It has the major function of dealing with *routing*, determining where a packet should go

Amongst a lot of other stuff, the IP header has network layer addresses

These are hardware independent, and in the same format across the entire Internet

Networks

IP

Each host on the Internet has an IP address that identifies it uniquely over the entire Internet

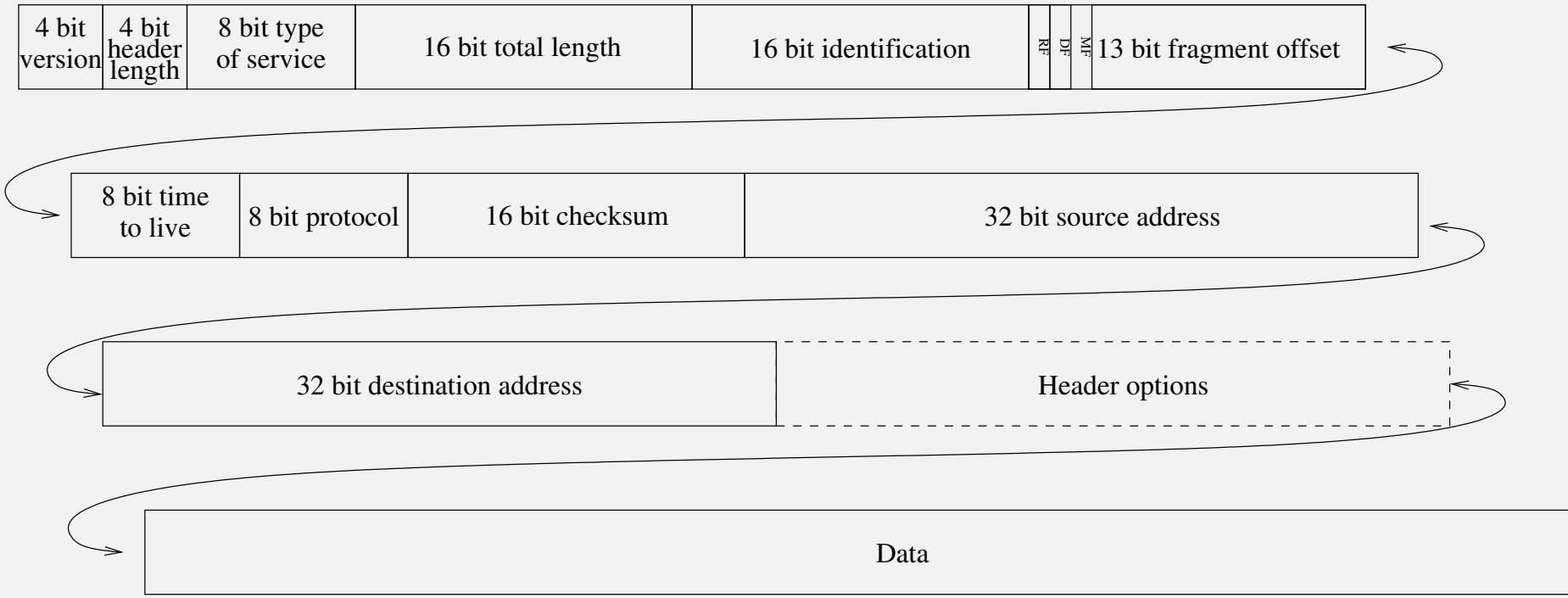
At least, that was the original intention

This is certainly no longer true, for reasons we shall explore later

But, for now, assume this is true

Networks

IP Header

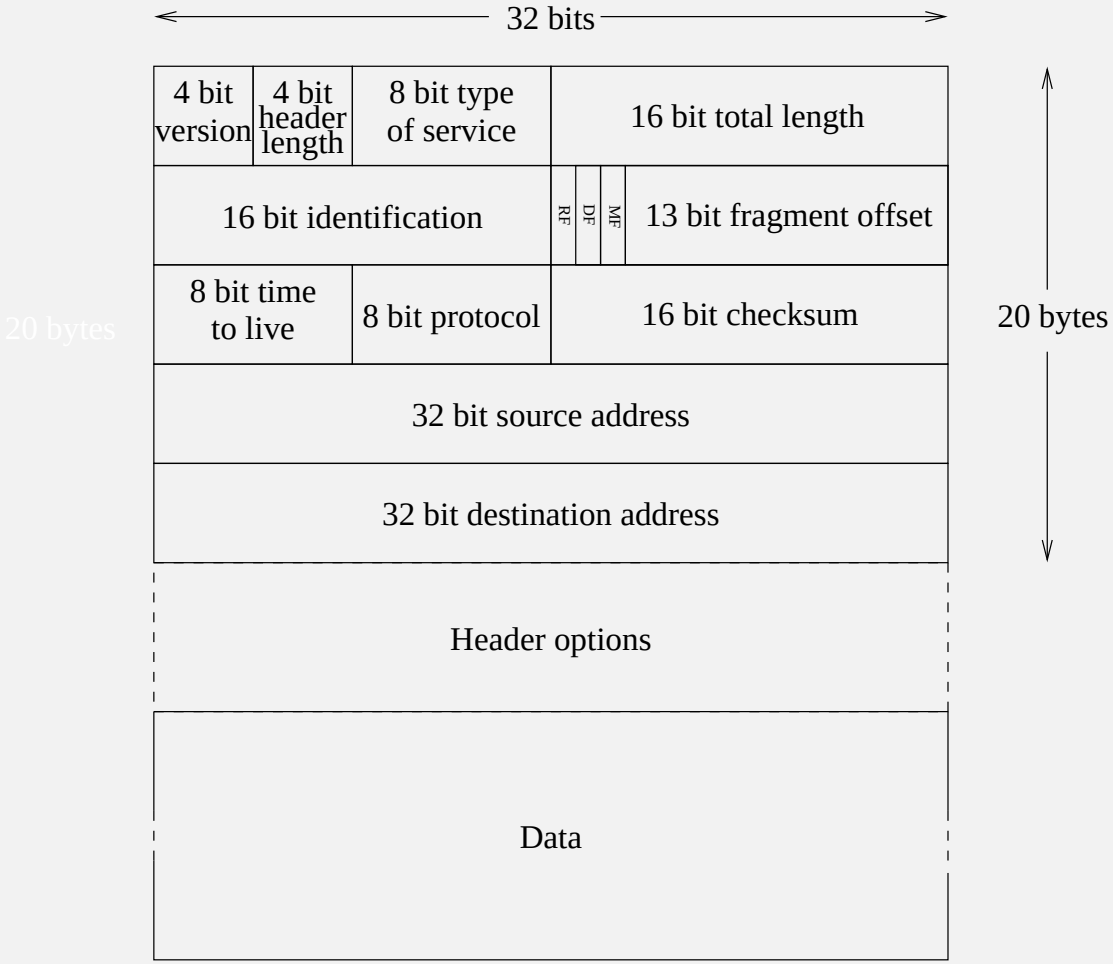


IP header

A bit hard to read, so conventionally we stack the header vertically

Networks

IP Header



IP header (usual layout)

Networks

IP

The source and destination addresses are both four bytes long: we shall come back to the other fields shortly

`10001010001001100010000000001110` is an example IP address, a 32-bit value

This is 2317754382 in decimal: not terribly easy to work with

So for convenience we write this as `138.38.32.14`, decimal representations of four 8-bit values. The dots are purely to make the number visually easier to read

But, importantly, there is *structure* in an IP address which helps with routing

Networks

IP

In this example, `138.38.32.14`. the first half `1000101000100110` (`138.38`) is a 16-bit *network* address, which identifies the University of Bath

And `32.14` is a 16-bit host address, which identifies a single machine on the University's network

Note that we write `138.38`, but this should really be thought of as `1000101000100110`, a bunch of 16 bits

Always remember that the dotted decimal notation is just a convenient way of writing a chunk of bits: there are no decimal numbers in the header!

Networks

IP

This division into network and host parts helps immensely in routing, as all packets destined for the University of Bath can be routed in the same manner

Only when a packet reaches the University is some local knowledge of the network needed

Indeed, the host part of this address splits further into *subnet* addresses that help local routing within the University

But the main point for now is that this IP address is independent of Ethernet and so can be used regardless of the hardware used

Networks

IP

So we have hardware independent addresses: but, now, there is an new problem

Suppose I want to send a packet to `138.38.3.40` on the local network. My data is (ultimately) encapsulated in an IP packet, with my IP address as source and `138.38.3.40` as the destination

(The question of how do we know the destination IP address must be answered later)

Now the IP packet must be further encapsulated in a hardware frame, Ethernet in this example. The OS can't send the packet on the physical medium until it knows the Ethernet address of the destination

Networks

IP

Ethernet does not know about IP addresses: it can carry any kind of data

IP does not know about Ethernet addresses: it can run on many kinds of hardware

And this separation of layers, as we know, is desirable

Networks

IP

We need some kind of *address discovery*, so given the IP address we can find the corresponding Ethernet address

This is done by the *Address Resolution Protocol* (ARP)

ARP is a very simple link-layer protocol that essentially broadcasts a special frame on the local medium to the effect of "who has IP address `138.38.3.40`?"

All hosts on the local network hear this broadcast and the host with that address replies "Me: and I have Ethernet address `08:00:20:9a:34:dd`"

(There are questions of security here. . .)

Networks

IP

The OS gets the ARP reply and can now use this information to write the correct address in the Ethernet frame

Only now can the original packet be sent

Networks

IP

We don't want to use ARP for *every* packet we send, so there is an ARP cache kept by the OS kernel that records the relation `138.38.3.40` $\leftrightarrow$ `08:00:20:9a:34:dd`

Entries in the cache time out and are removed after, say, 20 minutes

This is in case the host using `138.38.3.40` goes away and is replaced by a different host that uses the same IP address, but has a different Ethernet address: recall IP addresses are *not* associated with the hardware

Once expired, the next packet to `138.38.3.40` will need a fresh ARP

Networks

IP Routing

A quick note regarding when the destination is not on the local network

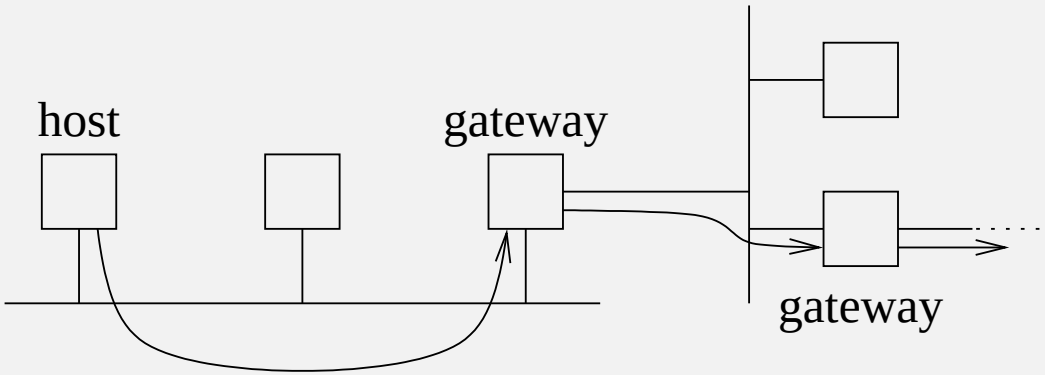
IP routing for the source host is quite simple: if the destination is on the local network, send the packet directly. This probably uses ARP (on the first packet) to get the hardware address of the destination

Networks

IP Routing

If the destination is *not* on the local network, to solution is to send the packet to a *gateway* host and let it deal with where to send it next

A gateway is just a machine on more than one network



Gateway

This keeps the complexity of the software needed on the hosts down: only the gateway will need to have a bit of intelligence about routing

Networks

IP Routing

So information a source host needs to know includes:

- its own address and network
- the address of a gateway machine

We shall see later how it gets this information

Networks

IP Routing

So, for a host the routing software is:

- is the destination on the local network?
- yes: send it directly, possibly with an ARP, if needed
- no: send it to the gateway, possibly with an ARP, if needed

Note in the latter case, the host might need to do an ARP for the *gateway*

Networks

IP Routing

In the non-local case, the packet is going to the gateway, so we would need to ARP for the hardware address of the *gateway*

The packet, with IP address of the final destination, is put into a frame with Ethernet address of the gateway

Since the packet needs to go to the gateway

So, here, the physical and network addresses in the Ethernet frame are completely unrelated!

Networks

IP Routing

This is another reason why we need both hardware and software addresses

The IP address is for the ultimate destination; the hardware address is for the next hop

Networks

IP Routing

ARP is not restricted to Ethernet and IP, but can be used to pair any physical and network layer addresses

Exercise Is ARP needed on a PPP connection?

ARP

ARP is a simple protocol

On an Ethernet, the ARP broadcast has to be put in an Ethernet frame, so what destination address does it put on the frame?

It broadcasts an *ARP Request* packet (protocol number 0806) in an Ethernet frame with destination hardware address `ff:ff:ff:ff:ff:ff` and source its own Ethernet address

All hosts on the local network read the frame

The target host recognises the request for its IP address

ARP

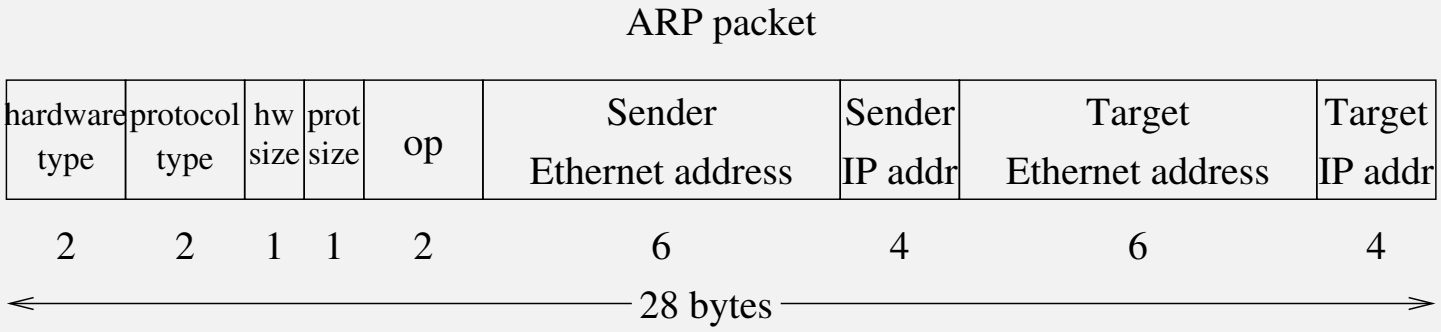
The target sends an *ARP Reply* packet (in a normal Ethernet frame) containing its own Ethernet address

It knows the source's Ethernet address as read from the request packet

The source gets the reply and reads out the target's Ethernet address. It can now use that Ethernet address to send IP packets

The other hosts on the network need do nothing

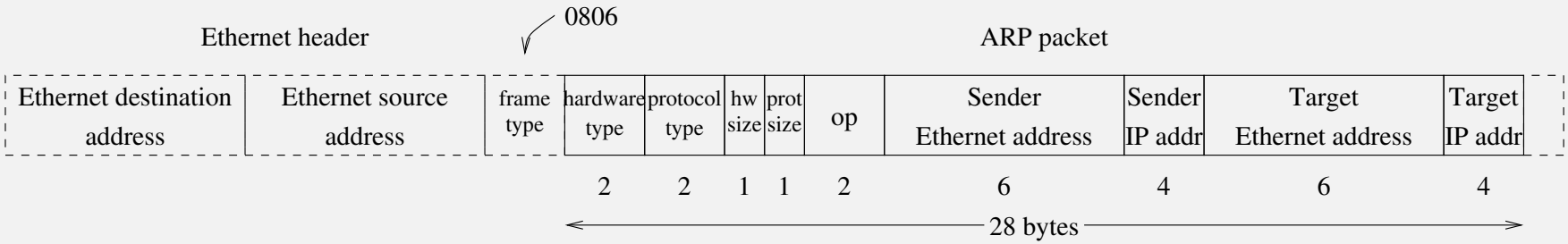
ARP



ARP packet

The Ethernet frame type for ARP is 0806

ARP

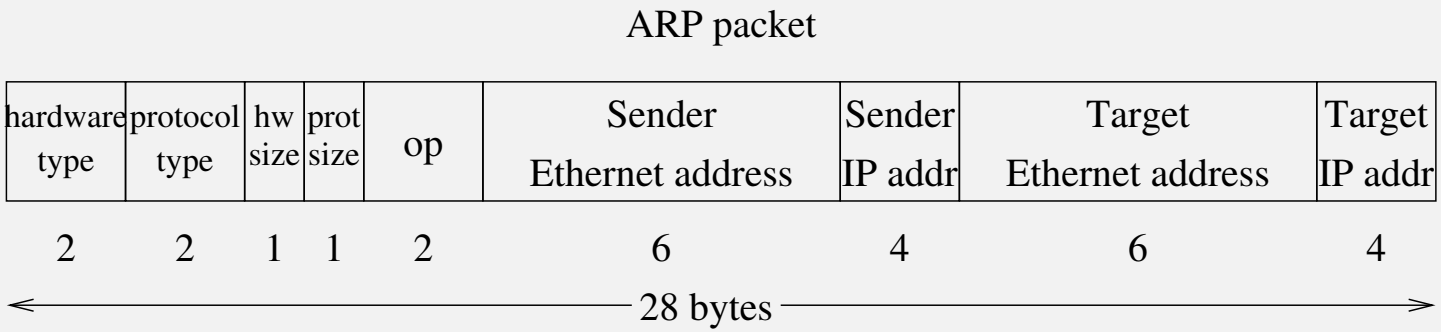


ARP packet within Ethernet frame

Contained within an Ethernet frame

The Ethernet type field allows the software that reads the packet from the Ethernet card to pass the contents of the packet to the software that implements ARP

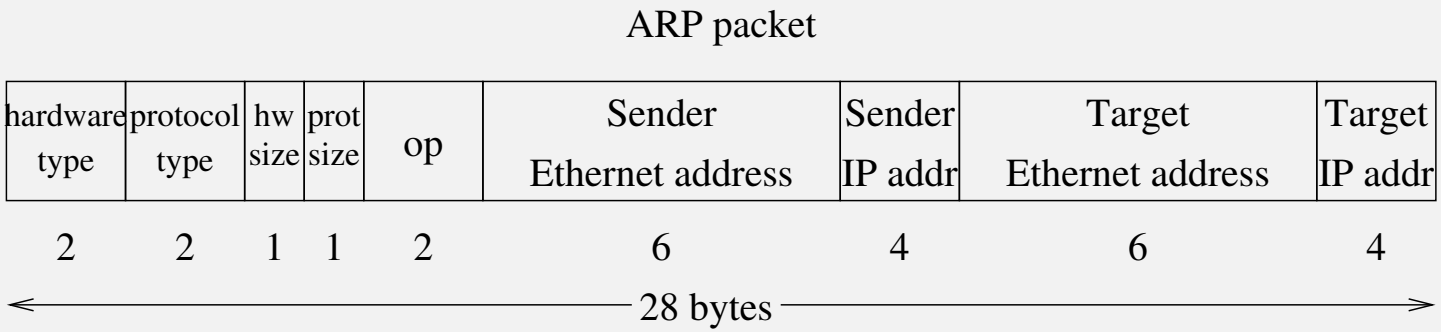
ARP



ARP fields

- Hardware type: 1 for an Ethernet address
- Protocol type: 0800 for an IP (version 4) address
- Sizes: sizes in bytes of the address fields, 6 for Ethernet, 4 for IP

ARP



ARP fields

- OP: 1 for a request, 2 for a reply
- Address fields, with lengths as given: the data
- In a request the destination hardware field is not filled in as this is what we are trying to find!
- In a reply the sender Ethernet address is the address we seek

ARP

If no machine on the local network has the requested IP address, or that machine is down, no reply will be forthcoming

In this case, after a few seconds, and a few repeated ARP requests, the OS returns an error message to the application trying to make the IP connection

This might be “no such host” or “host unreachable”

ARP

It is sometimes useful to give an ARP *reply* even if nobody has asked for it. For example a new machine joins the network or an existing machine changes its IP address for some reason

This is a *gratuitous ARP*

All machines on the local network are free to read any ARP request or reply they see and modify their own ARP caches accordingly

ARP

So a gratuitous ARP would help break old associations that are no longer valid but still cached

Without a gratuitous ARP a host might send an IP packet to the old cached, but now out-of-date hardware address

ARP

ARP is purely a local network thing: discover a hardware (next hop) address **on the local network**

And it makes no sense for gateway to forward an ARP to another network, which might not even be of the same physical type

ARP

There is a interesting trick that shows ARP can be used for things other than it was designed to do: and shows how the Internet Protocols are incredibly *malleable*

Used in the days before switches were common: this trick is unlikely to be used these days

ARP

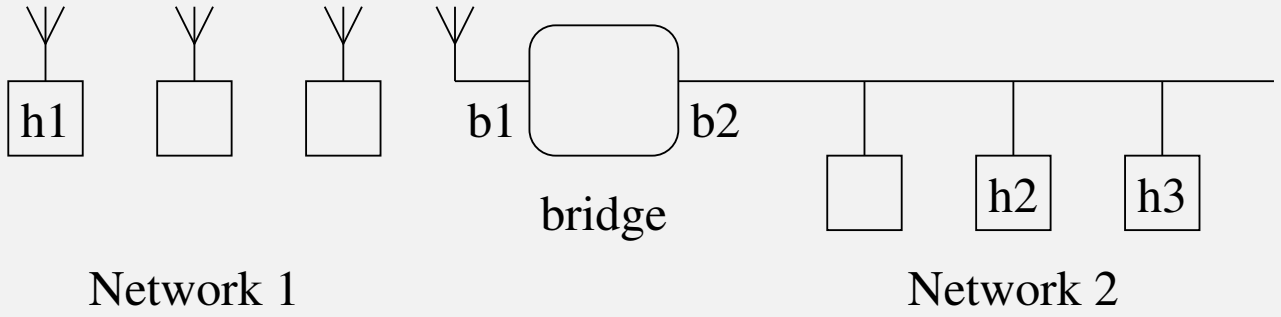
This trick allows us to extend an Ethernet (or other network) over a physically larger distance than its specifications allow, and to join a wireless network to a wired one so they appear to be a single network

A *bridge* is a host that joins two physical networks into one. It has two interfaces, one on each network

Note: this is different from a gateway we mentioned earlier, that connects two *different* networks

ARP

ARP Bridging



ARP bridge

This example joins a Wi-Fi to an Ethernet, but we could have any two networks that share a MAC address type

If host h1 wishes to send to host h2 it must determine its hardware address (as it is on the “same” local network)

So h1 does an ARP broadcast for h2, just as normal

The bridge sees this request and responds on behalf of h2 (a *proxy* ARP), but it supplies its *own* hardware address b1

ARP

ARP Bridging

Now h1 sends data to what it thinks is h2, but is actually the bridge

The bridge reads the data packet, sees it is destined for h2 (by its IP address) and forwards it to the other network where h2 can read it

Furthermore, it rewrites the forwarded frame's header to have h2 as destination and b2 as source

If h2 replies, it can either use b2 which it got from the original packet or do an ARP request, which the bridge proxies in a symmetrical way

ARP

ARP Bridging

In either case the packet goes to the bridge, which forwards it to h1, again rewriting the frame addresses appropriately

This is all transparent to h1 and h2 who believe they are on the same network

If h1 is communicating with both h2 and h3 its cache will show them to have the *same* hardware address b1: this is not a problem

ARP

ARP Bridging

Exercise Find out if your home network does ARP bridging, or if it simply acts like a switch on a single network

Exercise Make sure you understand the difference between what a gateway does, what a switch does and what a bridge does

Virtual Bridging

Bridging is useful, but shouldn't be taken too far

Larger networks have more traffic

Just think of the ARP broadcasts alone!

It is often better to split a large network into several smaller ones: see subnetting, later

RARP

Exercise Read about *Reverse ARP* (RARP): given a hardware address find the IP address

Network Layer

We have briefly seen the Network Layer header in the IP to see its addresses

We now need to look at the Internet Protocol (network layer) in more detail

It is the basis the Internet is built upon

It is actually relatively simple, but allows more complex stuff to be layered on top

We shall start by describing IP version 4, IPv4

And talk about IPv6 later

IP

IP is a best-effort, connectionless, unreliable, packet based protocol

Recall: “unreliable” means “not guaranteed reliable”

“best-effort”: no guarantee of delivery, or of any special features, like quality of service

“connectionless”: each packet is independent of all others, there is no relationship between individual packets (in this layer)

IP

IP's primary purpose is getting packets from source to destination

It doesn't rely on any particular property of a link layer, so it can run on top of almost any link layer, even unreliable ones

But, first, some vocabulary

IP

Recall that IP is a cooperative system: for a packet to get from source to destination it is handed from one network to the next, hop by hop

The nodes in the network have various roles:

- Host. A machine you actually use to do some work
- Bridge. Connects two physical networks together
- Gateway. Provides a connection off the local network
- Router. A machine joining two or more networks and whose primary function is to determine where a packet goes next (i.e., routing)

These are not mutually exclusive: gateways and routers can be hosts; gateways do trivial routing

IP

Marketing alert: things you see described as “routers” in the shops are unlikely to be actual routers, which are specialist bits of equipment

To them, the word “router” seems to means “a box you plug into the network”

It should really mean “a box that engages in routing protocols”

IP

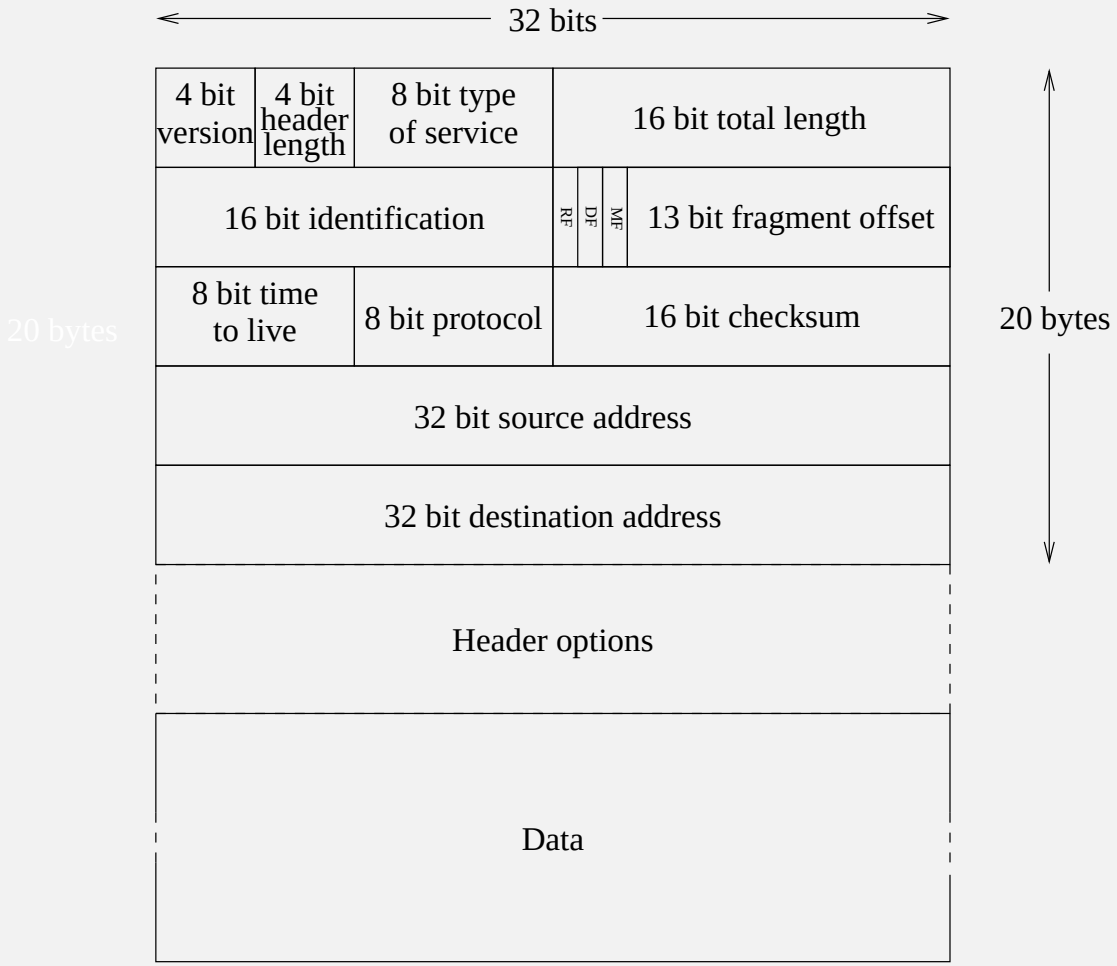
The basic idea is that a packet does not know how to get from source to destination: this is the routers' job (and it can be quite complex: see later)

The IP layer takes bytes from the transport layer and prepends a header to support routing (and other things), producing packets often called *datagrams* in this layer

The IP specification says datagrams can be up to 64KB in size, but they are usually in the region of 1500 bytes (Ethernet, again), but can be much larger (e.g., 9000 bytes) in specialist networks

We return to the IP header

IP



IPv4 datagram header

IP

- Version. Four bit field containing the value 4. A later version of IP (IPv6) contains 6
- Header length. There are some optional fields, so the header can vary in size, so this is needed to pinpoint the end of the header. Given as a number of 4 byte words. Four bits, maximum value 15, so maximum header length of 60 bytes
- Type of service. Eight bits. To indicate to a router how this datagram should be treated in terms of cost, speed and reliability (if possible)

E.g., for audio it is better to get data through quickly rather than 100% reliably as the human ear is more sensitive to gaps than occasional errors

IP

TOS/DS

The TOS field, these days called the *Differentiated Services Field* (DS field), is to inform routers on the best way to treat this datagram

This allows the implementation of *Quality of Service* (QoS)

The full range of options available is complex (see RFC2474 for details), but can indicate things like

- Minimise delay. Do not hold onto this datagram longer than necessary, and perhaps prioritise it over others

IP

TOS/DS

- Maximise throughput. Not quite the same as minimising delay, since collecting together several small datagrams and sending them off together may be more bandwidth efficient
- Maximise reliability. Try not to drop this datagram if the router is becoming overloaded; drop another datagram first
- Minimise cost. For this datagram cost is more important than reliability or speed. This datagram can be delayed if it makes transmission cheaper

IP

TOS/DS

Early routers ignored the TOS field, but these days QoS is very important

Modern routers do (or should) pay attention to the DS field

Here, as in some other parts of the IP specification, a router may ignore some information if it wishes. It might be the software is so old it does not recognise a modern field; or it might simply be unable to make use of the information. You are strongly recommended to act on the information, though

Exercise Look up the problems *Explicit Congestion Notification* (ECN) had when it was introduced

IP

- Total Length. Of the entire datagram, including header, in bytes. 16 bits, so giving a maximum size of 65535 bytes. Much larger than domestic networks need, but too small for high-speed networks

As usual, larger packet sizes mean lower overheads:

- Time overhead in hosts of splitting data into datagrams, adding headers, then removing headers and reassembling
- Bandwidth overhead as each header is 20 or more bytes that is not data; plus more gaps between datagrams
- Time overhead in routers of processing packets

IP

The Total Length field is essential over Ethernet as it pads short frames

Thus we get better independence from the MAC/PHY layers

IP

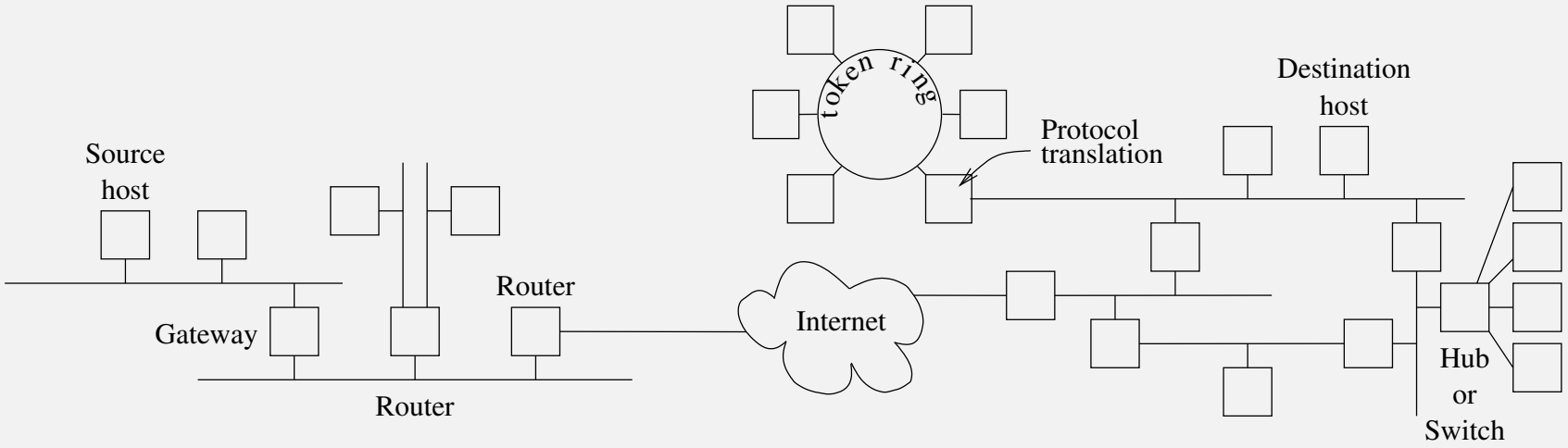
- Identification. 16 bits. A value that is unique to each datagram sent by the source, often incrementing by 1 for each successive datagram sent

Used in *fragmentation* to reassemble the fragments of a single datagram. All the fragments get their own IP header, but share the same identification

So we need to discuss fragmentation

IP

Fragmentation



Many kinds of hardware in the Internet

The path a packet takes from source to destination will typically go through a wide variety of differing kinds of hardware

IP

Fragmentation

For example, your home Ethernet/Wi-Fi might support 1500 byte packets, while your VDSL link might have a maximum of 1480 bytes

Thus IP must face the problem of differing link layer properties, in particular maximum frame size

IP

Fragmentation

If a big datagram hits a part of the Internet that can only cope with small datagrams, there is a problem

IPv4 deals with this by *fragmentation*: a datagram can be subdivided by a router into several smaller datagrams; it is the destination's problem to glue them back together

In the right order

The fragmentation fields in the IP header deal with this

IP

Fragmentation

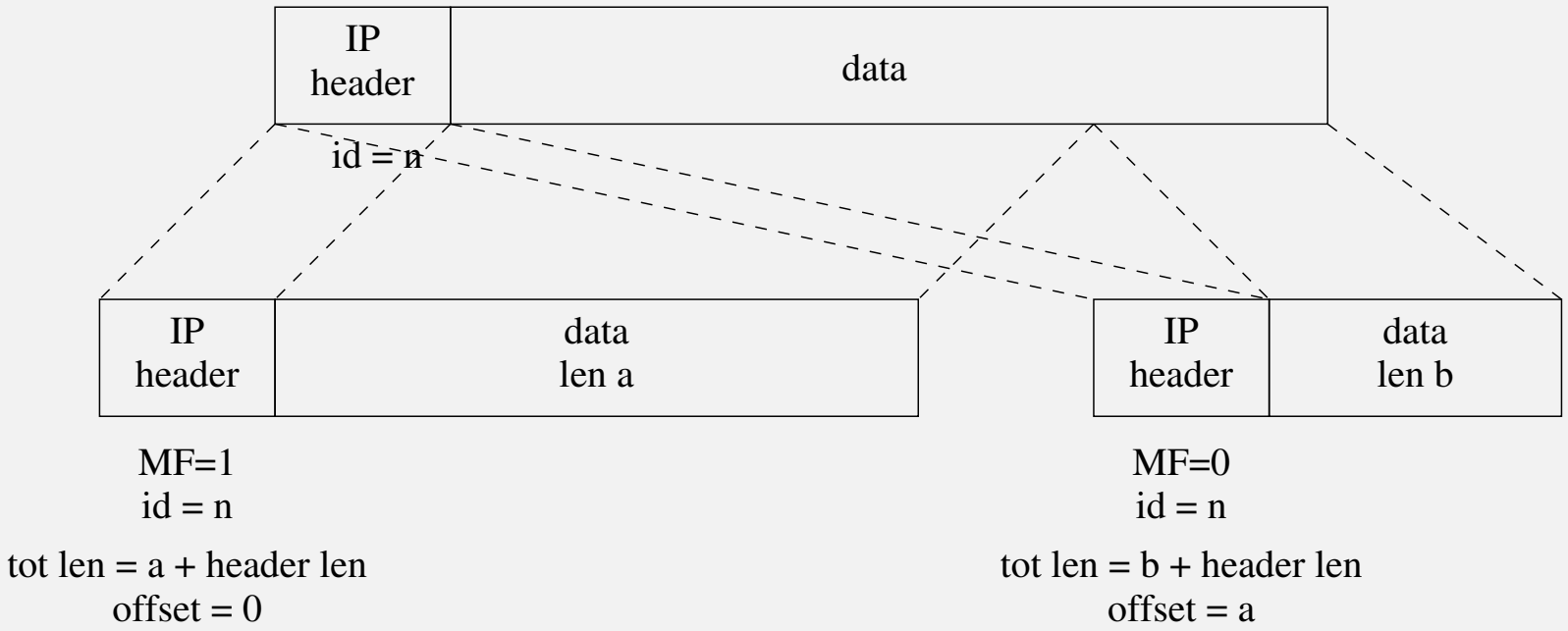
- Flags. Three bits: two used and one reserved

1. RF. Reserved for later use, must be 0 (see RFC3514 for a suggested use)
2. DF. Don't fragment. If a host can't (or doesn't want to) deal with fragments this bit is set to inform the routers on the path to the destination. A router might choose an alternative non-fragmenting route, or simply drop the datagram and send an error message back to the source which can then send smaller datagrams.
   All hosts are required to be able to accept datagrams of 576 bytes
3. MF. More fragments. All fragments except the end fragment have this set

IP

Fragmentation

Fragment Offset. Where this fragment came from in the original datagram



Fragmenting an IP datagram

IP

- Fragment Offset. 13 bits, giving the byte offset divided by 8. E.g., value of 20 means an offset of 160 bytes

So 13 bits is enough to cover the 16 bit range of sizes

And every fragment (apart from the end fragment) must be a multiple of 8 bytes long: the router doing the fragmentation must ensure this

Note that a datagram may be split into any number of smaller fragments, not just two

IP

Fragmentation

Every fragment has a copy of the original IP header, but with the various fragmentation and length fields set appropriately

In more detail: each fragment header will be a copy of the original header apart from

- total length: set to the fragment size
- MF: set to 1 if this is not the end fragment
- fragment offset: set appropriately
- (TTL and checksum: set appropriately)

IP

Fragmentation

In particular, all the fragments of the original datagram have the same the identification field value

When the fragment with MF $= 0$ is received, its fragment offset and length will give the length of the original datagram

The destination can then reassemble the original datagram when all the fragments have arrived

Fragments are IP datagrams, so as always, they can arrive in any order; or not at all

IP

Fragmentation

IPv4 spends a lot of effort coping with fragmentation

It is costly and should be avoided

- Performing fragmentation in a router takes time
- More overhead as more datagrams for a given amount of data
- More overhead as more datagrams are traversing the network
- More datagrams means a greater probability one will be lost or corrupted

IP

Fragmentation

- If a fragment is lost, the *entire* original datagram must be retransmitted: there is no mechanism in IP to indicate which fragment was lost
- Fragments are datagrams in their own right and can themselves be fragmented

Fragment processing software (particularly reassembly) has a history of buggy implementations leading to hacked machines

Exercise Consider what happens when a fragment is further fragmented. Differentiate the cases of MF $= 0$ and MF $= 1$

IP

Fragmentation

Fragmentation is such a costly process that modern implementations try very hard to avoid it

They employ *MTU Discovery*

IP

Fragmentation

Setting DF in the header prohibits fragmentation; if a router cannot avoid fragmenting it drops the datagram and returns a "fragmentation needed but DF set" error message back. The sender can then send smaller datagrams

This allows *MTU Discovery*. The *Maximum Transmission Unit* (MTU) is the largest datagram a host or network can transmit

The *path* MTU is the smallest MTU for the entire path from source to destination

A datagram not larger than the path MTU will not get fragmented

MTU Discovery works by sending variously sized datagrams with DF set, and monitors the errors returned

IP

Fragmentation

When a datagram reaches the destination with no fragmentation error we have found a lower bound for the path MTU

This bound is approximate as the network is dynamic and paths may change!

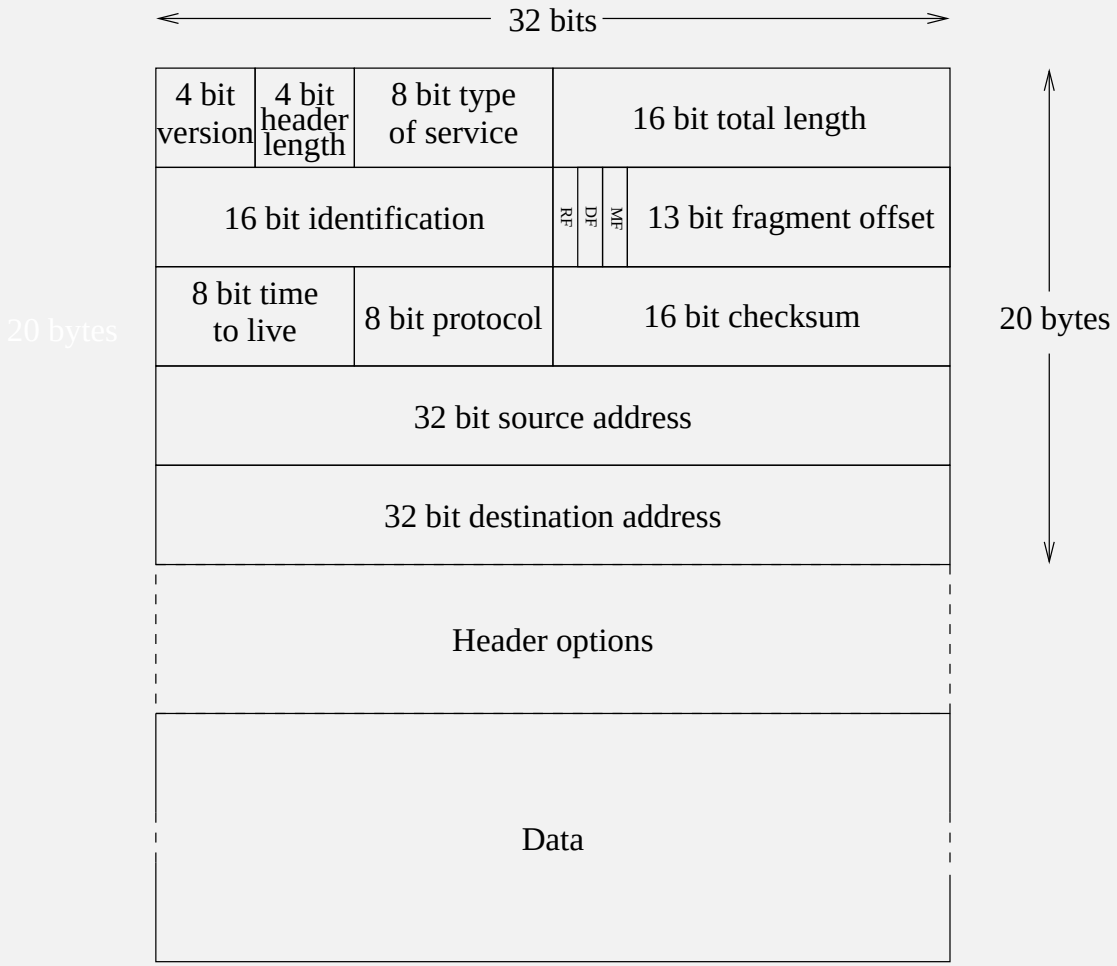
This is the approach IPv6 adopts: don't have fragmentation in routers, but require MTU discovery (see later)

Note: a host is not required to implement MTU Discovery in IPv4: it's just good if it does as fragmentation is such a large overhead

IP

Back to the IPv4 header fields

IP



IPv4 datagram header

IP

- Time To Live. An eight bit counter used to limit the lifetime of a datagram

Poorly configured routers might bounce datagrams back and forth or in circles indefinitely, thus clogging the network with lost datagrams

The TTL starts at 64, or 32, say, and is reduced by one as it passes through each router

IP

If a TTL ever reaches 0 at a router, that datagram is discarded, and an error message (“time exceeded in-transit”) is sent back to the source of the datagram

This limits errant datagrams: eventually the TTL must reach 0 and the datagram is dropped

Eight bits means a maximum path of length 255, but this seems enough for the current Internet: no valid paths as long as this are known

The *width* of the Internet is the length of the longest path: this is uncertain and constantly changing but definitely over 32

IP

Originally the TTL was to be a measure of *time*, reducing by one for each second in a router. In practice no implementations did this, but just decremented by one regardless. This is now the expected behaviour

Again: this is IP being pragmatic, following what people actually do in implementations, rather than the letter of the specification

Exercise Why doesn't everyone simply put 255 into the TTL field?

IP

- Protocol. This eight bit field connects the IP layer to the transport layer. This is a value indicating which transport layer software to pass the contents of the datagram to. For example, UDP is 17 and TCP is 6
- Header checksum. As for the Ethernet header, this is a simple function of the bytes in the IP header. If the checksum is bad, the datagram is silently dropped. A higher layer must detect this and perform whatever action it needs. Recall that the IP layer is not guaranteed reliable

IP

The checksum is checked in each router, as well as the final destination

This ensures corrupted datagrams are dropped as soon as possible

Note the checksum includes the TTL field so it must be recomputed and rewritten in the datagram by each router the datagram passes through (this increases the time a router must spend on each datagram)

IP

- Source and Destination Address. 32 bit numbers that uniquely determine the source and destination machines on the Internet

“Uniquely” is now not true

32 bits limits us to at most 4,294,967,296 hosts on the Internet

Not enough, and a significant chunk of those addresses are reserved for special purposes and can’t be used for host addresses

We will come back to this very important problem later

IP

The optional header fields allow for items that are either

- not common, so you don't want to pay the overhead of always having them, or
- protocol extensions that are useful but were not included in the original IP specification

When IP was first devised, networks ran over telephone modems at a few thousand bits/sec. Now we expect giga and terabits. The physical layer has changed immensely while basic TCP/IP is pretty much unchanged!

A flexible approach across the entire TCP/IP suite such as allowing for options is part of this

IP

IP layer options are not much used. Options include

- Security: privacy and authentication
- Record Route. Each router records its address in the option header as the datagram passes by
- Timestamp. Each router records its address and the current time in the header as the datagram passes by
- Strict Source Routing. A list of addresses that give the entire path from source to destination
- Loose Source Routing. A list of addresses that must be included in the path from source to destination

IP

Most IP options are for debugging or profiling behaviour, and not much used

These days, things like privacy and authentication are mostly done in other layers

Moreover, the IP header length field has a maximum value of 60 bytes; the fixed part of the header is 20 bytes; thus options can use up to 40 bytes

This severely restricts what we can do in the options

IP Addresses

We now go back and look at the IP addresses in more detail

Roughly (and incorrectly) speaking, every machine on the Internet has a unique address

These are not random, but allocated in such a way to make routing between hosts much easier

If there were no structure on the addresses every router everywhere would have to know where every host in the world was

Impossible, for technical, political and security reasons

(Note that MAC addresses do not encode any network information, another reason for separate software addresses!)

IP Addresses

Recall the Internet is a collection of networks

An IP address is split into two parts:

- A network number
- A host number on that network

The host number defines the host “uniquely” on a network

The network number defines a network “uniquely” on the Internet

IP Addresses

As we have already seen, to an end host routing is trivial

- If the destination is on the same network, simply put the packet out on the network
- If not, send the packet to a gateway, and let it deal with the problem

It can tell if the destination is on the same network as itself by comparing the network part of their addresses

If they are the same, they are on the same network!

IP Addresses

To a gateway or router the problem is to send the packet on towards the destination network

Which one? This is the difficult bit

But there are very many fewer networks than hosts, so this is already a great simplification

Currently (2023): around 115000 top-level networks

A router contains a table of IP addresses, with a next-hop neighbour router associated with each address

(Actually a more complicated datastructure than a simple table, but we can think of it as a table)

Containing network addresses, but individual host addresses are possible, too

IP Addresses

Each row in the table contains

- A destination address. This can be the address of a single host, but is usually a network address
- The address of the next hop router, i.e., the address of where to send the packet next. This is the address of a router that is directly connected to the current one
- Which interface to send the packet out on to get to that router. A router has many interfaces and this describes which one to use

IP Addresses

When a packet arrives at a router it checks the table

- If the packet destination matches a host address in the table, send the packet to the indicated host on the indicated interface
- Else if the packet destination's network part matches a network address in the table, send the packet to the indicated host on the indicated interface
- Else if there is an entry in the table marked "default", send the packet to the indicated host on the indicated interface
- Else error: drop the packet and return an error message "network unreachable" to the source

IP Addresses

For now, we can regard routers as machines with tables like this that tell them where to send packets. We will see how the tables are created later

End hosts have routing tables, too: they are very simple, just encoding the local/non-local decision

IP Addresses

```
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
138.38.96.0     *               255.255.248.0 U     0      0      0 eth0
127.0.0.0       *               255.0.0.0     U     0      0      0 lo
default         138.38.96.254 0.0.0.0       UG    0      0      0 eth0
```

A simple routing table as might be found in an end host on network `138.38.96`

- Send local traffic directly to the destination out on interface `eth0`
- Otherwise send to the default gateway `138.38.96.254`, also on interface `eth0`

IP Addresses

The *netmask* (`Genmask` in the example) tells us how to divide an IP address into network and host parts. More details later, but a 1 bit set in the mask indicates this bit is part of the network address

Work down the table ANDing the destination address on a packet with each netmask in turn. If the result equals the `Destination` value, we use this row to route the packet

A mask of 255.255.248.0 is 11111111111111111111110000000, so for this example the network part is the top 21 bits of the IP address

"`default`" is actually destination 0.0.0.0, and so always matches any address after an AND with mask 0.0.0.0

IP Addresses

There is also a *loopback* address `127.0.0.1` for a virtual internal network connecting the machine to itself on (virtual) interface `lo0`

This is useful for many things, such as testing

IP Addresses

A table from a machine with more than one real interface:

```
Destination     Gateway         Genmask          Flags Metric Ref  Use Iface
213.121.147.69  *               255.255.255.255  UH    0      0      0 ppp0
172.18.0.0      *               255.255.0.0      U     0      0      0 eth0
172.17.0.0      *               255.255.0.0      U     0      0      0 eth1
127.0.0.0       *               255.0.0.0        U     0      0      0 lo
default         213.121.147.69  0.0.0.0          UG    0      0      0 ppp0
```

There are three interfaces: `eth0`, `eth1` and `ppp0` (as well as `lo`)

A packet with address `213.121.147.69` goes directly out on interface ppp0; Packets with addresses in the network `172.18` go directly on interface eth0; Packets with addresses in the network `172.17` go directly on interface eth1; Otherwise packets are routed to the gateway `213.121.147.69` on the interface ppp0; The first row of the table is actually redundant here, but is kept as it can speed up the checking in a common case Other

IP Addresses

Exercise Look at the routing table on your machine. E.g.,

```
ip -r route show
```

under Linux.

IP Addresses

How should we divide the 32 bit IP address into network and host parts?

8 bits for network? Then $2^8 = 256$ networks each with $2^{24} = 16777216$ possible hosts

- Not enough networks for the entire Internet!
- Too many hosts for most institutions

IP Addresses

24 bits for network? Then 16777216 networks each with 256 possible hosts

- Plenty of networks
- Not enough hosts per network for large installations

IP Addresses

16 bits for network? Then 65536 networks each with 65536 possible hosts

- Not really enough networks
- Plenty of hosts per network

IP Addresses

One solution: do *all* of the above

Divide the space of IP addresses into parts, where each part has a different network/host split

We shall now describe a *class* scheme that is historically important, but no longer used — for reasons that will become clear

IP Addresses

Class A networks. From `0.0.0.0` to `127.255.255.255`

The leading bit of the address is 0

Have 7 bits for network and 24 bits for host

This is 126 networks (networks 0 and 127 are reserved) each with 16777216 host addresses

The dotted-decimal representation is helpful here: address `x.y.z.w` has `x` as network, `y.z.w` as host

IP Addresses

Class B networks. From `128.0.0.0` to `191.255.255.255`

The leading bits of the address are 10

Have 14 bits for network and 16 bits for host

This is 16384 networks each with 65536 host addresses

The address `x.y.z.w` has `x.y` as network, `z.w` as host

IP Addresses

Class C networks. From `192.0.0.0` to `223.255.255.255`

The leading bits of the address are 110

Have 21 bits for network and 8 bits for host

This is 2097152 networks each with 256 host addresses

The address `x.y.z.w` has `x.y.z` as network, `w` as host

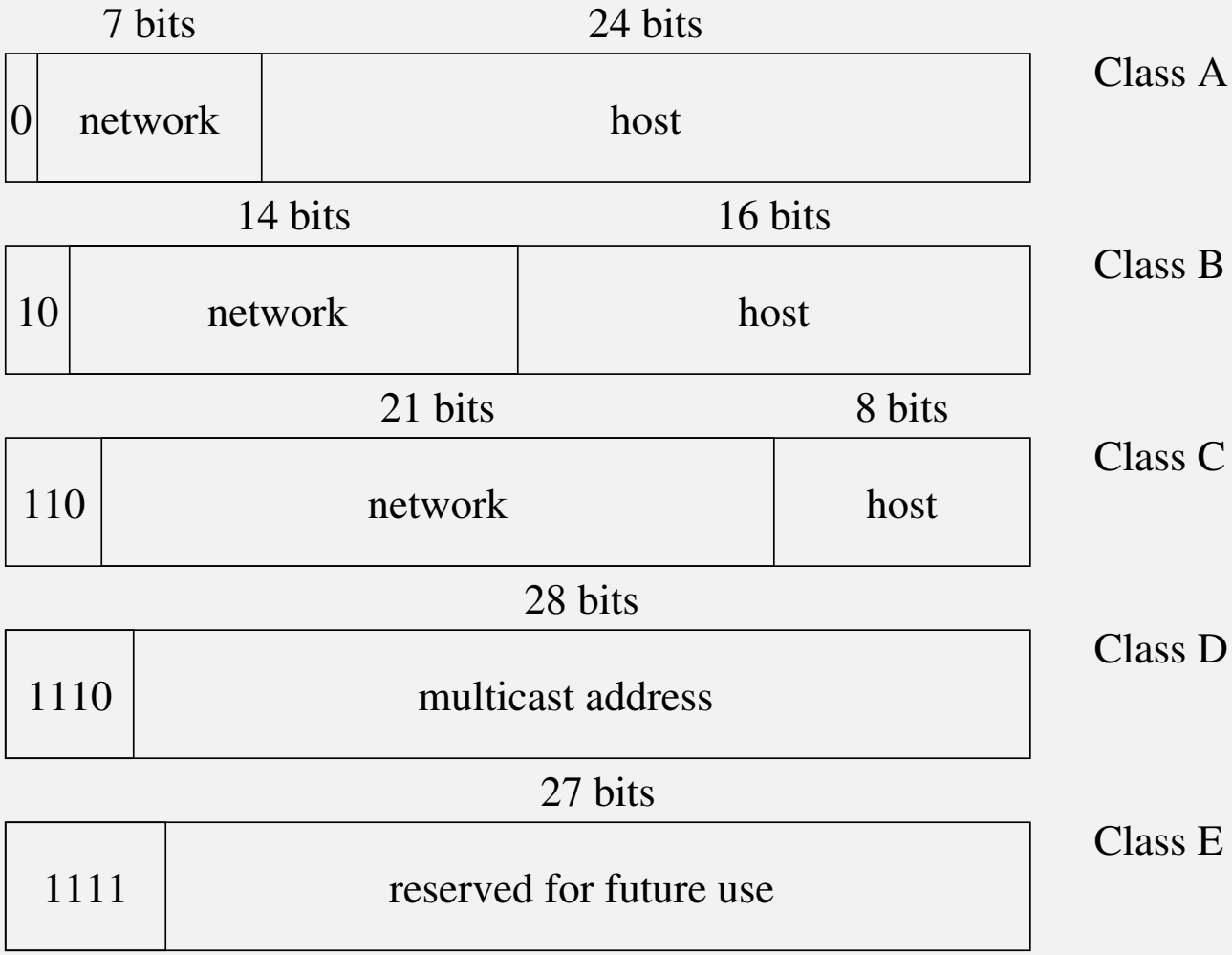
IP Addresses

The remaining addresses are kept for separate purposes

Class D. `224.0.0.0` to `239.255.255.255`; leading bits 1110.
Used for *multicasting* (details later)

Class E. `240.0.0.0` to `255.255.255.255`; leading bits 1111.
Reserved for future use

IP Addresses



IP address ranges

IP Addresses

An example: the University of Bath has been allocated addresses in the network `138.38.0.0`

This is in the class B address range and so there are 65534 possible hosts

Network 17.0.0.0, a class A address, is allocated to Apple

Network 193.0.0.0, a class C address, is allocated to *Réseaux IP Européens* (RIPE), the Internet Registry responsible for the allocation of IP addresses within Europe

IP Addresses

Two of the host addresses on each network are treated specially

Host parts of “all zeros” and “all ones” are not used as general host addresses, but are reserved for a special purpose

E.g., `138.38.0.0` and `138.38.255.255` in a class B

Thus the number of usable host addresses in a network is 2 fewer than you might think

IP Addresses

- Host part all 0s: “this host”. Originally specified to refer back to the originating host. But some implementations mistakenly used this as a broadcast address, so for safety it is not commonly supported as a valid host address. For, say, a class B network `17.16`, a packet sent to `172.16.0.0` *should* boomerang right back to the sender. But rarely does
- Host part all 1s: broadcast address to network. E.g., `172.16.255.255` sends to all hosts on the `172.16` network; very commonly used
- (Network part all 0s: “this network”. E.g., `0.0.12.34` would send to a host on the current B network. Again, not often implemented)

IP Addresses

So this is why you have two fewer addresses available than you might think

- . . . 255 is a broadcast address
- . . . 0 may or may not be supported, so best to avoid it

IP Addresses

And there are several special addresses, for example loopback addresses:

- Network `127.0.0.0`: the *loopback network*. Always implemented. The address `127.0.0.1` is commonly used as a way for a host to send a packet to itself over the internal loopback network on interface `lo`.
- Notice this is different from the same host sending to itself via an external network (e.g., using the interface's own address) as the former packet possibly won't go through the normal Ethernet/whatever software and hardware.
- The loopback network is there even if there is no real network hardware attached

IP Addresses

So the class scheme allowed IANA to allocate large chunks of addresses to people who need them, and small chunks to those that only need a few

This scheme has been historically very successful, but with the growth of the Internet has revealed several weaknesses. These days, a *classless* allocation is used (CIDR, later)

Thus this allocation is sometime called *classful*

To understand classless allocation, we first need to look at *subnetting*

IP Address Subnetting

Suppose you have been allocated class B network: 64 thousand host addresses are very hard to manage

Think of the broadcast traffic (e.g., ARP)

Physical/Technical issues (e.g, limits on Ethernet)

Political issues (e.g., traffic from one department must be kept separate from another department)

A single big network is not a very good idea

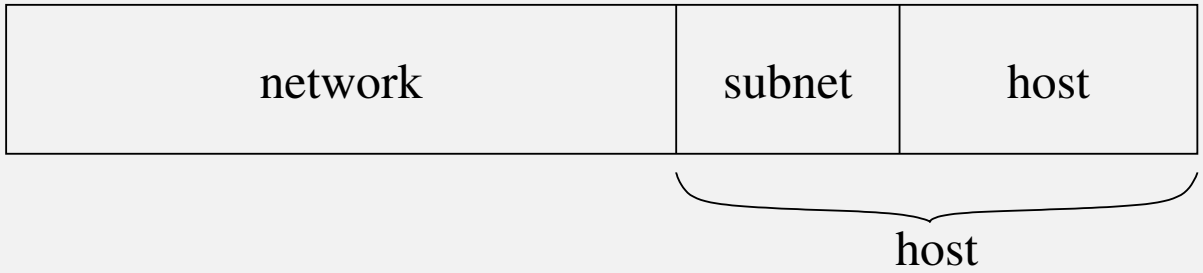
IP Address Subnetting

We can use *subnetting* to split our network into smaller pieces

Subnets can be administered by separate departments and are joined by routers

Just like the Internet!

And to do this, also just like the Internet, we further split the host part into some bits for the subnetwork and the rest for the actual hosts



Subnet addressing

IP Address Subnetting

Hosts will need to know which bits are the subnet part to be able to decide how to route packets: there is no class system here

We use a *subnet mask*

For example, the University of Bath has a class B, address `138.38`. The top 16 bits are the network address

The netmask 11111111111111000000000000 indicates which bits are in the network part

IP Address Subnetting

The Department of Mathematical Sciences has a subnet consisting of addresses `138.38.96.0` to `138.38.103.255` (2048 host addresses)

This corresponds to the netmask
11111111111111111111100000000000

IP Address Subnetting

network address	138.38.96.0	10001010 00100110 01100000 00000000
broadcast address	138.38.103.255	10001010 00100110 01100111 11111111
netmask	255.255.248.0	11111111 11111111 11111000 00000000

A machine can tell if an address is on a network if the address ANDed with the netmask gives the network address

This is not on a nice byte boundary, so visually is harder for humans to work with using decimal `x.y.z.w` style notations

IP Address Subnetting

So `138.38.100.20` *is* on the subnet

host address	138.38.100.20	10001010 00100110 01100100 00010100
netmask	255.255.248.0	11111111 11111111 11111000 00000000
AND	138.38.96.0	10001010 00100110 01100000 00000000
network address	138.38.96.0	10001010 00100110 01100000 00000000

IP Address Subnetting

But `138.38.104.20` is *not* on the subnet

host address	138.38.104.20	10001010 00100110 01101000 00010100
netmask	255.255.248.0	11111111 11111111 11111000 00000000
AND	138.38.104.0	10001010 00100110 01101000 00000000
network address	138.38.96.0	10001010 00100110 01100000 00000000

IP Address Subnetting

`138.38` is split into many subnets of appropriate sizes for each Department, Centre or other sub-part of the University

Outside of `138.38` the subnetting is invisible so no changes to global routing tables are necessary if we rearrange our network

Subnets can be further subnetted for exactly the same reason

IP Address Subnetting

The subnet is described as "`138.38.96.0`, netmask `255.255.248.0`"

More commonly as "`138.38.96.0/21`", where 21 is the number of 1 bits in the netmask

You don't have to use the top *n* bits for a netmask, but it is overwhelmingly common to do so

The /*n* notation is only for a top-*n*-bit netmask

The "all 0s" and "all 1s" addresses now apply within the *subnet*: all 1's broadcasts to the subnet; and don't use all 0s

IP Address Exhaustion

So how does this relate to classful addressing?

Everybody wants a class B as C is too small and A is too large

Called the *Three Bears Problem*

There are no class Bs left: they have all been allocated

But, as the Internet grows, people want more addresses

IP Address Exhaustion

Can we split some class As?

Doable, but needs everyone to take care their software understands that those addresses are no longer class A

Most class A's have now been split and the subnets allocated to various institutions

IP Address Exhaustion

Can an institution simply use several class Cs?

Yes, but awkward as this leads to multiple networks, each needing separate routing

For example, having eight class C networks 194.24.0.0 to 194.24.7.0 would require all global routers' tables to have eight entries that all point to the same destination

And internally to the institution there are eight separate networks, too

IP Address Exhaustion

Class E has 286 million reserved addresses; can we use them?

Wouldn't last long; perhaps under a couple of years if allocated

More problematically, class E addresses are treated as illegal by much software, particularly on routers, so they are difficult to bring into play

(A recurrent problem with improving Internet protocols: a lot of software out there assumes the old way of doing things is the only way, and rejects any patterns or protocols it doesn't recognise)

IP Address Exhaustion

Some while ago it was recognised that the growth of the Internet meant that a new way of allocating addresses was needed

Three solutions are used:

- Change the way classes are defined and used
- Use private addresses with network address translation
- Increase the number of addresses available by changing the IP

We shall be looking at each of these

CIDR

Classless Interdomain Routing (CIDR) takes class C networks and joins them together in such a way that simplifies routing

Blocks of C addresses are allocated to regions, e.g.,

194.0.0.0-195.255.255.255	Europe
198.0.0.0-199.255.255.255	North America
200.0.0.0-201.255.255.255	Central and S America
202.0.0.0-203.255.255.255	Asia and the Pacific

CIDR

Starting with about 32 million addresses per region

This allows easy routing: anything 194 or 195 goes to Europe

Repeat the idea within each region: contiguous block of C networks are allocated to ISPs or organisations

Keeps simple routing within the region

Note that the software within routers does need to be updated to support this: but this has now been done everywhere

CIDR

E.g., `194.24.0.0` to `194.24.7.255`, normally written `194.24.0.0/21` or even `194.24/21`: exactly like subnetting

194.24.0.0	11000010 00011000 00000000 00000000
194.24.7.255	11000010 00011000 00000111 11111111
255.255.248.0	11111111 11111111 11111000 00000000

Any packet with address that has addr AND `255.255.248.0` = `194.24.0.0` should be routed to that ISP or organisation

A network of $2^{32-21} = 2^{11} = 2048$ addresses, i.e., 2046 hosts

CIDR

This is a very flexible and backwards-compatible scheme

End hosts do not need to know about CIDR

Classless networks can be subnetted

CIDR has allowed the continued growth of the Internet well beyond the original possible size by using addresses that would otherwise be wasted: allocated but not used

And we have repurposed class A and B networks similarly

CIDR

In fact, **classful networks are no longer used**: CIDR is the only way addresses are currently allocated

CIDR merges small networks into a larger one

Subnetting divides a large network into smaller ones

CIDR is sometimes called *supernetting*

Thus we have:

- Classful: implicit, fixed split of network/host
- Classless: explicit (netmask), variable split of network/host

CIDR

CIDR has been very successful, and has extended the life of the Internet significantly by providing a source of addresses from the previously underutilised classful ranges

Not enough. . .

Addresses

There are currently about 51 billion devices connected to the Internet (`www.statistica.com`; 2023)

But there are only about 4.3 billion usable IPv4 addresses

How is this possible?

NAT

This brings us to the second approach to address exhaustion

Some IP addresses are reserved for *private networks*, originally reserved to allow local experimentation:

- 10.0.0.0-10.255.255.255 (Class A)
- 172.16.0.0-172.31.255.255 (Class B)
- 192.168.0.0-192.168.255.255 (Class C)

One class A-size network, 16 class B and 256 class C-size networks are guaranteed never to be allocated for public use in the Internet

NAT

Routers on the public Internet will never forward packets with such addresses, and will simply drop them immediately

They are called *unroutable* addresses

NAT

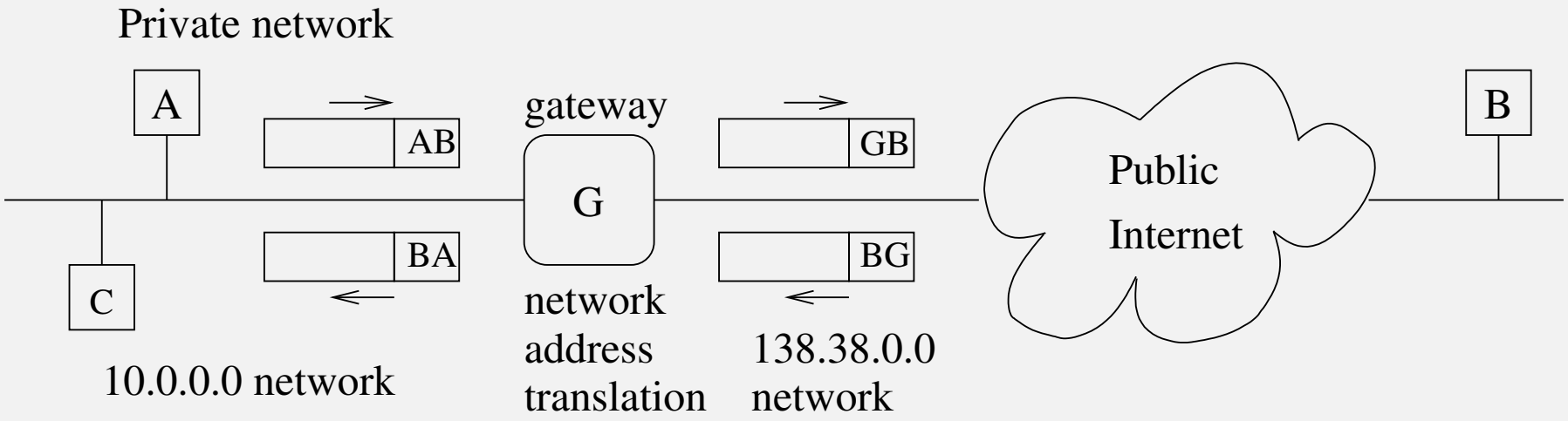
But such addresses can be used by anyone locally for any purpose: a common use is NAT

Network Address Translation (NAT) uses the malleability of packets to map many hosts onto a single address

A private network can be set up, using one of the above address ranges, e.g., `10/8`

A gateway host joins the private network to the public Internet, rewriting the addresses on packets as they go past

NAT



Network Address Translation

A packet from `10.0.1.1` (A) is sent to `212.58.226.33` (B); B is not on the local network so the packet is sent to the gateway; The gateway overwrites the source address with its own public address (G) and forwards the packet; The packet reaches B in the normal way; B replies with a packet with destination address G; The gateway recognises this packet as a reply to A

NAT

G needs to keep a record of connections from A to the world and recognise replies to outward travelling packets

C will want to do the same as A; so G must be able to distinguish replies to A from replies to C; even if both were communicating with B

And rewrite the replies to C with C's address

This is all doable in practice!

Explanation later, in the next layer

NAT

Exercise If both A and C are communicating with B, what are the addresses on their packets as they reach B? And on the replies as they reach G?

Exercise Compare with *bridging*, a similar idea but for very different reasons

NAT

As a fortunate side-effect, NAT provides some measure of protection to hosts on the private network from external attack

Machines on the public Internet (e.g., B) cannot initiate traffic to A as `10.0.1.1` is a private, unroutable address

No public router will forward a packet with such an address: it will simply drop it

External hosts will generally not even know what A's address is as they never get to see it

Even if a packet somehow gets to the gateway, the gateway will not know how to rewrite its address as this was not a reply to an outgoing packet; so it get dropped here, too

NAT

NAT has helped immensely to mitigate the address exhaustion problem

Previously, every host on a network would need a unique public IP address

The growth of the Internet at home, for example, would have sucked up addresses at a huge rate

But now all your home appliances can share just one public address

Exercise Count the number of network attached devices you have at home

NAT

Problems arise when the *data* in the packet contain IP addresses that, say, will be used to set up new connections. E.g., the *File Transfer Protocol* (FTP)

(Original) FTP would send an IP address to the server to indicate where to set up a new connection

In our example, this would be its private, unroutable address that the external server couldn't contact

Unless the gateway is intelligent enough to realise this is an FTP exchange, look inside the data and know where the IP addresses are to be found (in the application layer data) and rewrite them (in the application layer data) the addresses will remain untranslated and the protocol will fail

NAT

Not many protocols do this kind of thing these days, but each one of those that do must be treated specially by the NAT gateway

Note this is a problem due to a violation of layering in the protocol: IP layer information in the application layer

Exercise Read about FTP, *Universal Plug and Play* (UPnP) and the *Simple Service Discovery Protocol* (SSDP)

NAT

NAT is used widely as it is very effective

It allows you to have many machines but only use one public address

Many mobile phone companies are now using *carrier grade NAT* to supply IP connectivity to the millions of phones they manage

Carrier grade NAT: NAT done in the ISP rather than by the end-user

Exercise What IP address does your phone have for its mobile data connection (not its Wi-Fi connection)?

Exercise Read RFC6598 and about `100.64.0.0/10`

NAT

Without NAT, public IP addresses would have run out years ago

But there are costs to NAT

- Complexity in the gateway software
- Scalability problems in the gateway tracking large numbers of connections
- Bad interactions with some protocols
- Difficulty of making end-to-end connections when both ends are behind a NAT gateway (e.g., Skype, SIP)
- Loss of “an IP address identifies a host uniquely”: a problem for law enforcement

NAT

There is also the inability for external hosts to initiate connections to hosts behind NAT

So you can't run servers on hosts behind the NAT

But this invisibility is generally a good security feature

This can be worked around, though not neatly

Exercise Read about port forwarding (later)

Exercise Read about STUN

NAT

NAT is the reason the Internet did not grind to a halt many years ago through the lack of available addresses

Thus putting off the need for a proper solution to the problem

Some people still argue that there is no reason to do anything else than use more NAT

Even to the extent of using multi-level NAT (NAT within NAT)!

NAT

But even with CIDR and NAT, the entire range of usable IPv4 addresses has now been allocated

2011: IANA has distributed all its reserves of addresses to the Regional Internet Registries (RIRs)

2019: RIPE's allocation (covering Europe, Middle East and Central Asia) have run out

Old addresses that are no longer needed get recycled; there is even a black market in IP addresses!

We need a more radical solution

IPv6

The next approach to the IP address exhaustion problem is to change IP itself

The next version of the IP is IPv6 (occasionally called IPng for “IP next generation”)

Slowly growing in use, it will take a while to replace all of IPv4

128 bit addresses; CIDR-style allocation only

Exercise Find out about IPv5. And IPv0-IPv3

IPv6

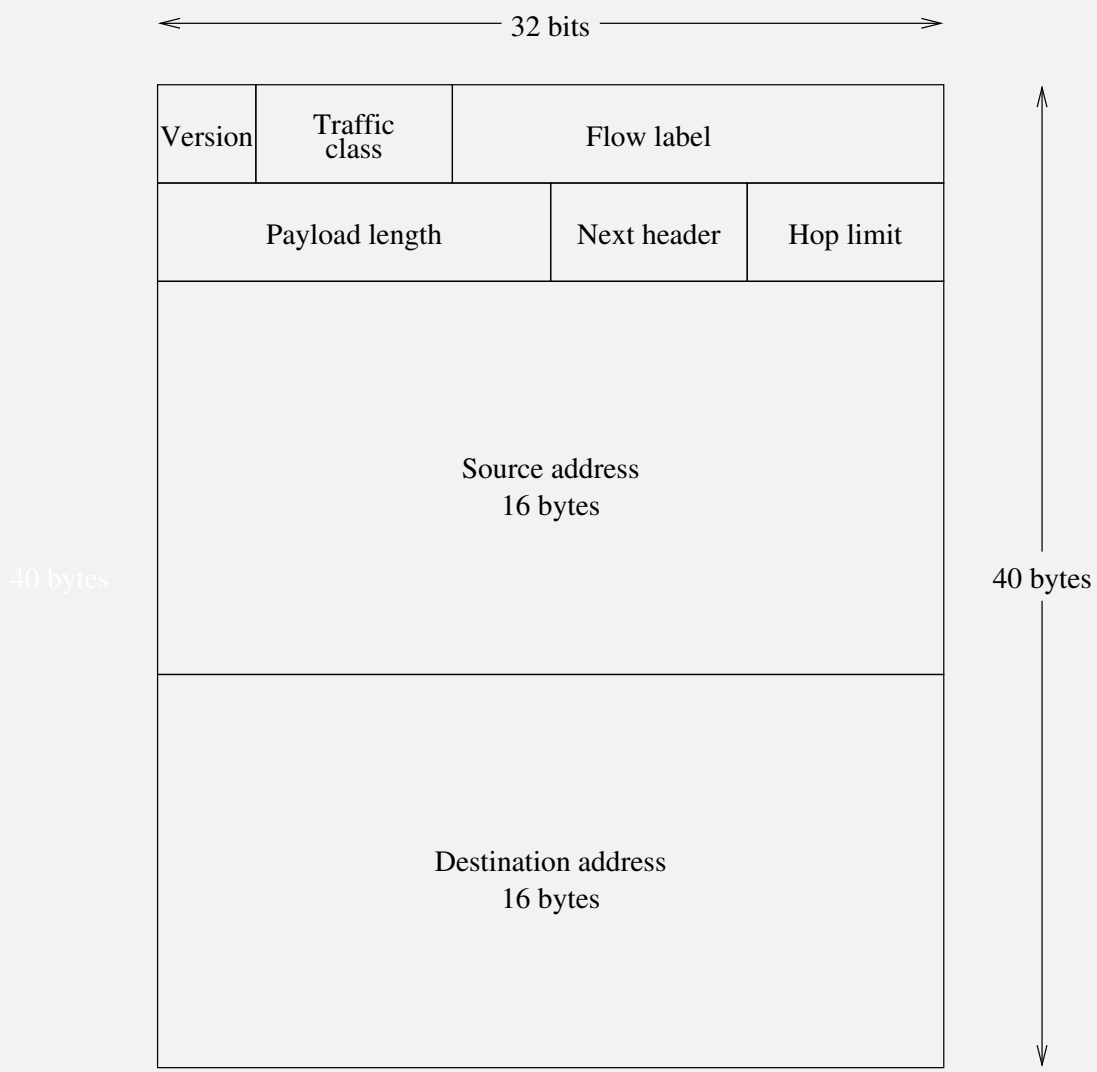
IPv6 was designed to

- have a larger address space
- reduce the size of router tables
- simplify the protocol so routers can process packets faster
- provide security and authentication
- pay proper attention to type of service (DS)

IPv6

- have better multicasting support
- have mobile hosts with fixed IP addresses
- allow room for evolution of the protocol
- permit IPv4 and IPv6 to coexist during the transition

IPv6



IPv6 Header

IPv6

- Version, 4 bits. The number 6. This is identical in position to IPv4 and can be used to distinguish packets in mixed-version environments. Additionally, in an Ethernet frame, IPv4 has protocol number 0800, while IPv6 is 86DD, but remember you might be using a different physical layer that does not give the type of its data
- Traffic class, 8 bits. Like TOS (DS) in v4
- Flow label, 20 bits. Allows routers to recognise related packets in a single flow and treat them identically (and so faster)

IPv6

- Payload length, 16 bits. The number of bytes following the fixed 40 byte header. Unlike v4, this not the packet length as it does not include the header in the count
- Next header, 8 bits. Like the protocol field in v4, but also allows for v6 optional header fields, if any
- Hop limit, 8 bits. The TTL field, renamed to make it clear how it is actually used

IPv6

- Source and destination addresses, 128 bits each.

Four times as long as v4 addresses

$2^{128} = 3 \times 10^{38}$ addresses, enough for an address for every molecule on the surface of the Earth

There are unicast, multicast and anycast addresses: details later

IPv6

Addresses are typically written in hex, with colon separators, e.g., `fe80:0000:0000:0000:21c:c0ff:fea3:99f4`

A `::` may appear once as a shorthand for a string of `0`s. As many as you need to make the address up to 128 bits

Thus the above address can be written
`fe80::21c:c0ff:fea3:99f4`

Remember this is notation for:
`1111111010000000 0000000000000000 0000000000000000`
`0000000000000000 0000001000011100 1100000011111111`
`1111111010100011 1001100111110100`

IPv6

The University of Bath has been allocated
`2001:0630:00e1::/48`

Meaning $128 - 48 = 80$ bits of address for hosts on the University network

$2^{80} = 1.2 \times 10^{24}$ addresses, which is about 280 trillion times the size of the whole current IPv4 Internet!

Exercise Check my arithmetic

Exercise Look up the IPv6 address of `facebook.com`

IPv6

Back to the other IPv6 header fields

There are no fragmentation fields

A router never fragments IPv6, but simply drops the packet and sends back a “packet too big” message to the source. The source can then send smaller packets

Processing within a router is therefore much simpler and packets can be sent onwards much faster

Every IPv6 host is required to do path MTU discovery

IPv6

The flow label helps identify packets within a single “flow”, i.e., a connection or session

Packets with the same flow label can be treated identically and so sent onwards faster by a router

In essence a session identifier

Exercise Reflect on this: aren’t sessions supposed to be done in a different layer?

IPv6

No header length field: the header is always 40 bytes

No fragmentation fields: no fragmentation in routers

No checksum field: there are checksums in other layers. The protocol designers thought that yet another checksum would not be helpful here. IP is not required to be reliable, anyway

Also we don't have to recompute a checksum in every router as the TTL decreases. Again, faster in routers

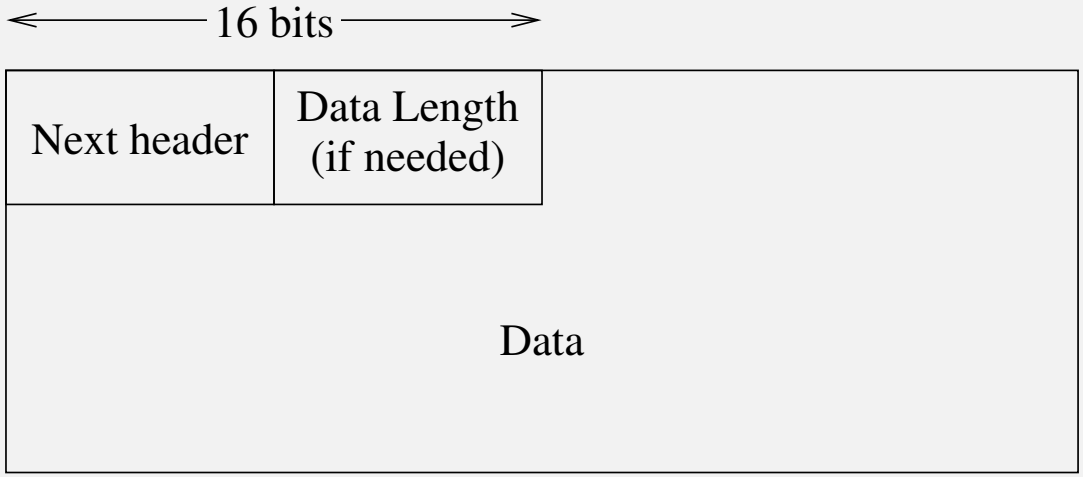
IPv6

v4 has 13 fixed fields; v6 has 8; much simpler for a router to process

v6 addresses are 4 times the length, but the header is only twice as long

IPv6

The *next header* field daisy-chains options, called *extension headers*, or gives the protocol (TCP, UDP, etc.) of the next layer



Option Header

Thus the only limit on the options is the total datagram limit

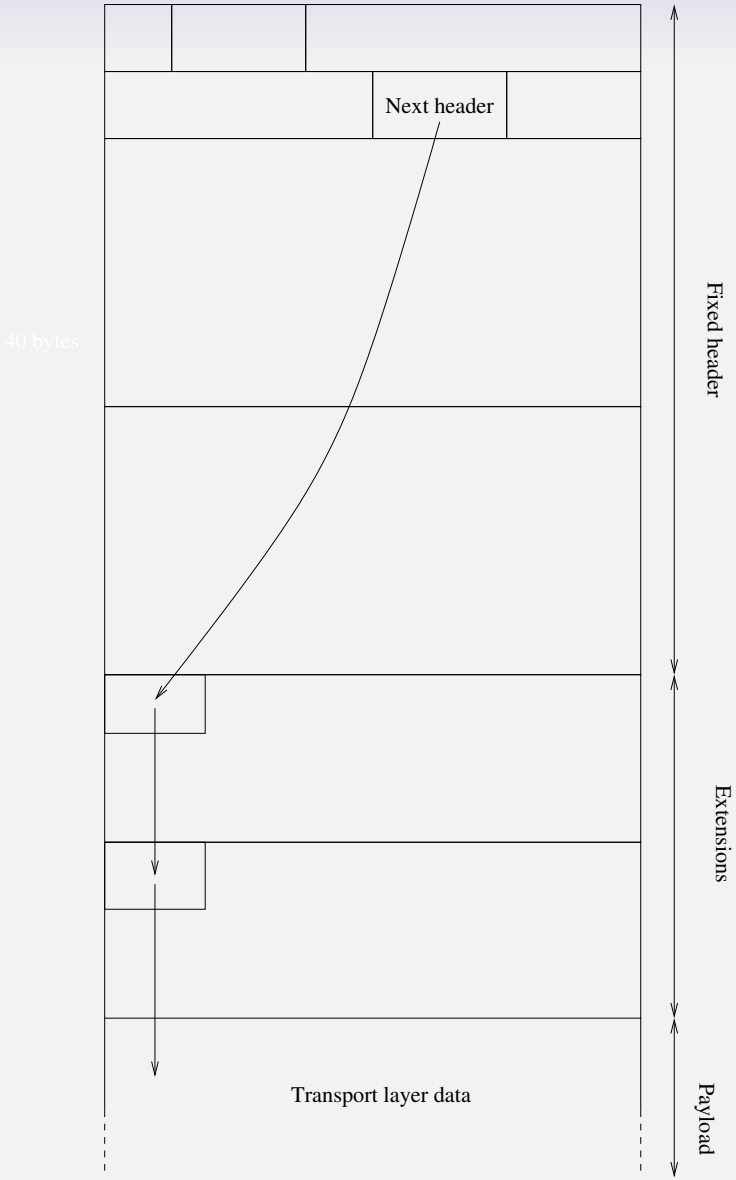
Furthermore, most options are not even looked at by routers: again to get faster processing in the routers

IPv6

Optional headers include:

- Routing options: c.f., loose source routing in IPv4
- Authentication
- Security
- Jumbograms: packets up to 4GB in length!
- And others

Note the type of the header option is given in the *previous* header option, or the main IPv6 header for the first option



IPv6 options daisychain

IPv6 Jumbograms

A note on IPv6 jumbograms

It needs, of course, the hardware, link and transport layers to support large packets

For example, Infiniband supports up to 4k frames, while a lot of modern Ethernet hardware seems to support 9216 byte frames

We'll see later that UDP and the handshake TCP MSS have only 16 bit length fields (64k bytes), so tweaks are needed there, too (RFC2675)

IPv6 Jumbograms

Jumbograms can only be used locally, e.g., within datacentres, as the outside world almost certainly won't support them!

Exercise Frame CRC algorithms were designed when frames were small. Read about the problems they have with jumbograms

IPv6

Transition to v6

IPv4 address allocations have run out, so we need to move to IPv6

But it is expensive to do so, as it needs application rewrites, as a lot of application software assumes IP addresses are 4 bytes long and thus fits in an integer on a typical computer

So many people (ISPs, websites etc.) are pretending the exhaustion problem does not exist

Even though the majority of modern routers and end hosts contain the necessary IP (and transport) level software support

IPv6

Transition to v6

We can’t turn off the Internet and replace v4 by v6 overnight

Though, by design, the two protocols can run side-by-side on the same networks

IPv6

Transition to v6

IPv6 was devised in 1996, but has yet to achieve mainstream use

As of October 2023 figures from `ipv6-test.com` say they see about 64% of world traffic is IPv6

About 56% of UK traffic is IPv6

Uruguay (top): 94%

Many countries are under 1%

IPv6

Transition to v6

Some large companies, e.g., Google, support IPv6 connections (as well as IPv4): they want to encourage the transition

But many ISPs don't as it requires extra work and support: so many home users can't use it

There have been a variety of transition mechanisms suggested, often based on NAT-like packet mangling

But they are all complicated and unsatisfactory, for the same reasons NAT is unsatisfactory

IPv6

Transition to v6

Exercise Read about NAT64 (RFC6146) and DNS64 (RFC6147) for connecting IPv6-only clients to IPv4 servers

Exercise Read about *IPv4 mapped addresses*, that allows server code that is purely IPv6, but accepts IPv4 client packets

Exercise Read about 464XLAT (RFC6877) for IPv4-only clients that translates IPv4 addresses to IPv6 addresses for transport and then back to IPv4 addresses for the destination server

IPv6

Transition to v6

In the near future IPv6 will need to be supported properly by everybody

Exercise Find out if your home ISP supports IPv6

Exercise RFC6177 suggests giving home users a /56 network. How many host addresses does this correspond to?

Addresses

We now take another look at IP addresses

In particular there are several types of address that can refer to more than one host at a time

Addresses

IPv4 has three types of address

- Unicast: an address refers to a single destination (ignoring NAT!). A “normal” address
- Broadcast: as in the link layer, a single packet goes to every host in the local network. But, now, the “network” is at the IP layer, so may comprise more than one link layer network
- Multicast: in between uni- and broadcast. A single packet goes to one or more hosts

Addresses

IPv6 adds

- Anycast: a packet goes to any **one** of a selection of servers, usually the “closest” in some sense

In fact, IPv6 also removes broadcast as its job can be done by multicast

So we need to look at four types of address

Unicast Addresses: v4 & v6

Unicast

- 1-to-1 data flow; one source, one destination
- Most current IP traffic is unicast

Broadcast Addresses: v4

Broadcast

- 1-to-many data flow; one source, “all” destinations
- Broadcast is simple: a single packet read by all hosts on the local network
- Reduces traffic on the local network as (for most link layers) we don’t have copies of mostly-identical packets, one for each destination, but just one packet that is read by every host
- Scales well (locally): it is independent of the number of destination hosts
- Don’t have to know how many destination hosts there are

Broadcast Addresses

Broadcasts are generally limited to the local network: otherwise the entire Internet would be permanently flooded

We have seen IPv4 broadcast addresses before: when the host part of the IP address is all 1s

E.g., `172.16.1.255` on the subnet `172.16.1/24`

We can also use `255.255.255.255` as a broadcast to the local network for when we don't yet know our network address

Broadcast Addresses

As mentioned, IPv6 does not support broadcast separately, so there are no IPv6 broadcast addresses per se

IPv6 uses multicast to achieve the same effect

Multicast Addresses: v4 & v6

Multicast

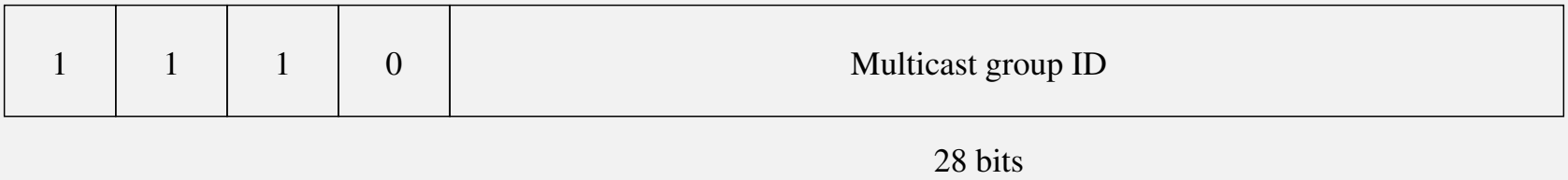
For sending a single packet to multiple hosts, not necessarily all hosts

E.g., for streaming radio we could send individual unicast packets to all listening hosts, but it would be much more efficient to send a single packet that the listening hosts receive and the non-listening hosts don't

Also, we can't use broadcast as broadcast is network-limited: listeners can be spread far and wide over multiple networks

Multicast Addresses: v4

One class of IPv4 addresses is reserved for multicast



Multicast addresses

In IPv4, class D (`224.0.0.0` to `239.255.255.255`) addresses are used for multicast

Multicast Addresses: v4

Multicast groups are formed from those hosts that wish to receive packets from a given source. e.g., a group to listen to BBC Radio 4

A multicast group id is a 28 bit number with no further structure: about 270 million possible groups

The set of hosts listening to a particular multicast address is also known as a *host group*

Host groups can cross multiple networks and there is no limit on the size of a group; and generally you can't know how big the group is

Multicast Addresses: v4

Some group addresses are preallocated by IANA: the *permanent host groups*

- `224.0.0.1`: all multicast aware hosts on this subnet (not all IPv4 hosts support multicast)
- `224.0.0.2`: all multicast routers on this subnet

Multicast Addresses: v4

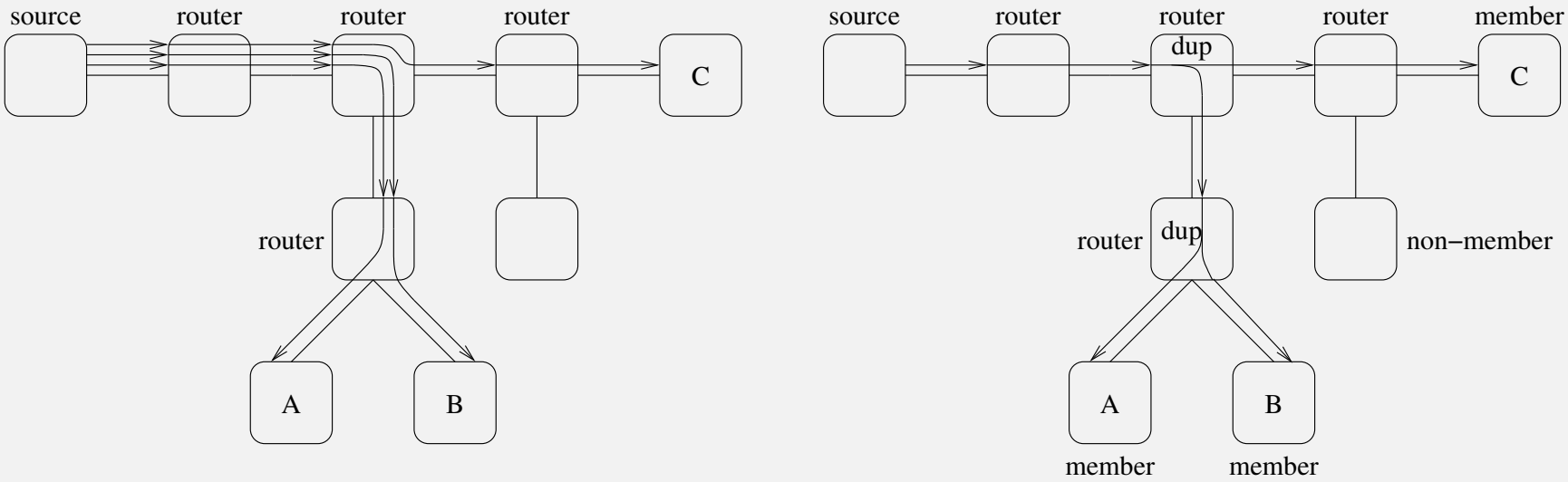
The process of joining and leaving groups is governed by the Internet Group Management Protocol (IGMP)

A host that wishes to join a multicast group provided by a server sends an IGMP message towards the server

The routers on the path to the server take note and so know to route multicast packets for this group towards the joining host

The server itself is not interested or involved in the IGMP message

Multicast



Unicast vs. Multicast

Multicast Addresses: v4

Similarly for a host leaving a group: a host is supposed to send an IGMP message towards the server that the routers can read and act upon

Extra complication arises as hosts may not (or can't if they crash) always send "group leave" messages

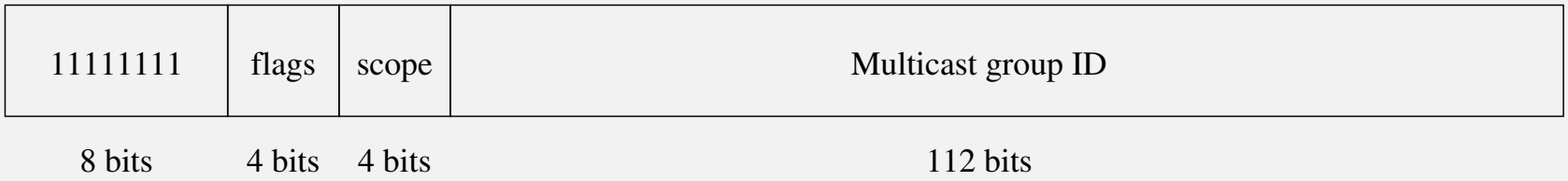
So there is more protocol to monitor and maintain groups using timeouts and maintenance messages

Exercise Read about this

Multicast Addresses: v6

While multicast was optional in IPv4, it is required in IPv6 (otherwise it would not have broadcast!)

IPv6 multicast is much as v4, but simplified



IPv6 multicast addresses

- Addresses start with hex FF
- Four bits of flags, including the T bit which means transient group (as opposed to a permanent IANA allocated group)
- Four bits of scope. Limit the range of this multicast to, e.g., the local network; the organisation; the country; worldwide

Addresses

Multicast

Exercise Read about how IPv4 uses the TTL to limit scope

Exercise Find out what IPv6 needs to do to broadcast to the local network

Addresses

Multicast

Multicast is not used as much as it should be

It is used in routing protocols (i.e., those protocols that help routers create their routing tables), but relatively little elsewhere in IPv4

Exercise Read about the *Simple Service Discovery Protocol* (SSDP)

Exercise And the *Multicast Domain Name System* (mDNS)

Addresses

Multicast

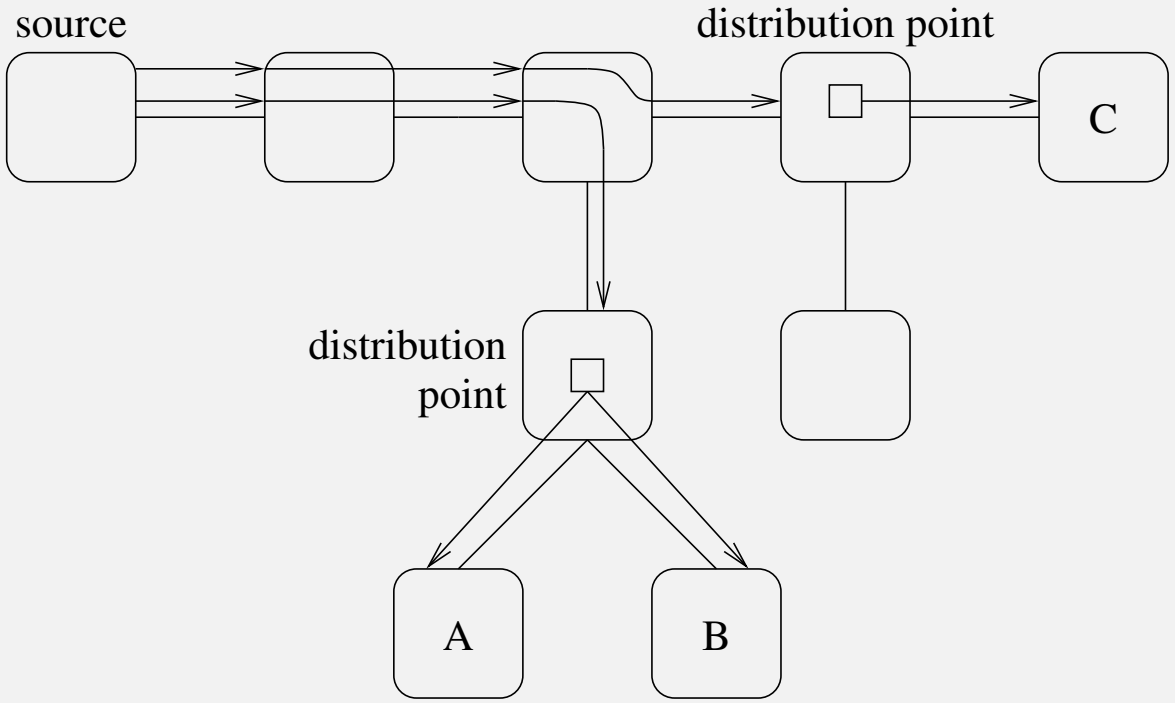
Multicast is hard to use for an on-demand system (e.g., BBC iPlayer, Netflix) as it requires everyone in the group to be receiving the same thing at the same time

While ideal for a live transmission, multicast does not work when everyone wants to watch things at different times

Most big streaming providers rely on having many local distribution points containing identical data, even for live streams

Addresses

Not Multicast



Content distribution points

The source supplies (relatively few) distribution points using unicast, which serve content directly using unicast

Exercise Read about *content delivery networks*

Addresses

Multicast

Furthermore, most providers use have to use unicast as multicast is not well supported in home systems

And routing companies want to avoid supporting multicast, claiming undue complexity to support it: each group needs extra state in every router the multicast traffic passes through, making scaling to the full Internet a problem

A router must keep a record of all multicast paths passing through it, so routers on popular paths (e.g., in internet exchanges) might need to keep a large amount of data

Addresses

Multicast

Multicast is used by some pay-tv services, but usually in the context of a closed and controllable system, e.g., a institutional intranet multicasting a seminar, or holding a multi-way video conference

Generally in the case where the same institution owns all the infrastructure from source to destinations

Exercise Read about BT TV

Addresses

Anycast: v6

Anycast

Anycast in IPv6 sends a single packet to a single destination chosen out of several possible destinations

For example, replicated Web servers: have many servers around the world with identical content and the same anycast address. A browser would get pages from the closest server, thus sharing load

The reply would be unicast

Addresses: v6

Anycast

Only works well with connectionless transport protocols (see later) as multiple requests might go to different servers: this doesn't fit well with connection-oriented protocols

Address format?

Any unicast address that happens to be assigned to more than one server. It is up to the routers to figure this out

Addresses: v6

There are anycast groups, much as multicast groups and a join/leave protocol

Notice the symmetry: muticast is groups of clients, while anycast is groups of servers

Anycast has plenty of potential, but we need to be using IPv6 to get it properly, though some people do support it in IPv4

Exercise `1.1.1.1` is an anycast address. Investigate

Addresses

How does a host get an IP address?

An Ethernet address is burned into the hardware, so there's no problem there

IP addresses are software addresses, so they must be set up somehow

The simplest way is for the host simply to be configured to have that address, stored in a configuration file on the host somewhere

An administrator takes into account certain criteria, e.g., network or subnetwork addresses, and gives the machine a currently unused address

But it is not always feasible to do this

DHCP

- Not all machines have administrators, e.g., home PCs
- Some administrators are not sufficiently competent to allocate addresses correctly, e.g., home PCs
- Some installations have too many machines to get around and configure them all, e.g., in the library
- Some installations have machines that come and go all the time, e.g., laptops in the library

DHCP

The best way of approaching this fairly simple but time consuming task is to use a computer

The *Dynamic Host Configuration Protocol* (DHCP) does just this

When a machine needs an IP address it can use DHCP to get one

DHCP

When a host boots and finds it needs an IP address, it makes a DHCP *broadcast*

This is like the host saying “can anyone give me an IP address?” to the network

In contrast to ARP, with DHCP there is usually just one (occasionally more as backup) host that is configured to respond to DHCP requests, as allocation of addresses must be centrally managed to avoid duplication

Again in contrast with ARP, this request is a network layer local broadcast, actually using an IP packet with address `255.255.255.255`

DHCP

A DHCP server (i.e., the DHCP program running on some host) listens for such requests; it will choose a currently unused IP address and send it back to the requesting client

The value might be chosen by the server according to some defined policy, or (more usually) the next free address taken from a list of currently unused addresses

The client gets this reply and reads its IP address which it can then use to configure itself

DHCP

In outline:

1. the client broadcasts “Who out there is willing to do DHCP with me?” (a `DHCPDISCOVER` message)
2. one or more servers broadcast a reply. “I will. Here’s an address” (`DHCPOFFER`)
3. the client picks a server and broadcasts “Can I have that address, please?” (`DHCPREQUEST`)
4. the chosen server broadcasts “OK, it’s yours” (`DHCPACK`)
5. the client sets its IP address

DHCP

Exercise Find out the details, e.g., what happens if a packet gets lost? For example, the `DHCPACK`

DHCP

DHCP runs over (UDP over) IP, so DHCP packets must have IP source and destination addresses

But the client doesn't yet know its own IP address, or any server's address, but it must fill in the IP address fields with something

Source: `0.0.0.0`. This is what we are trying to find

Destination: `255.255.255.255`. A local network broadcast

Exercise So what would the link layer address be?

DHCP

Packets returning from the DHCP server will have the server's IP address as source, and the broadcast `255.255.255.255` as destination

Again, the client doesn't yet have an IP address, so we have to resort to a broadcast to everybody

This is extra work for all hosts on the network (reading then ignoring the DHCP reply packets), but DHCP exchanges are relatively rare so it's not so bothersome

There are security implications though...

There is an identification field in the `DHCPOFFER` that allows a host to recognise a reply is for itself and not mistakenly take an offer for some other host that is doing DHCP at the same time

DHCP

DHCP

A DHCP server has a pool of available addresses that it can assign to hosts as they need them

When a host leaves the network, it should send a `DHCPRELEASE` to the DHCP server

Thus releasing its IP address to be reused for another host

But not all clients are well behaved, or might have crashed before sending a release

DHCP

DHCP

To fix this, DHCP gives a *lease time* on an address

The address is usable by the requesting host for this period of time

If the lease expires the host can request a *renewal* of the lease from the server

Which will then grant a further lease on the address

The renewal request and reply can be a normal unicast (non-broadcast) interchange, as the client already has an IP address

DHCP

If a host leaves the network or crashes, a renewal request will not be forthcoming

Thus the server can know, when the lease has expired, that the allocated IP address is no longer needed, and can be put back into the pool

(There are many protocols like this, that need a timeout to catch something bad happening)

DHCP

How long is the lease time?

This is configurable by the DHCP server's administrator

A short period is used when there is a fast turnover of machines (e.g., laptops in the library)

A long period, up to infinity, is used for more permanent machines, e.g., desktops

The administrator of the DHCP server needs to pick suitable values

Exercise What is the lease time from your access point on your home network?

DHCP

Besides addresses, DHCP can supply

- IP address
- netmask
- gateway
- name servers
- lease times
- print servers
- boot servers
- mail servers
- host name
- web servers
- and so on

DHCP

But usually just

- IP address
- netmask
- gateway
- name servers (for DNS, see later)

which is the minimum needed to get a host up and running and talking to the wider Internet

And the lease time

DHCP

After getting a new address, a client might broadcast an ARP reply containing its new address

This unrequested *gratuitous ARP* informs other hosts on the network of the new address association so they can update their ARP caches, e.g., invalidating an old association with this IP address

DHCP

Thus, DHCP solves the address, gateway and netmask (and other) configuration problem

But there is a wider issue we've alluded to several times that we must now discuss

Internet/Network Layer

Many times, in many circumstances, when things go wrong, we have said things like “blah blah and send an error message back”

For example, in MTU discovery we had “drop the packet and send an error message back”

Or when a TTL drops to zero, we had “drop the packet and send an error message back”

Internet/Network Layer

So how is such a message sent?

We only have packets, so the message must be in a packet

Just another IP datagram, with particular contents

An *Internet Control Message Protocol* (ICMP) packet

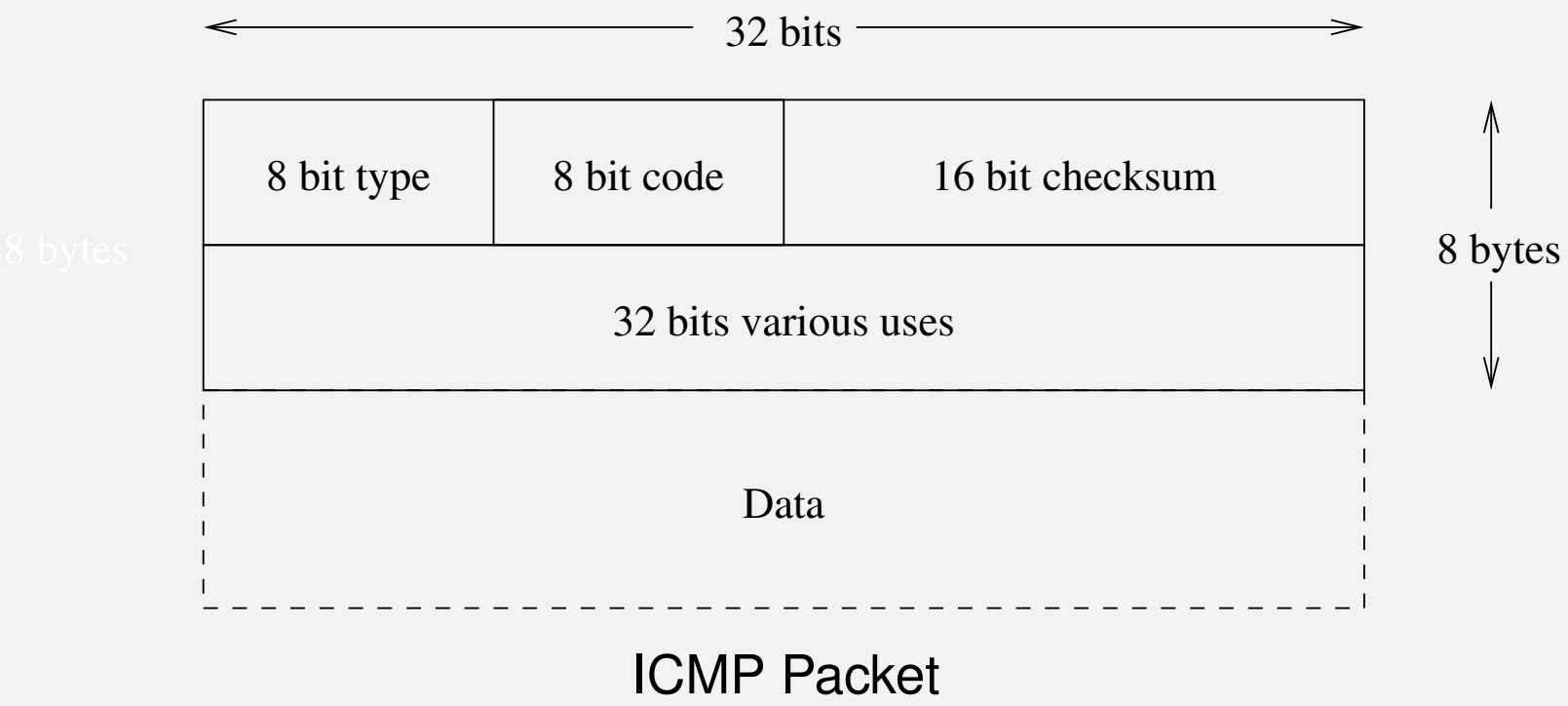
ICMP

ICMP is used for general control of the Internet, in particular errors

ICMP packets are contained within IP packets, but are considered to be part of the network layer

Thus the data field in an IP datagram might contain transport layer stuff, or it might contain network layer stuff

ICMP



ICMP Packet

ICMP

- Type: kind of message, e.g., “TTL expired”, “destination unreachable”, “fragmentation needed but DF set”
- Code: additional information, e.g., “destination unreachable” has “network unreachable” and “host unreachable” codes
- Checksum
- A fixed size field that has varying purposes for different types
- A general data field, if needed

ICMP

Thus ICMP packets of various types are used to indicate the different kinds of error message

For example, when a TTL on a packet decrements to zero, the router drops the packet, creates an ICMP “TTL expired” packet and sends it back to the source address, as given in the dropped packet

This message (in an IP packet) will have IP source address of the router; and destination address the source of the problem packet

ICMP

But, remember, ICMP packets are IP packets and so can be lost, delayed, duplicated or otherwise corrupted

And so ICMP errors can be generated for ICMP packets, with certain reservations

ICMP messages are classed as either a *query* or an *error*

E.g., ICMP “echo request” (ping) is a query, but “TTL expired” is an error

ICMP

ICMP errors are not generated for

- ICMP errors (e.g., TTL expires on a ICMP packet)
- a packet whose destination is a broadcast or multicast
- a packet whose source is a broadcast or multicast
- a packet whose link-layer address is a broadcast
- any fragment other than the first

This is to prevent broadcast storms, where a single error is multiplied up into many ICMP packets

Non-initial IP fragments don't contain enough identifying information for the OS to do anything useful with them, so don't bother with them (**Exercise** How do you know if you have an initial fragment?)

ICMP

Type	Err	Code
ECHOREPLY		reply from a ping
DEST_UNREACH	e	network unreachable
	e	host unreachable
	e	port unreachable
	e	fragmentation wanted but DF set
REDIRECT	e	routing redirect for network
	e	routing redirect for host
ECHO		ping
TIME_EXCEEDED	e	TTL reached 0
	e	fragment reassembly time exceeded

Messages marked “e” are errors. There are many other types and codes, but the above are the most common in practice.

ICMP

Ping

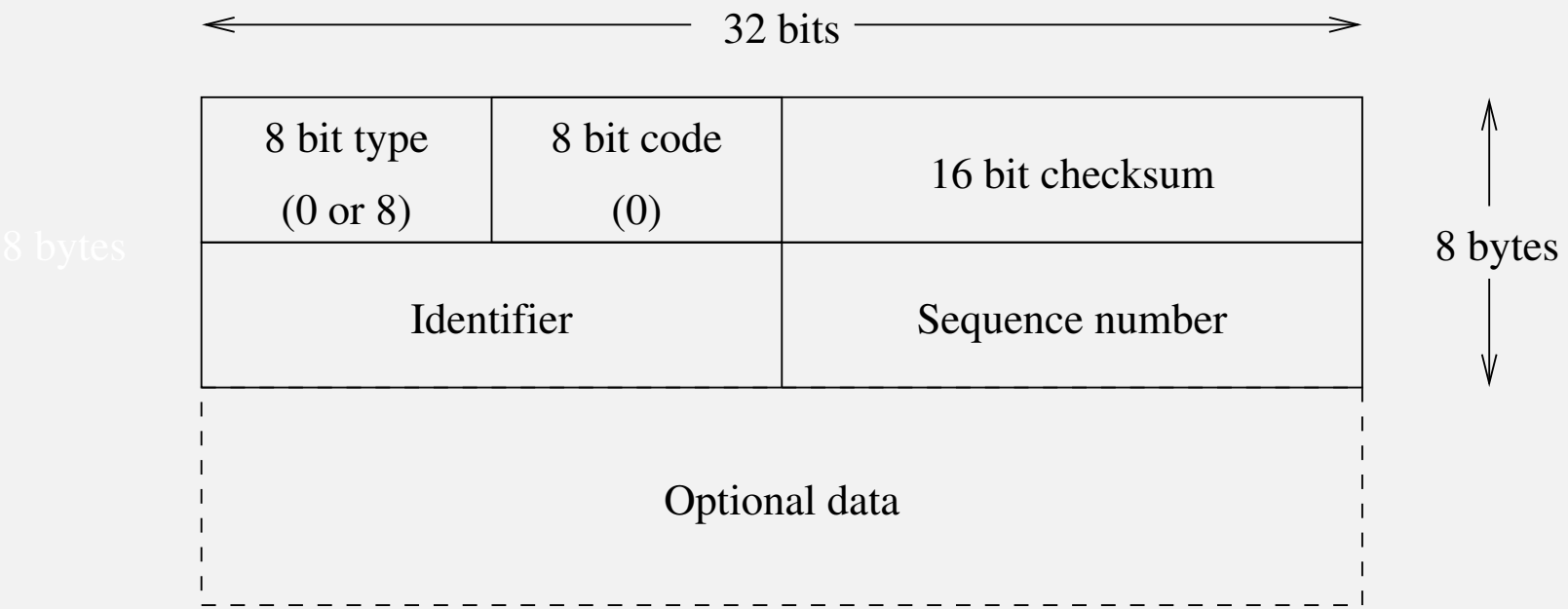
ICMP has many other uses

For example, we can discover if a machine is up and running using ICMP *ping*

A program, usually called `ping`, sends an ICMP “echo request” (also usually called a “ping”) packet, waits a second, then repeats

ICMP

Ping



ICMP ping headers

- ICMP type 0, code 0, with some random identifier
- A functioning host OS that gets a ping should return a "echo reply"
- This has ICMP type 8, code 0, and a copy of the identifier, sequence and data

ICMP

Ping

- The identifier field allows the originator OS to match up replies with requests
- The sequence starts at 0 and increases by 1 for each ping sent

This allows us to spot lost, duplicated or reordered packets

ICMP

Ping

```
% ping www.yahoo.co.uk
PING homerc.europe.yahoo.com: 56 data bytes
64 bytes from rc3.europe.yahoo.com (194.237.109.72): icmp_seq=0. time=160. ms
64 bytes from rc3.europe.yahoo.com (194.237.109.72): icmp_seq=1. time=154. ms
64 bytes from rc3.europe.yahoo.com (194.237.109.72): icmp_seq=2. time=176. ms
64 bytes from rc3.europe.yahoo.com (194.237.109.72): icmp_seq=3. time=159. ms
64 bytes from rc3.europe.yahoo.com (194.237.109.72): icmp_seq=4. time=161. ms
^C
----homerc.europe.yahoo.com PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 154/162/176
```

The `ping` command also keeps track of *round trip time* (RTT), the time between sending a request and getting the corresponding reply

Note lots of variance in the RTT: this is typical

ICMP

Ping

Some versions of ping can enable the IP header option record route: this makes IP save the address of each intermediate router in the header

But, as noted earlier, there can only be 60 bytes of options in IPv4, giving space for up to 9 addresses (with the overheads of the option header and other bits and pieces), so only 9 addresses are recorded

ICMP

Ping

```
% ping -R www.bbc.co.uk
PING www.bbc.net.uk (212.58.244.70) 56(124) bytes of data.
64 bytes from bbc-vip115.telhc.bbc.co.uk (212.58.244.70): icmp_seq=1 ttl=52
 time=89.0 ms
RR:     rjb.cs.bath.ac.uk (172.16.2.1)
        fire.cs.bath.ac.uk (138.38.108.253)
        swan-fwsm.bath.ac.uk (138.38.1.46)
        university-of-bath.ja.net (146.97.144.38)
        xe-0-0-0.bathbc-rbr1.ja.net (146.97.67.46)
        xe-1-0-0.brisub-rbr1.ja.net (146.97.67.33)
        swr.londpg-sbr1.ja.net (146.97.37.202)
        ae29.londpg-sbr1.ja.net (146.97.33.2)
        ae0.londhx-sbr1.ja.net (146.97.35.105)

64 bytes from bbc-vip115.telhc.bbc.co.uk (212.58.244.70): icmp_seq=2 ttl=52
 time=25.7 ms        (same route)
^C
--- www.bbc.net.uk ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 25.734/57.370/89.006/31.636 ms
```

ICMP

Traceroute

There are lots of routes of more than 9 hops, so using ping to discover a route is limited; besides many routers ignore or discard this option

The `traceroute` program is a clever way to find routes by deliberately generating errors and looking at the ICMP messages that result

It sends a packet to the intended destination, but with an artificially small time-to-live

ICMP

Traceroute

When the TTL drops to zero on a hop, the packet is dropped and an ICMP “TTL exceeded” is returned by the router that dropped it

As the source address on this ICMP error is the router’s, this tells us where the packet had got to

Repeat for increasing values of TTL to get the entire route

```
% traceroute mary.bath.ac.uk
traceroute to mary.bath.ac.uk (138.38.32.14), 30 hops max, 46 byte packets
 1  136.159.7.1 (136.159.7.1)  0.779 ms  1.131 ms  0.642 ms
 2  136.159.28.1 (136.159.28.1)  1.369 ms  0.910 ms  1.489 ms
 3  136.159.30.1 (136.159.30.1)  2.339 ms  1.937 ms  0.988 ms
 4  136.159.251.2 (136.159.251.2)  1.458 ms  1.071 ms  1.831 ms
 5  192.168.47.1 (192.168.47.1)  1.434 ms  1.554 ms  1.008 ms
 6  192.168.3.25 (192.168.3.25)  29.192 ms  30.094 ms  25.374 ms
 7  REGIONAL2.tac.net (205.233.111.67)  25.413 ms  33.002 ms  32.677 ms
 8  * * *
 9  * 117.ATM3-0.XR2.CHI6.ALTER.NET (146.188.209.182)  82.403 ms  58.747 ms
10  190.ATM11-0-0.GW4.CHI6.ALTER.NET (146.188.209.149)  56.376 ms  67.898 ms  7
11  if-4-0-1-1.bb1.Chicago2.Teleglobe.net (207.45.193.9)  66.853 ms  46.089 ms
12  if-0-0.core1.Chicago3.Teleglobe.net (207.45.222.213)  48.817 ms *  75.093 m
13  if-8-1.core1.NewYork.Teleglobe.net (207.45.222.209)  106.198 ms  94.249 ms
14  ix-5-3.core1.NewYork.Teleglobe.net (207.45.202.30)  75.286 ms  89.873 ms  9
15  us-gw.ja.net (193.62.157.13)  143.686 ms  159.212 ms  166.020 ms
16  external-gw.ja.net (193.63.94.40)  172.803 ms  189.216 ms  191.260 ms
17  external-gw.bristol-core.ja.net (146.97.252.58)  206.403 ms  185.438 ms  19
18  bristol.bweman.site.ja.net (146.97.252.102)  196.685 ms  206.221 ms  183.76
19  man-gw-2.bwe.net.uk (194.82.125.210)  197.968 ms *  174.809 ms
20  bath-gw-1.bwe.net.uk (194.82.125.198)  209.307 ms  221.512 ms  199.168 ms
21  * * *
22  mary.bath.ac.uk (138.38.32.14)  250.670 ms *  186.400 ms
```

ICMP

Traceroute

The `traceroute` command sends *three* probes for each stage so we can see time variations

Hop 8: no error packet was received for this TTL. There are many possible reasons, e.g., on a long route it is possible the router is setting an initial TTL on the reply that is too small to reach us

An increasingly common possibility is that the router refuses to send ICMP errors for TTL exceeded in a weak attempt at security

ICMP

Traceroute

A $*$ before the name means the name lookup took so long traceroute decided to stop waiting and carry on. The name subsequently turned up

Sometimes the same line is repeated: this is because some routers forward packets with TTL 0. This is a bug

There are many bugs out there in the real world!

Exercise Traceroute usually sends out UDP packets as probes, while some implementations use ICMP pings, while others use TCP SYNs. Find out why

ICMP

Traceroute

ICMP errors are required to contain the IP header and **at least 8 bytes of the original data** in the packet that caused the problem

This is so the OS in the source machine can match up the ICMP packet with the original packet and relay the error message back to the appropriate original application

There may be several applications running, sending packets, and getting ICMPs back

Eight bytes contains the interesting parts of the next layer headers (in particular the ports of UDP and TCP) and this will be enough to identify which outgoing packet this is a reply to

ICMPv6

ICMPv6 in IPv6 plays a similar, but expanded role

For example, the ICMPv6 *Neighbour Discovery Protocol* also does the job of ARP

With a *Neighbour Solicitation* request and *Neighbour Advertisement* reply

Exercise Read about this and compare with ARP

Routing

We now look at one of the fundamental aspects of IP: routing

A packet does not know how to get to its destination

It must rely on the routers to send it in the right direction

So how do the routers do that?

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

There are many relevant criteria:

- The smallest number of hops
- The fastest: some links might be faster than others, e.g., undersea cable vs. satellite
- The cheapest: transit is not free!
- The most reliable
- And so on (c.f., the TOS field in IP)

Routing

There is also *policy based routing*, where non-technical issues must be taken into account

You may wish to restrict where your traffic passes through

At one point, there was a law in Canada that said all traffic that starts and ends in Canada must never leave Canada. Even if it would be cheaper and faster to go via the USA, say

Routing

We normally think of two classes of routing:

- Local routing within an organisation, requiring an *interior gateway protocol* (IGP)
- Non-local routing between organisations, requiring an *exterior gateway protocol* (EGP)

Very different requirements, with exterior protocols mostly driven by politics and economics

Host Routing

We should revisit small routing tables:

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
213.121.147.69  *               255.255.255.255 UH    0      0        0 ppp0
172.18.0.0      *               255.255.0.0     U     0      0        0 eth0
172.17.0.0      *               255.255.0.0     U     0      0        0 eth1
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         213.121.147.69  0.0.0.0         UG    0      0        0 ppp0
```

The destination address on a packet is ANDed with each netmask (`Genmask`) in turn: if the result is equal to the `Destination`, route via the given interface

Use the first match moving from the longest mask to the shortest: top to bottom in this table

Host Routing

Flags:

- U: the interface is up (i.e., working)
- G: the route is to a gateway/router. Otherwise the destination is on the local network
- H: the route is to a host. The destination address is a single host, not a network
- D: this entry was created by an ICMP *redirect*
- M: this entry was modified by an ICMP *redirect*

Host Routing

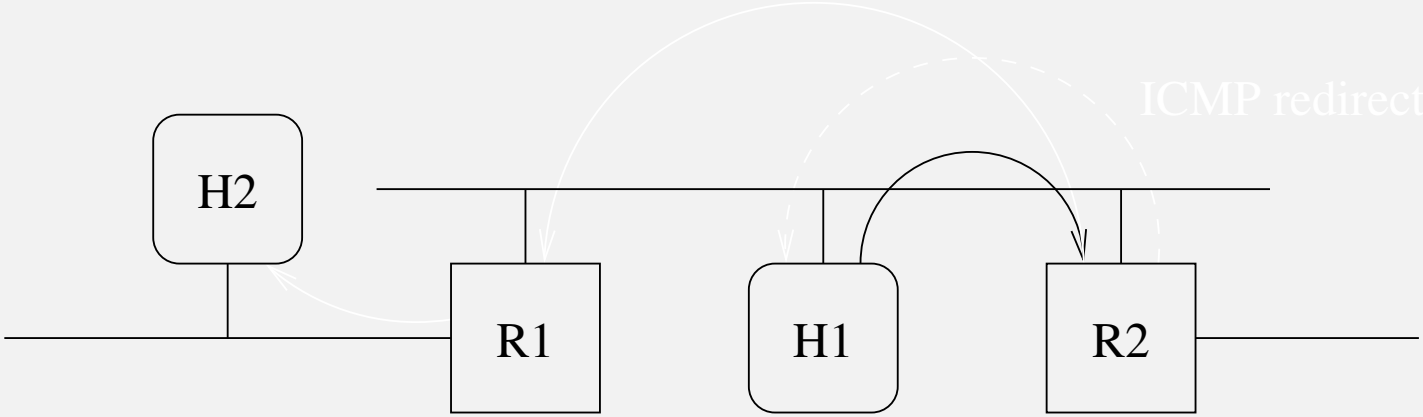
A *static route* is one added by hand, e.g., the `ip route add` command in Linux

Routing tables on most non-routers are trivial and set up "manually" by the operating system at boot time, often with the use of DHCP

In this context, DHCP is regarded as "setting by hand"

However, sometimes routing tables are not perfectly set up

Host Routing



ICMP Redirect

H1 wants to send to H2 but H1's routing table tells it to route via R2; When the packet reaches R2, R2 sees it should be routed out on the interface it came in on: so R2 knows H1's table needs improving; R2 forwards the packet to R1 and sends an ICMP *redirect* to H1; H1 gets the redirect and uses it to update its routing table. The route will be marked D or M; Next time H1 will be able to route directly to R1.

ICMP redirect
H2
R1
H1
R2
ICMP Redirect

Routing

ICMP can modify routing tables in a small way, but is not the main way routes are set up in big routers

We could get administrators to set up the tables

But better is to get the routers to do it themselves

Dynamic routing is the passing of routing information between routers

Dynamic Routing

There are many dynamic routing protocols:

- Routing Information Protocol (RIP)
- Open Shortest Path First (OSPF)
- Border Gateway Protocol (BGP)
- Exterior Gateway Protocol (EGP)
- And so on

Each protocol is suited to a certain purpose, no single protocol fits all

Dynamic Routing

We start at routing the top level, namely the Internet. Local networks have different requirements

The Internet is managed as a collection of *Autonomous Systems* (AS), each administered by a single entity, e.g., a University or company

Between ASs run *exterior gateway protocols* (EGP), currently BGP and formerly EGP (now obsolete)

Each AS chooses a suitable routing protocol to direct packets *within* itself: these might be interior gateway protocols, e.g., RIP and OSPF. Large institutions might even run BGP internally and have their own internal ASs

BGP

An AS is denoted by a 32 bit integer

There are currently (2023) over 115000 ASs

Top-level routers will need an entry in their tables for each AS

BGP

The University of Bath is within AS786, JANET

All of JANET is one big AS: routing within JANET is an internal issue

In fact JANET runs BGP internally. Bath has internal AS64857 within JANET

BGP

BGP allows policy based routing: it's not just the shortest or fastest path that it chooses

It is a *distance-vector protocol*

Routing

There are two main ways of finding routes:

- distance-vector protocols
- link-state protocols

- distance-vector protocols
  - distance-vector
  - path-vector
- link-state protocols

And distance-vector is usually sub-divided into distance-vector and path-vector

Routing

Distance-vector gathers collections (vectors) of hop counts (distances) from its neighbouring routers to selected destinations. From this it computes its own vector of distances

RIP is an example of a distance-vector protocol, occasionally used in smaller networks

In contrast, *link-state* gathers graphs of connectivity from all the routers (or some subset) and uses this to compute its own map. OSPF is an example

Distance-vector is simple, but has problems

Link-state is more complex, but has advantages

Routing

In either case routers periodically send all or parts of their view of the world to their neighbours

Some protocols use broadcast, some multicast

A message would be “My view of the network is this. . .” in the case of link-state

Or “I know a route to this destination using this number of hops” in the case of distance vector

Routing

The routers can then update their tables in light of this

For example, in distance vector:

- suppose a router knows a route (i.e., the next hop router) for a given destination of a given number n of hops
- it receives a message from another neighbour that includes a route of m hops to that destination
- if $m + 1 < n$ it can update its route to now go through that neighbour, as that is a shorter (fewer hops) route

Routing

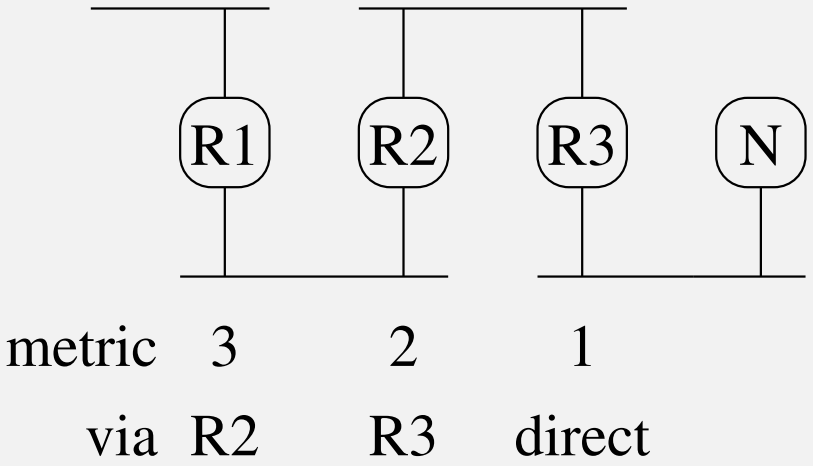
This is simple to implement and run, but such protocols have a *slow convergence* problem

This means that if the network changes (e.g., a link is broken, or a new link is made) it takes many interchanges of information for the routers to adjust to the new routes

And this can manifest in bad ways

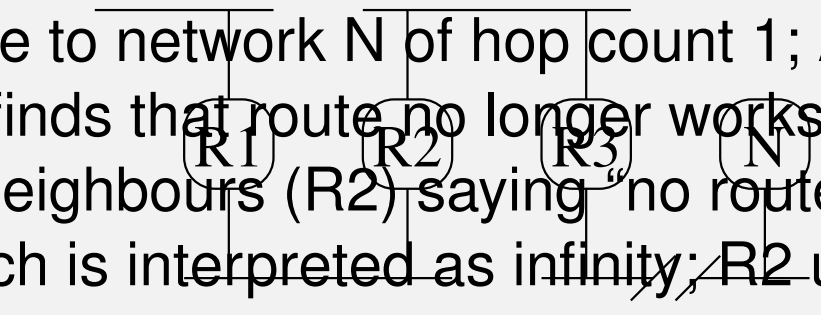
Routing

Slow Convergence



Route count 1

R3 knows a route to network N of hop count 1; After a break in the network R3 finds that route no longer works; So it sends a message to its neighbours (R2) saying "no route to N". It uses a count of 16, which is interpreted as infinity; R2 updates its routing table; But R2 also gets a periodic update message from R1 saying "route of 3 hops"; So R2 now thinks the best route is via R1, 4 hops; And when R2 sends its periodic update



Routing

Meanwhile real data packets are bouncing forwards and back between the routers

The local information that distance vector provides is not enough

RIP uses distance vector and this is a real problem for it

So RIP should only used on small networks that are fairly stable

Link state protocols, e.g., OSPF, converge faster, but need more complicated graph traversal algorithms to determine best routes

BGP

BGP is a *path vector* variation of distance vector: this includes the path (multiple hops) to the destination, which can be used to spot the loops that lead to count-to-infinity

ASs do not change very much so slow convergence is not such a big problem anyway

Exercise Read about path vector systems

BGP

BGP does have other problems, particularly authentication

Through accident or malice it is easy to trick BGP

For example, it would be relatively easy to get BGP to transit data through an evil third party

Also, see the problem with the route to Youtube, earlier

BGP

Exercise Read about the 2018 hack on the cryptocurrency website `MyEtherWallet.com` that started by subverting BGP to send DNS traffic to a rogue server

Exercise Read about the BGP problem of April 2021, where Vodafone Idea (AS55410) published bad routes

Exercise Read about the proposed *Resource Public Key Infrastructure* (RPKI), RFC6810

Exercise Read about the *Mutually Agreed Norms for Routing Security* (MANRS) initiative for ISPs and routing exchange operators

BGP

Exercise Read about RIP

Exercise Read about Dijkstra's algorithm for finding shortest paths in a graph; and OSPF which uses this algorithm

Transport Layer

We now move up a layer: the Transport Layer

The Internet Protocol has three main protocols that run on top of IP: two are for data, one for control

The data protocols are complementary

- one is fast, unreliable, connectionless: UDP
- the other is more sophisticated, reliable and connection-oriented: TCP

The control protocol, ICMP, we have already seen and is usually considered as part of the network layer

Other data protocols exist in this layer, but TCP and UDP are currently the important ones

Transport Layer

Ports

Both UDP and TCP use the concept of *ports*

On a single server machine there can be many programs running, web, email, and so on: how does a client indicate which service it wants from the server?

And when a reply packet arrives back at a client, how does the OS know which of the many processes running on the client that packet should be delivered to?

This is done by ports

A port is just a 16 bit integer: 1-65535

Transport Layer

Ports

Every TCP and UDP connection has a *source port* and a *destination port*

When a service starts
— i.e., a program that will deal with the service starts —
it *listens* on a port
— i.e., it informs the operating system that it wishes to receive data from packets directed to that port number

E.g., an email server may indicate it wants packets addressed to TCP port 25; a browser would listen on port 80 (and 443)

Transport Layer

Ports

The OS checks that port is not already being used by another program, and subsequently ensures packets with that destination port are sent to that service program

So when a TCP packet with destination port 25 arrives its data will be given to the email program

An analogy: a host as a block of flats. To address a letter to a specific person you need both a building address (IP address) and a flat number (port)

Transport Layer

Ports

TCP and UDP ports are entirely separate: one service can be listening for a TCP connection on a port and another service for UDP on the same port number

The OS can distinguish the two as they are port within different protocols

TCP and UDP are completely separate and do not interact at all (at the transport level)

Transport Layer

Ports

Certain *well-known* ports are associated certain services

- web server on port 80 (or 443 for a secure version)
- email server on port 25
- FTP on port 21
- Microsoft SQL server on 1433
- hundreds of others. See `/etc/services` and RFC6335

A range of ports are reserved for privileged (root/administrator) programs; most are available to any program that wants to use them

Typically, port numbers under 1024 are reserved for privileged programs

Transport Layer

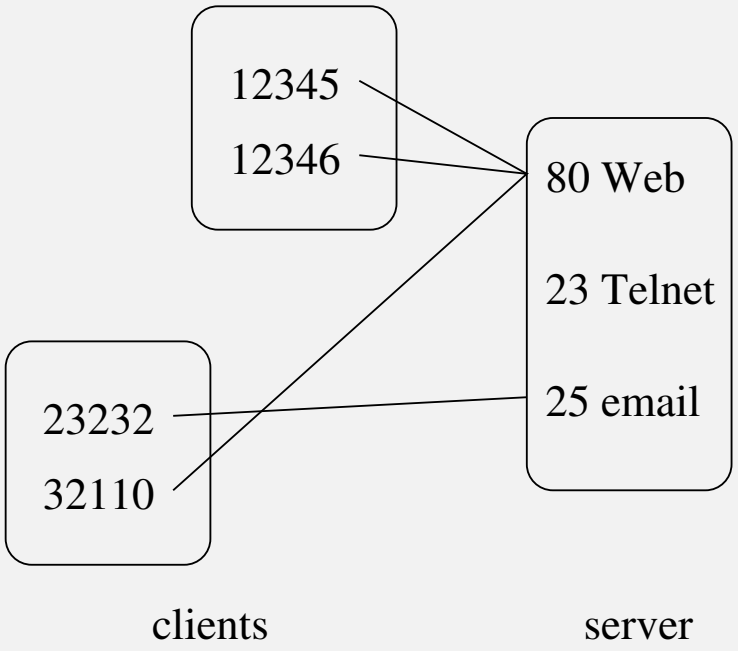
Ports

These associations of port numbers to services are purely convention and for convenience only: no port is special and you can run any service on any port

It just means you don't have the extra problem of determining the port for, say, the web server: it is almost always 80 (or 443)

You can run a web server on port 25 if you wish: you will just confuse anyone who tries to send you email

Transport Layer



Transport layer ports

Ports also enable multiple simultaneous connections between two machines, e.g., fetching several web pages

The source port (destination port on the returning packet) allows the client OS to identify which packet belongs to which client program

Transport Layer

Ports

Source ports are usually chosen afresh “at random” (usually: just increment by 1 for each time) for each new connection and are called *ephemeral* ports as they only live for the duration of the connection

There is no technical difference between ephemeral and well-known ports, just the way they are used

The quad

source address
source port
destination address
destination port

specifies a connection uniquely: the hosts involved and the processes on those hosts

Transport Layer

Ports

The pair (source address, source port) is often called a *socket*

A full quad is then called a *socket pair*

Both TCP and UDP have port fields early in their headers: this is so that the port numbers are included in the “IP header plus 8 bytes of data” that an ICMP error contains

Thus the OS can identify which process an ICMP belongs to

And a non-initial IP fragment won’t have such identifying information, so this is why ICMPs are not generated for errors involving such fragments

Transport Layer

NAT and Ports

And ports are how a NAT firewall does its magic of matching returning reply packets to request packets

It keeps a list of private (internal) socket pairs against public (external) socket pairs

And this is enough to match up replies with requests

Transport Layer

NAT and Ports

Exercise Read about *Port Address Translation*

Exercise Sometime we wish to allow an external host to initiate a connection with a private host behind NAT. Read about *port forwarding*

Exercise Reflect upon the idea that ports are “process addresses”, namely a way to identify a particular process within a destination

UDP

We start with the *User Datagram Protocol* (UDP) as it is simpler, though historically it came along much later than TCP

UDP is the transport layer for an unreliable, connectionless protocol

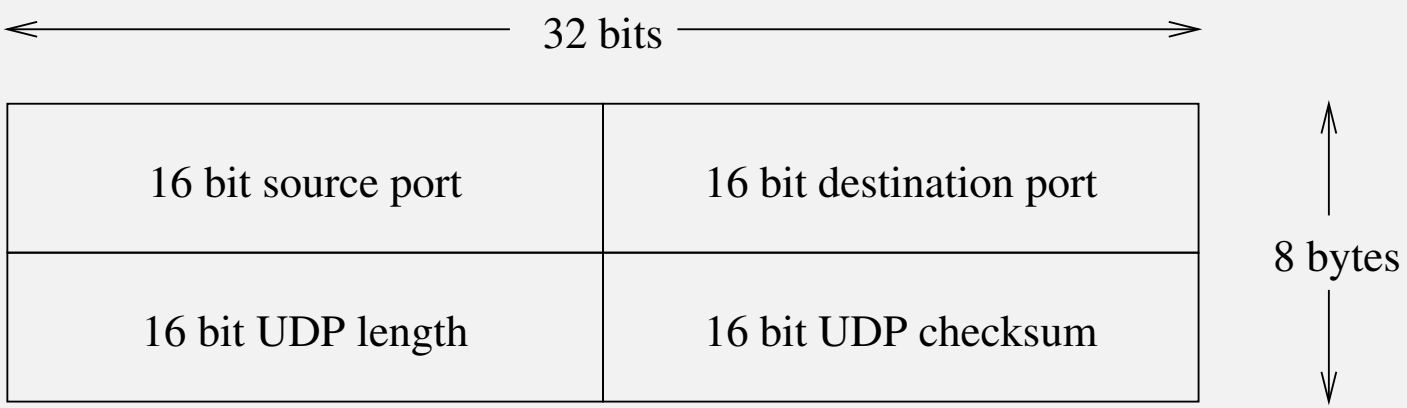
Recall that “unreliable” means “not guaranteed reliable”

UDP is not much more than IP with ports

UDP packets are typically called *datagrams* (like telegrams: simple individual messages)

UDP

Header



UDP header

- Ports: as described
- Length: of the entire packet, including the 8 bytes of the header: this could be deduced from the IP layer, but this keeps layer independence
- Checksum: of the UDP header, the data *and some fields from the IP header*

UDP

Incorporating fields from the IP header is poor design, as it ties UDP to IPv4

Changing the Network layer (e.g., to IPv6) involves changing the way this checksum is computed

Thus adding extra complication to the v4 to v6 transition

The checksum is optional: put 0 in this field if you want to save a little time: recall UDP is unreliable!

UDP

UDP is a very thin layer on top of IP

It is as reliable or unreliable as the IP it runs on

It is just about as fast and efficient as IP, with only a small overhead (8 bytes)

UDP

UDP is widely used as it is good in a few areas:

- One shot applications. Where we have a single request and reply. For example, DNS
- Where a fast response is required. We have no overhead in setting up a connection before data can be exchanged (see TCP). E.g., DNS
- Where speed is more important than accuracy. For example, media streaming, where the occasional lost packet is not a problem, but a slow packet is

UDP

No provision is made for lost or duplicated packets in UDP. Any application that uses UDP must deal with these issues itself, as required

For example, DNS over UDP sets a timer when a request is sent. If the reply takes too long in coming, assume the request or the reply was lost and resend the request

Duplicates are not a problem with DNS

A video streamer might just patch over a lost packet with a copy of a previous packet; and so on

Exercise UDP is ideal for streaming video and audio, but a lot of services use HTTP over TCP. What are the advantages and disadvantages of doing this?

UDP

UDP is a widely used protocol (e.g., streaming video or audio), but we also require a reliable way of sending data

Thus the need for TCP

TCP

The *Transmission Control Protocol* (TCP) is the transport layer for a reliable, connection-oriented protocol

Often called “TCP/IP”

It is *hugely* more complicated than UDP as it must create a reliable transport from the unreliable IP it runs on

There is a lot of complication to deal with the error cases, such as packet loss and packet duplication

There is overhead in setting up (and taking down) the connection to manage these mechanisms

And more to complexity improve performance and flow control

A lot of state about each connection needs to be stored by the OS

TCP

The basis of the reliability is the use of acknowledgement (ACK) packets for every packet sent

If host A sends host B a packet, B must send an ACK packet back to A to inform it of the safe arrival of the packet

If A does not get an ACK, it resends the packet

But ACKs on their own do not solve all the problem

TCP

This is due to the *Two Armies Problem*: suppose two armies A and B wish to coordinate an attack on C

A sends a message to B: “attack at dawn”

How does A know that B got the message? A cannot safely attack until it knows B is ready

So B sends an acknowledgement to A: “OK”

But the ACK might be intercepted and A might not get the ACK

TCP

B can’t attack until it knows A got the ACK

So A should send an ACK for the ACK back to B

But this might not get through. . .

For full reliability it looks like we need an infinite regress!

TCP

TCP avoids the Two Armies Problem by using timeouts and packet retransmissions

For every packet:

A starts a *retransmission timer* when it sends to B

If the timer runs out before it gets an ACK, it resends the packet and restarts the timer

Repeat until A gets an ACK (or A gives up)

TCP

Problems to solve include:

- how long to wait before a resend? This might be a slow but otherwise reliable link and resending will just clog the system with extra duplicate packets
- how many times to resend before giving up? It might be the destination has gone away entirely (perhaps disconnected or crashed)
- how long B should wait before sending the ACK? You can *piggyback* an ACK on an ordinary data packet, so it may be better for B to wait until some data is ready to be returned rather than sending an otherwise empty ACK. This saves on packets sent
- IP datagrams can arrive out of order, so we need some way to recognise which ACK goes with which packet

TCP

Other problems TCP also needs to address include:

- how to maintain order in the data? IP datagrams can arrive out of order, so we need some way of reassembling the original data stream in the correct order
- how to manage duplicates? Resends can produce duplicate packets (if the original was not actually lost) so we need some way to recognise and discard extra copies
- Flow control: how to increase the rate of sending packets when things are going well, and decrease the rate when they are not

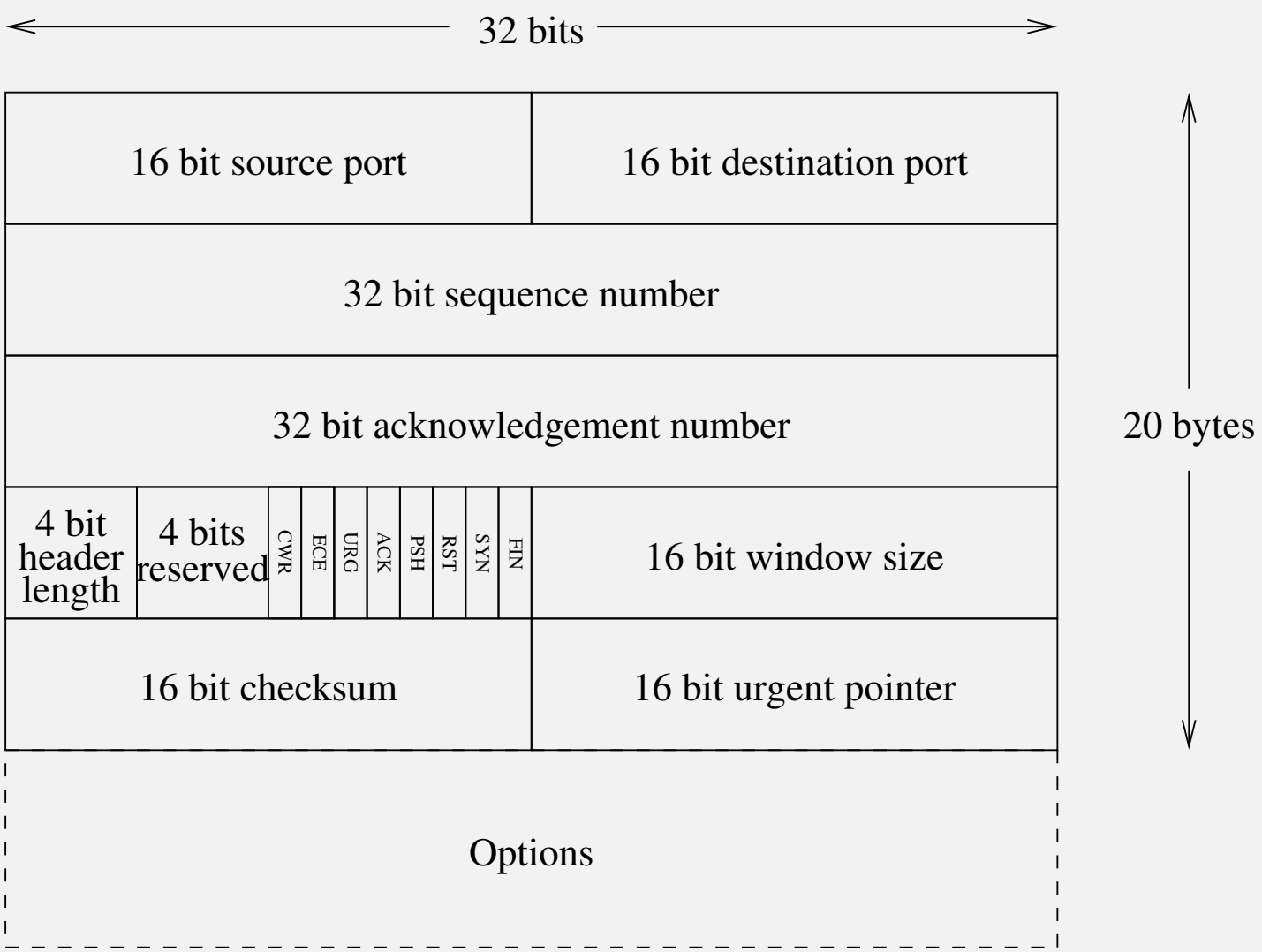
TCP

TCP packets are often called *segments*

(Reminder: “segment”, “packet”, “datagram”, “frame” all mean pretty much the same thing, just in different layers)

A TCP header is complicated as it must address many complex issues

TCP



TCP header

TCP

- Ports: identical to UDP (on purpose: actually UDP copied TCP)
- Two 32 bit values: *sequence* and *acknowledgement*

TCP

Sequence numbers

These numbers are the heart of TCP's reliability

Every **byte** in a TCP connection is numbered

The 32 bit sequence number starts at some random value and increases by 1 for each byte sent

So if a segment contains 10 bytes of data, the sequence number on the next segment sent will be 10 greater

TCP

Sequence numbers

The sequence number in the header is the number of the first byte of data in the segment

The destination acknowledges those bytes it has received by filling in the ACK field with the appropriate byte number and setting the ACK flag

TCP

Sequence numbers

The reverse connection from destination to source has its own sequence number as TCP is fully duplex

Everything we say here is true for data travelling in the reverse direction: the reverse traffic has its own independent sequence numbers and flow control

TCP

Sequence numbers

Note that a destination might not immediately get the whole segment that was sent due to fragmentation in the IP layer

IP must wait for all the fragments and reconstruct the segment before it can pass it on to TCP and then TCP can send the ACK

And this can play havoc with TCP's timers

Another reason to avoid fragmentation

TCP

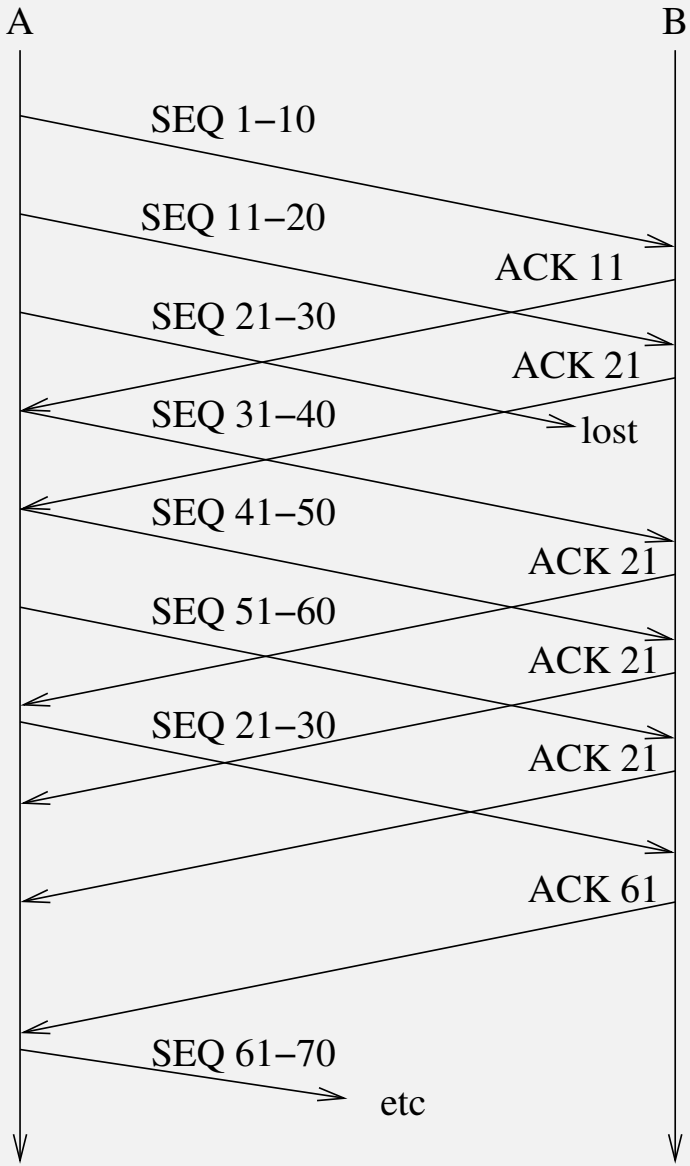
Sequence numbers

The returning ACK field contains the sequence number of the next byte the destination expects to receive, e.g., if the sequence number is 20001 and 14 bytes are received it returns 20015 in the ACK field

ACKs can be *piggybacked* on normal returning data packets, they don't need to be separate packets

This helps reduce the amount of network traffic

TCP



ACKing lost segments

A is sending 10 byte segments to B, and B is ACKing them; The segment containing bytes 21-30 is lost; When B next gets a segment it still ACKS with 21: that's the byte it wants next; While the ACK travels back to A, A is still sending new data; Eventually A gets duplicate ACKs from B: this is a sign of a problem; A resends bytes 21-30; When B gets these bytes it can

TCP

Sequence numbers

In fact this diagram is not realistic: it is over-simplified to fit on the slide

TCP specifies that A should continue until it get gets *three duplicate ACKs* (i.e., four ACKs with the same sequence number, not piggybacked on data and not changing the advertised window) before resending

This is to avoid triggering resends too easily, e.g., it might be just a case of A's packets being slightly reordered in transit, where a resend is not actually required (remember TCP runs on top of the unreliable IP)

Exercise When might we receive many ACKs with the same sequence number, but nothing is in error?

TCP

Sequence numbers

The sequence number wraps around after
$2^{32} - 1 = 4294967295$ bytes

This is under 10 seconds for a 10Gb/s Ethernet

Additional mechanisms to extend the count have had to be devised in the light of modern fast networks

Exercise E.g., using the TCP header timestamp option. Read about PAWS

Much more on SEQ and ACKing later, but note that sequence numbers solve the segment ordering problem, too

TCP

Back to the TCP header

- 4 bits header length: measured in 32 bit words: the header can have options, so is of variable length

So maximum is 60 bytes. Minimum is the fixed part: 20 bytes

- Many flags performing various functions

TCP

Most of these will be described in more detail as we go along:

- URG: urgent data
- ACK: the acknowledgement field is active
- PSH: push this data to the application as fast as possible
- RST: reset (break) the connection
- SYN: synchronise a new connection
- FIN: finish a connection
- ECE: congestion notification
- CWR: congestion window reduced
- 4 reserved bits, set to 0

TCP

Flow Control

- 16 bits of *advertised window size*: for flow control

TCP implements *flow control*, i.e., adjusting the rate of sending packets up or down to make best use of current conditions (a) in the network and (b) in the receiving host

The advertised window deals with (b)

The destination has only a limited amount of buffer memory it can store new segments in

If the application is not reading the data as fast as it arrives, the buffer will fill up

TCP

Flow Control

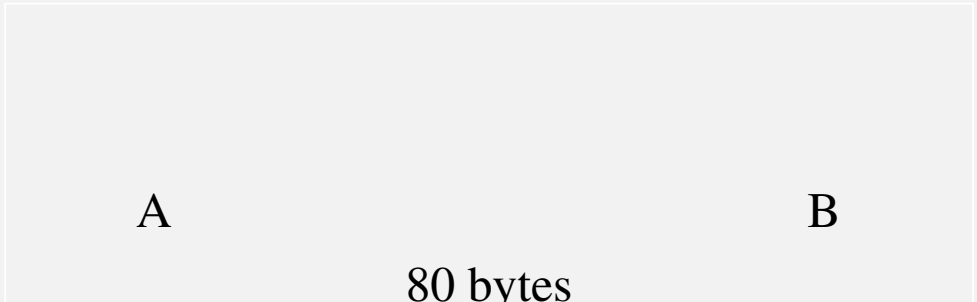
The window size is the amount of buffer the receiver has left: the receiver sends this value in each segment going back to the sender

If the space left is very small, the sender can slow down sending until space in the receiver is freed up

TCP

Flow Control

Initially B has space 100 in its buffer



TCP

Flow Control

Thus B can tell A to slow down or speed up as appropriate to its remaining buffer space

16 bits gives a maximum buffer of 65535 bytes: much too small for modern hosts that have megabytes to play with

There is a header option to scale this up to something reasonable

Symmetrically, A has its own advertised window that it sends to B

The other flow control mechanism to deal with varying conditions in the network comes later

TCP

- Checksum of the header, the data, *plus some fields of the IP layer*

Again, bad design!

- Urgent pointer: active if the URG flag is set

The urgent pointer is a pointer into the data stream that indicates where the current *urgent data block* ends

Urgent data includes things like interrupts that need to be processed before any other data that is buffered

TCP

The OS should notify the application when an URG is received, e.g., using an interrupt

The OS interrupt code would then read through the urgent data block and act appropriately on what it finds there

TCP

In a similar vein we have the

- PSH flag: set to indicate the destination OS should pass data to the application as soon as possible

The destination OS might be holding back data for some reason before passing it on to the application, e.g., collecting together segments into one large buffer for efficiency reasons

Or holding back notifications to the application that data has arrived: again not to swamp the application with loads of notifications of small amounts of data

This flag says send the buffered data to the application, don't wait

TCP

Originally it was intended the client application could set the PSH when it felt the server should not be hanging about buffering data

These days, there is no mechanism (in the sockets API) for applications to specify this, but the TCP software itself sets PSH when appropriate, e.g., when the client's send buffer empties

The idea here is that there is no point for the receiver waiting for more data, as there is no more to send right now

TCP

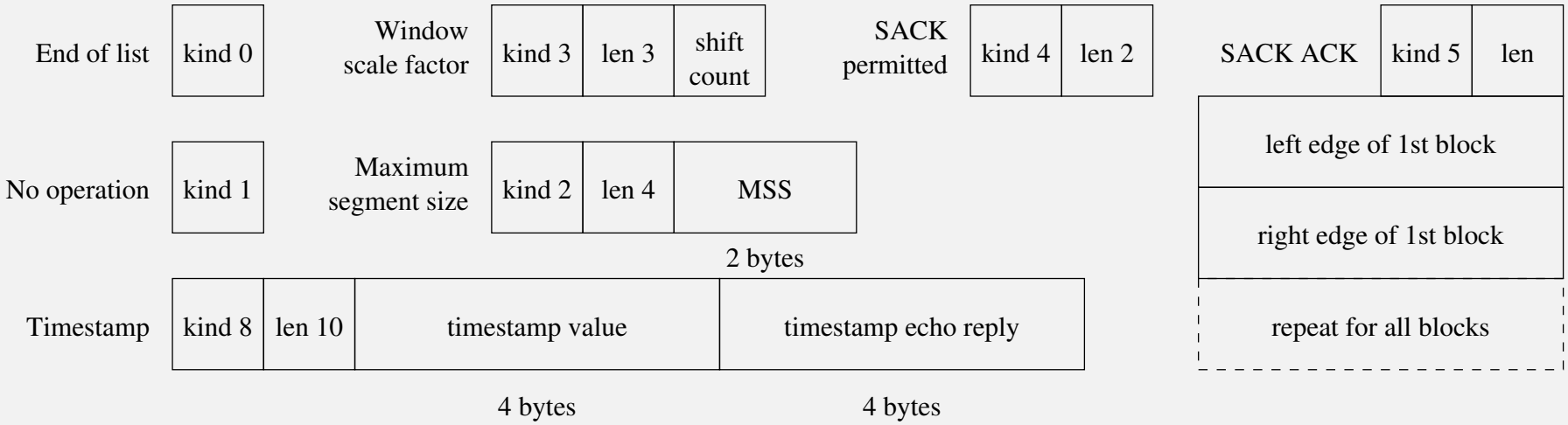
After the fixed header there are the options, including *window scale* and *maximum segment size*

After the options header is the data, which can be empty, e.g. for a pure ACK

TCP

Options

TCP Options are many and varied



Some TCP optional headers

TCP

Options

Options start with a 1 byte *kind* which indicates what the option is to do

Kinds 0 and 1 are one byte long; others have a length field

No operation (NOP) is used to pad to align fields to a multiple of 4 bytes

Maximum segment size (MSS) specifies how large a segment we can cope with: the headers are not included in count

TCP

MSS

The MSS is the largest TCP segment the host can process

Note that this segment *might* be reconstructed from more than one IP fragment, so might not be directly related to the MTU

However, if we want to ensure no IP fragmentation, the MSS must be set to the MTU minus headers: $40 = 20 + 20$ bytes for IP and TCP

Thus a TCP implementation must be able to process a MSS of $576 - 40 = 536$ bytes

The MSS is usually communicated in the option header in the setup of the TCP connection, and is typically set to avoid fragmentation

TCP

Options

As previously mentioned, the *window scale* option allows us to multiply up the value in the advertised window size header field

This optional field contains a value from 0 to 14

A value of n scales by 2^n: thus a maximum window of $2^{14} \times 65535 = 1,073,725,440$ bytes (a gigabyte)

But that's still only about a second's worth of data in a 10Gb/s Ethernet!

A large window is very important is modern fast networks to get the most out of the available bandwidth: we don't want the client to have to keep stopping to wait for the server

TCP

Options

My desktop uses a window scale of 7: $2^7 \times 65535 = 8388480$ bytes, or a maximum of 8MB buffer space per connection

Its initial window size on a new TCP connection is 14600, meaning $2^7 \times 14600 = 1868800$ bytes, so a buffer of a bit under 2MB has been allocated (for this socket)

Exercise Go back and re-read the section on advertised windows

TCP

Options

Timestamp (TS val) puts the time of day into the segment header, allowing accurate measurement of the *round trip time* (RTT) of a segment and its ACK. Useful for computing retransmission times (see later)

Timestamp Echo Reply (TS ECR) in an ACK segment is the timestamp being returned to the sender so it can compute the RTT

Selective acknowledgement (SACK) is an extension of the ACK mechanism that allows more flexible ways of acknowledging segments. SACK is negotiated in the connection setup with a *SACK Permitted* option

TCP

Options

Several options are only allowed in the first segment of a new connection, e.g., Window scale, MSS and SACK Permitted

This is because some things, e.g., buffer space, need to be set up before a connection and varying them mid-connection is difficult or makes little sense

TCP

Setup and Teardown

TCP is *connection oriented*, meaning a connection is set up between source and destination, and all packets that flow within this connection are related, through the sequence numbers, and they all use the same state, such as advertised window

For example, a connection to fetch a web page from a server will involve many segments

Note that *each* TCP connection is separate from all others and has its own state

It is important to realise that this is a connection in the *transport layer*

TCP

Setup and Teardown

The underlying layer, IP, is not connection oriented, and each individual datagram is treated individually, e.g., might take a different route to its destination: IP is *connectionless*

Thus TCP connection has a weak version of sessions: though no further session mechanism is provided, e.g., no session resumption

TCP

Connection(less)

UDP is not connection oriented. Each datagram in UDP is treated individually

UDP is a *connectionless* protocol

Of course, both connection oriented and connectionless protocols are useful in the right circumstances

TCP

Setup and Teardown

Setting up a TCP connection is complicated, as there is a lot of state that must be set up, e.g., sequence numbers, initial advertised windows, and buffers amongst other things

Similarly, closing a connection is not trivial: we must ensure all segments in flight have been ACKed properly. Perhaps segments need to be resent. Thus a connection will hang around for a little after closing to ensure everything is tidied up

Fortunately for the application programmer, all this detail is taken care of by the TCP layer software in the operating system: though it does have occasional repercussions in the application if the connection needs to outlive the application for a while

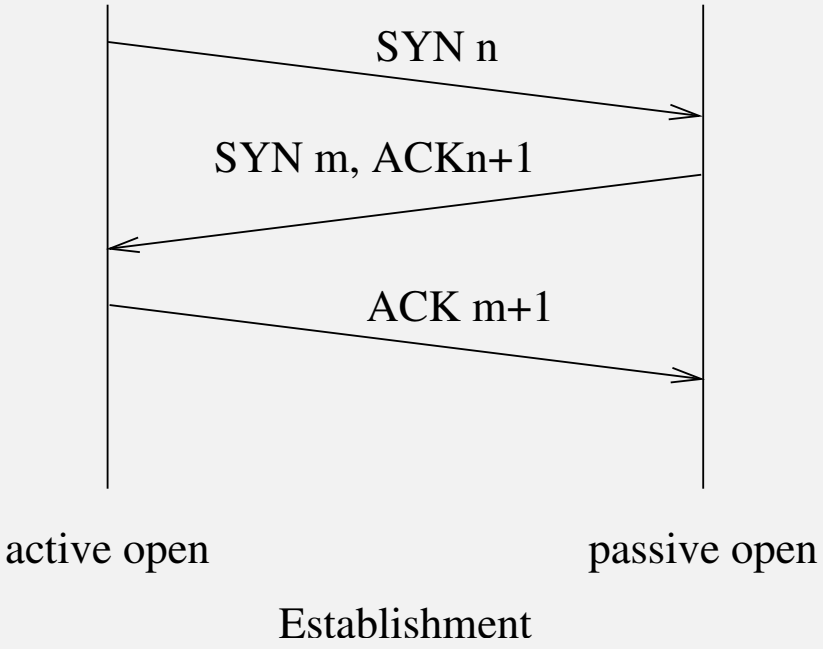
TCP

Setup and Teardown

Before TCP can send data, it exchanges some packets with the setup information

TCP

Setup and Teardown



TCP setup handshake

Three segments are needed to exchange the information needed to make a new connection; The initiator, the *client*, sends a segment with the SYN flag set and its *initial sequence number* (ISN), *n*, is randomly generated; The receiver, the *server*, replies with another SYN segment containing its own

TCP

Setup and Teardown

This is called a *three way handshake*

These segments contain no user data: they are overhead in setting up the connection

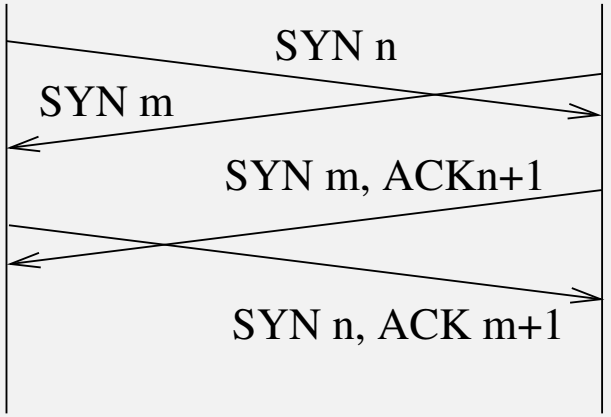
Overhead in time and overhead in packets on the network

After the handshake we can start sending data

The client (first one to initiate) is said to do an *active open*, while the server does a *passive open*

TCP

Setup and Teardown



Establishment

TCP simultaneous open

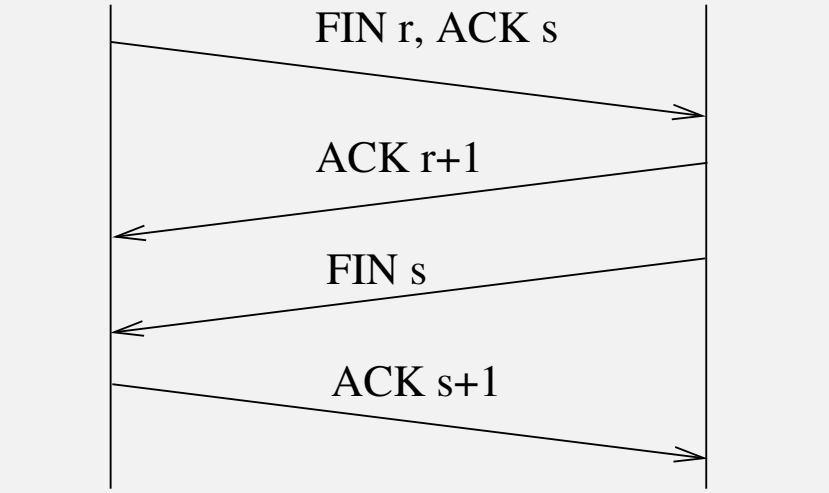
It is possible (but rare) for *both* hosts to do an active open, where the SYNs cross each other in flight

Matching TCP port numbers will identify when this happens

This is defined to produce *one* new connection, not two

TCP

Setup and Teardown

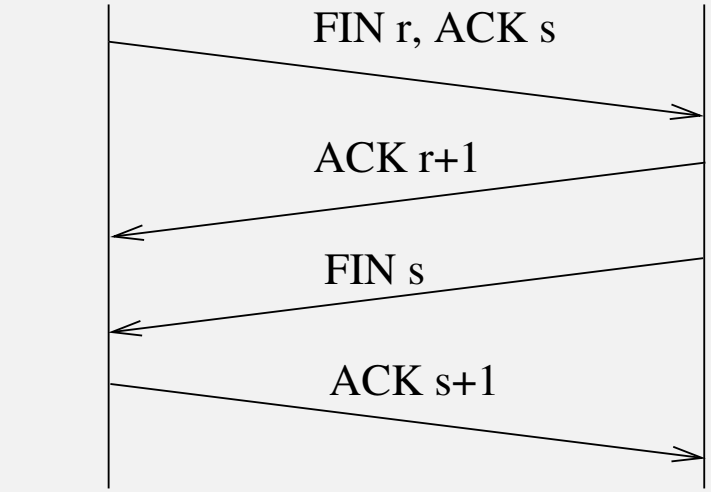


TCP teardown

Closing a connection takes up to four segments; TCP is full duplex, and a connection in one direction may be closed independently of the other; The FIN flag is set to indicate a *half close*: this indicates no more data will be sent from this end; We can still *receive* data at this end; The FIN is ACKed; When

TCP

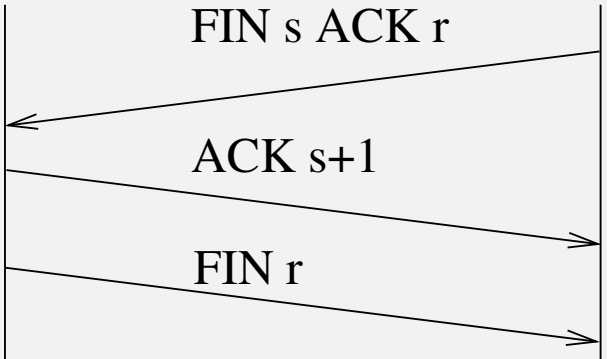
Setup and Teardown



active close passive close

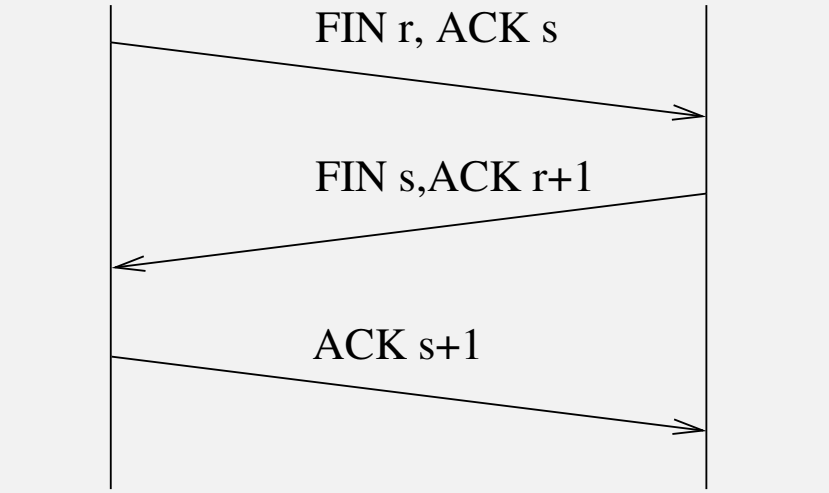
Normal Termination

Active close from left



TCP

Setup and Teardown



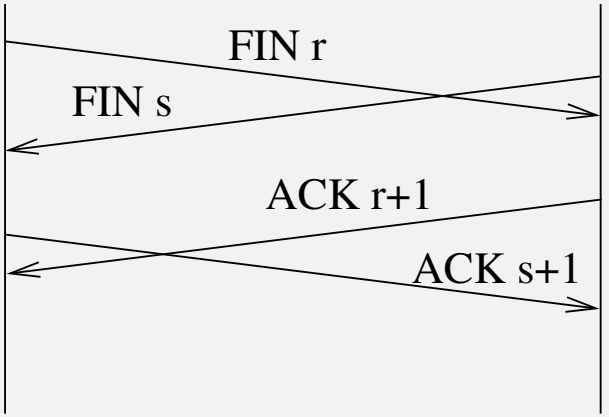
3 Segment Termination

Three segment close

The passive close FIN can be piggybacked on the ACK: this then takes only three segments

TCP

Setup and Teardown



Termination

Simultaneous active close

There can (rarely) be a simultaneous active close: this takes four segments again

TCP

Termination

Connections are almost always ended by the FIN handshake, but there is another way to end a connection when something is badly wrong

This is to send a *reset* (RST) segment, i.e., with the RST flag set

This is for error cases, e.g., a segment arrives that doesn't appear to be for a current connection, the server will reply with a RST

For example, if a server crashes and reboots while the client is still sending the server will not know what to do with the segments it is receiving; so it replies with a RST

TCP

Termination

When a host gets a RST it ends the connection immediately, discarding all state and buffered segments

Often seen by the application as a “connection reset by peer” message

TCP

Termination

A connection ended by FINs is called an *orderly release*; if ended by a RST it is an *abortive release*

RSTs are not ACKed: the connection ends right here

Exercise Think about the security aspects of this: a third party can inject a RST segment into a connection to kill it

TCP

TCP State

The various stages a TCP connection can be in (setting up, tearing down, transmitting data, etc.) are complicated

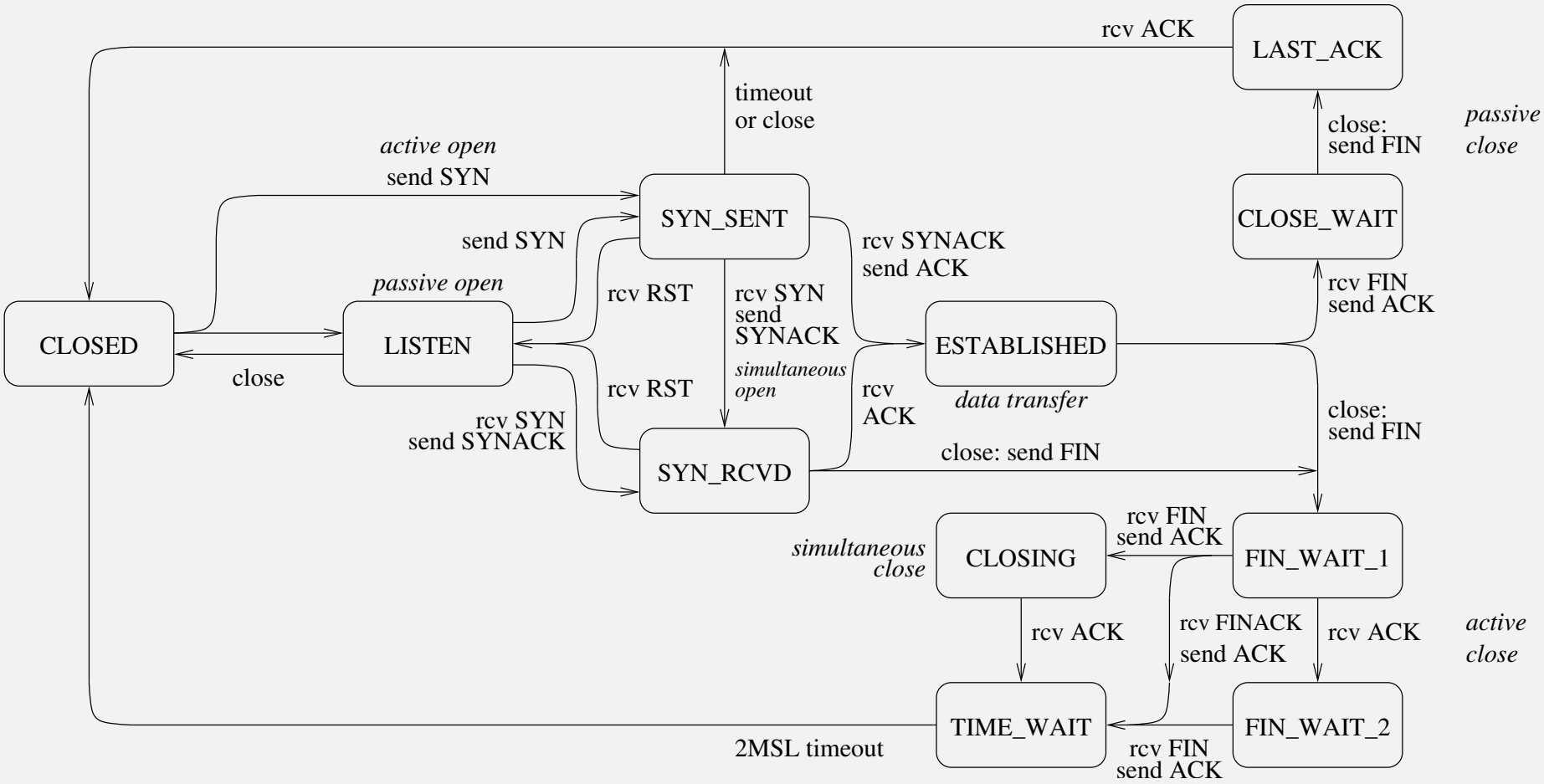
There is a standard TCP state diagram that describes how TCP should act in most cases

Though it only covers non-error cases: it does not say what to do if, say, a SYNFIN segment arrives

And it shows little about timeouts and retransmissions

TCP

TCP State



TCP State Diagram

TCP

TCP State

We start (and end) in CLOSED

There are the two opens: active and passive

LISTEN is a server waiting for a connection

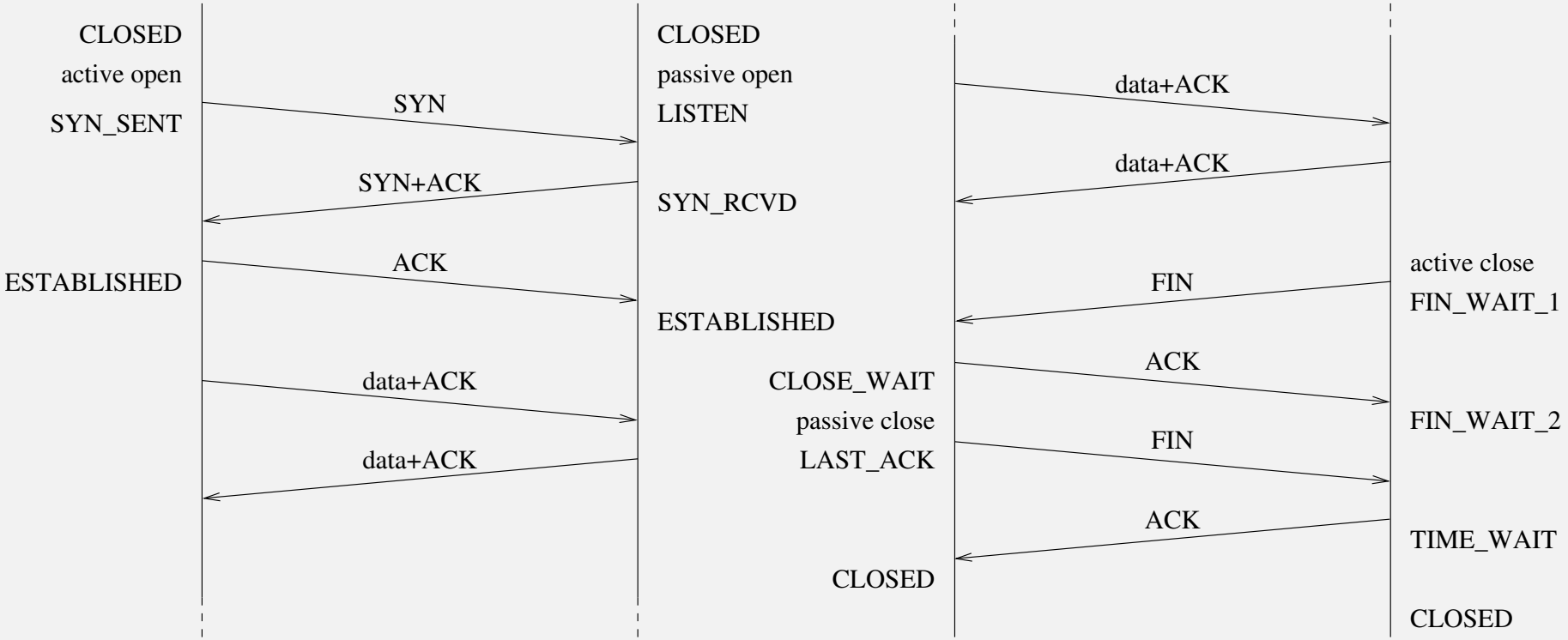
ESTABLISHED is the normal data transfer state

And the two closes: active and passive

This state diagram is followed for each end of a connection, i.e., each socket in the socketpair

TCP

TCP State



Typical TCP Timeline

TCP

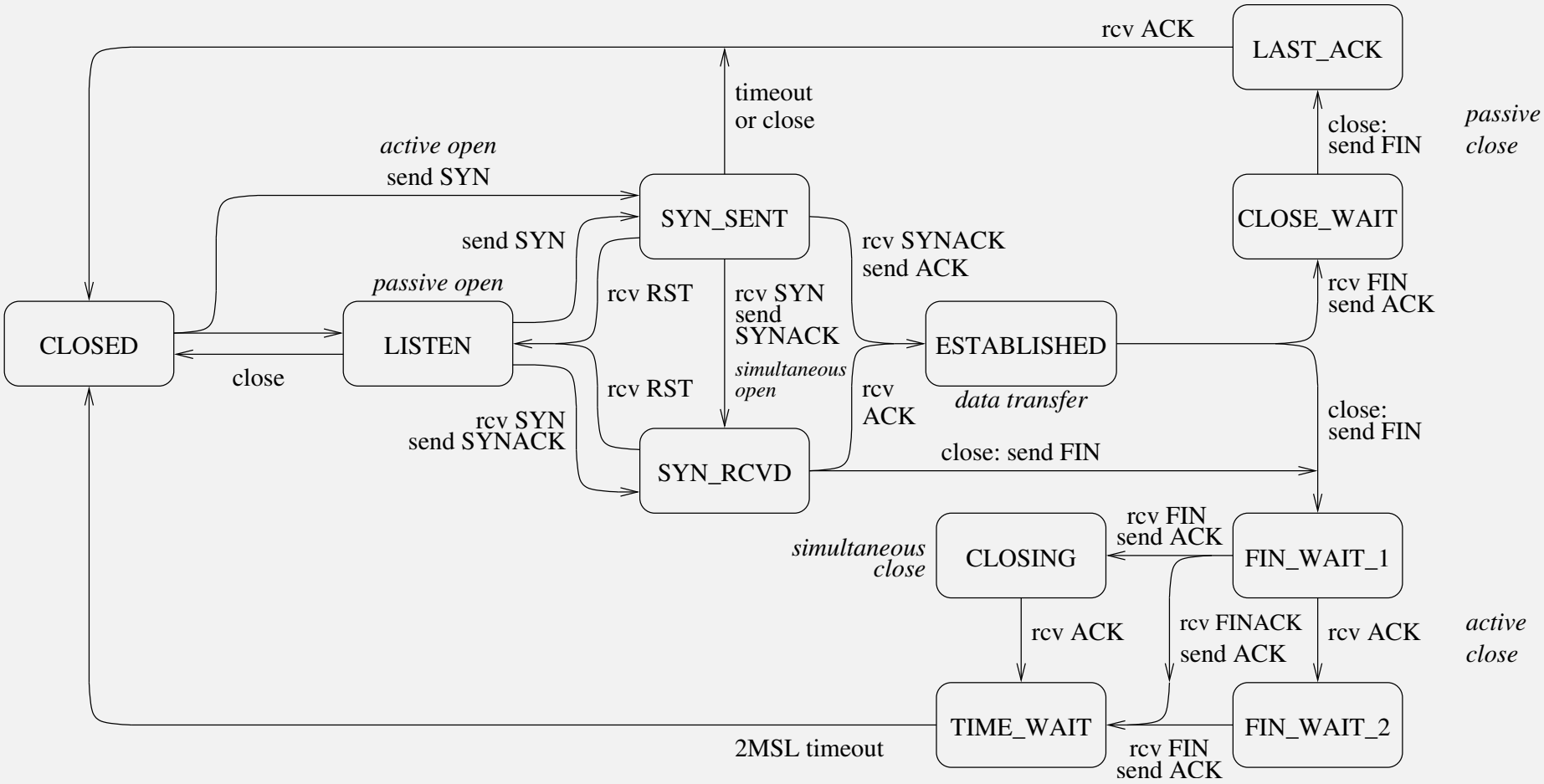
TCP State

The active close is somewhat complicated by the need for reliability

The TIME_WAIT state (also called 2MSL state) appears before the final close: the active-close end of the connection must remain non-closed until a time period has passed

TCP

TCP State



TCP State Diagram

TCP

TCP State

At this point this end of the connection has received a final ACK and sent its final ACK

In a perfect world this would be enough to close the connection

But we have to deal with the case of the final ACK being lost

And resent if it didn’t get to the other end

TCP

TCP State

Just because the application is done with the connection, it doesn't mean the connection is finished and the OS can discard all the connection state

The *maximum segment lifetime* (MSL) is a value that represents the longest time a segment can live in the network before being discarded (probably through TTL expiry)

This was originally defined to be 2 minutes, but implementations often choose smaller values, like 60 seconds

A TCP connection is required stay in TIME_WAIT for twice the MSL

TCP

TCP State

This is in case the final ACK (of the final FIN) was lost and needs to be retransmitted

The OS has to keep the connection hanging around for a little to cover this case

Even if the process that used the connection has exited

And while in this wait state if a new process tries to make a connection using the same ports it will be denied: the old connection is still active. We don't want to deliver late packets to the new process

In this sense the TCP connection and the process using it are quite separate entities

TCP

Teardown

When an application exits, the OS sends FINs on behalf of the application for all currently open connections. This makes sure everything is tidied up nicely (even if the programmer didn't)

And if it was an active close, OS needs to hold the connection in the `2MSL` state for a while: the connection definitely outlives the application!

If a host is shut down normally, rather than crashing, the operating system will (should!) send FINs for all currently open connections

It really should do the `TIME_WAIT`, but often implementations don't bother as this would hold up the shutdown

TCP Strategies

We now take a look at how TCP manages to get the best out of a connection

For example: TCP gets reliability by acknowledging every byte sent. Does this mean two segments for every data packet: one data packet out, one ACK packet back?

It is possible to implement TCP like this, but performance would be poor

So a typical TCP implementation will be a bit more smart on its use of ACKs: we have already mentioned delaying an ACK to let it piggyback on a returning data segment

TCP Strategies

That is just first of many strategies a TCP implementation can employ while still following the TCP protocol

We shall look at a few basic strategies, starting with more detail on the advertised window

TCP Strategies

Advertised Window

As data arrives at its destination the OS puts it into a buffer, ready for the receiving application to read it. We have already seen the TCP *advertised window* in a returning segment which indicates how much of this buffer space is left

The space left depends on

- how fast the sender is sending the data
- how fast the application is reading the data

If the data arrives faster than it is read, the buffer will fill up

TCP Strategies

Advertised Window

The advertised window is how TCP tells the source to slow down or speed up

It is a *sliding window* mechanism, used as a form of flow control

Imagine the bytes being sent as a long stream, starting at byte 0 (actually byte *n*, given by the initial sequence number) and going up

A sliding window describes the range of bytes in the stream the sender can transmit next

TCP Strategies

Advertised Window

As the window gets smaller, the sender should send more slowly

As the window gets bigger, the sender can send more quickly

The sender recomputes the space available in the receiver every time it receives an ACK

TCP Strategies

Advertised Window

The left hand edge of the window is defined by the acknowledgement number in the latest ACK

The right hand edge is then given by adding on the size of the advertised window

The window size is sent in every ACK segment

As more ACKs are received, the window *closes* as the left edge advances

TCP Strategies

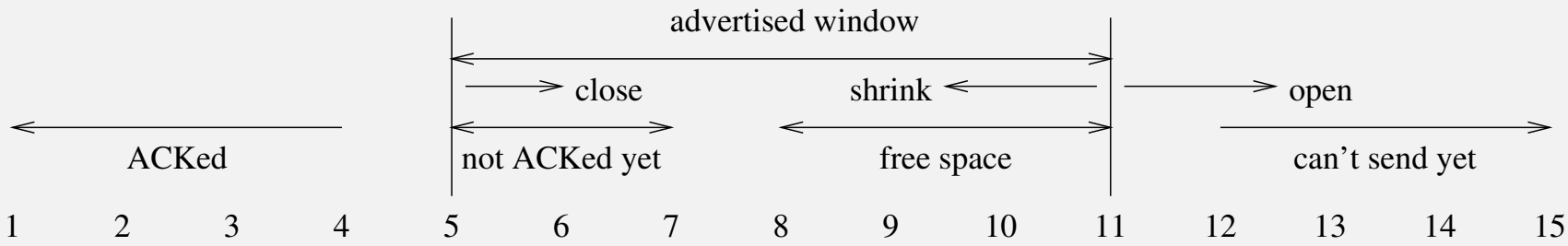
Advertised Window

As the application reads data, the window *opens* as the right edge advances

Rarely, the window can *shrink* (right edge recedes), perhaps if the buffer shrinks due to the memory being needed elsewhere

TCP Strategies

Advertised Window



TCP sliding window

This is from the point of view of the sending end of a connection; The situation is that we have just sent a segment with bytes 5-7; then received an ACK of 5 with a window of 7; Bytes to the left of the window (1-4) have been ACKed and are safe in the destination; The advertised window tells us there is space for 7 bytes in the destination: bytes to the right (12 onwards) cannot be sent yet as the destination has nowhere to

TCP Strategies

Advertised Window

It is not unusual for the window to reduce to 0, for example when the destination application is reading its data slowly

The sender will have to wait before sending more data

When the receiver is ready to receive more data it will send a duplicate ACK with the same ACK number as the ACK with window 0, but now with a non-zero window: this is a *window update segment*

It may or may not contain data itself

Complications arise if this window update gets lost: the *Persist Timer* (see later) is used here

TCP Strategies

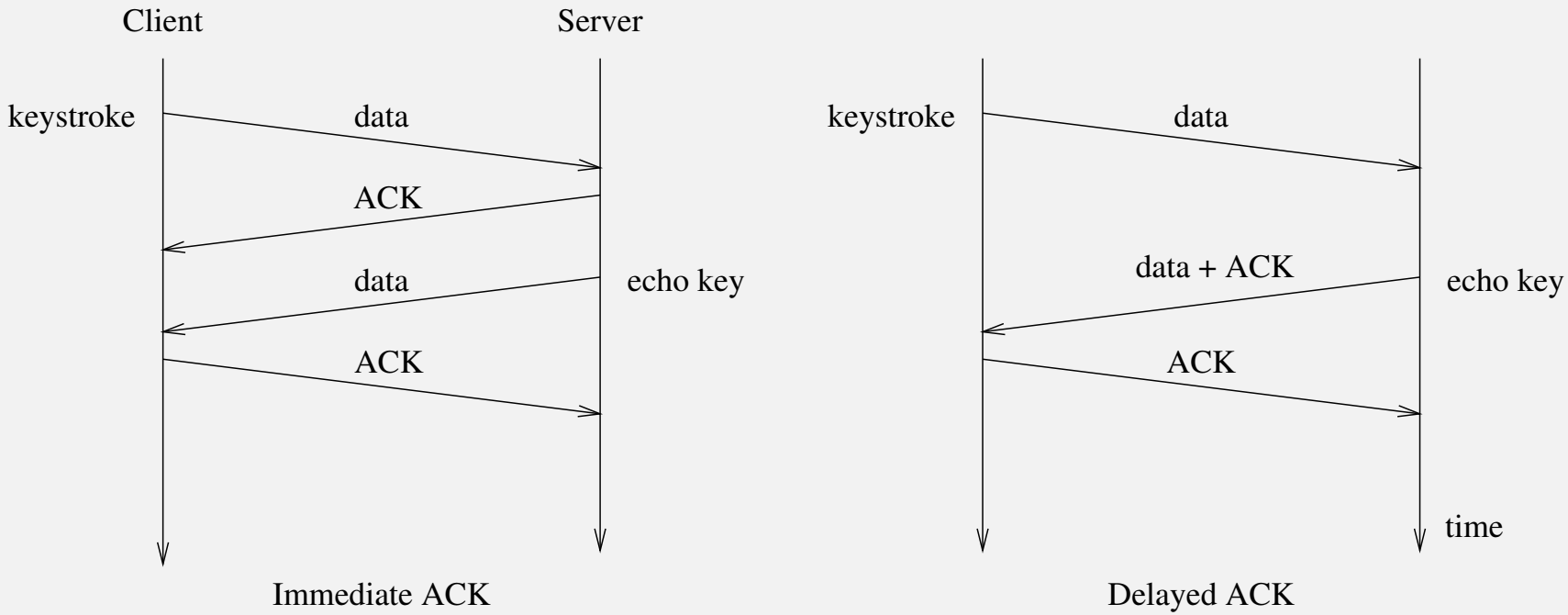
Delayed ACKs

The next strategy we have mentioned before

Instead of immediately ACKing every segment, we can slightly delay it and *piggyback* it on returning data

TCP Strategies

Delayed ACKs



Immediate vs. delayed ACK

For example, when logged in to a remote terminal each keystroke is echoed back to your screen; An immediate ACK would use four segments; A delayed ACK piggybacking on the data for the echoed key uses just three segments

TCP Strategies

Delayed ACKs

As far as the user is concerned, they see the keystroke echo in the same way, with no extra delay, but fewer segments are sent

It is important to reduce the traffic on a heavily loaded network

It also reduces the chance of a lost segment

TCP Strategies

Delayed ACKs

By delaying, we might also be able to ACK more than one segment at a time

If we receive, say, two segments in a period we are delaying, we can simply ACK the last segment: this implicitly ACKs the previous two segments

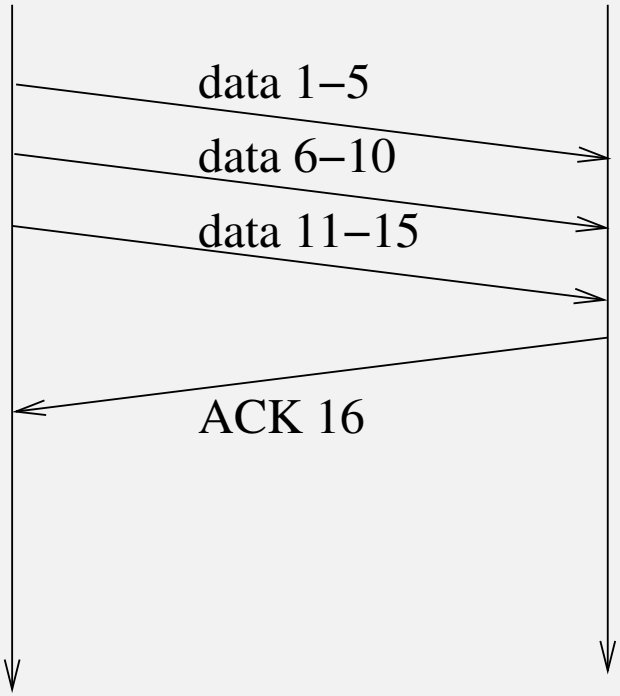
An ACK is actually about acknowledging bytes, not acknowledging segments, but will usually align with segments

So an ACK indicates which byte we are expecting next and says all previous bytes have been safely received

This reduces traffic again

TCP Strategies

Delayed ACKs



ACKing bytes received

ACKs acknowledge bytes received, not segments

TCP Strategies

Delayed ACKs

So how long to delay an ACK?

If too long, the sender might think the segment was lost and resend

If too short, we do not get so many free piggybacks or multiple ACKed segments

A typical implementation will delay for up to 200ms

TCP Strategies

Delayed ACKs

The TCP specification says you should send an ACK for at least every second full-sized segment and you *must not* delay for more than 500ms

This one of the many timers associated with TCP

Each time you receive a data segment the TCP software should set a timer for that segment that expires after 200ms

TCP Strategies

Delayed ACKs

If the segment has not already been ACKed (e.g., on a returning data segment), ACK it when the timer expires

Many operating systems have a single global timer that fires every 200ms rather than a timer per segment received

When the timer goes off, all unACKed segments are ACKed

Not so accurate as per-segment timers, but much easier to implement

TCP Strategies

Delayed ACKs

There is another rule concerning delayed ACKs

If you get an out-of-order segment (its sequence number is not the one you are expecting next, e.g., a segment was possibly lost), you *must not* delay, but send an ACK immediately

This might well be a duplicate ACK of one you sent earlier. This is to inform the sender as soon as possible that something might have gone wrong

Though the other end will wait for three duplicate ACKs just to be sure before resending

TCP Strategies

Nagle

Next strategy: when sending keystrokes (or other small data) over a network there is a lot of wasted bandwidth

A keystroke could be 1 byte

This is sent in a TCP segment that has 20 bytes of header

This is contained in a IP datagram with 20 bytes of header

And so on down the layers

TCP Strategies

Nagle

So we are sending (for the sake of argument) a 41 byte packet for each byte of data

Such a packet is called a *tinygram*

The proliferation of tinygrams causes additional congestion in a network

Nagle created a strategy for reducing this

It applies to the sender of the tinygram (client)

TCP Strategies

Nagle

Nagle's Algorithm:

> *a TCP connection can have only one outstanding un-ACKed small segment: no additional small segments can be sent until that ACK has been received*

If you are sending tinygrams, only send one and wait until you get its ACK before sending any more

Any small segments waiting to be sent should be collected together into a single larger segment that is sent when the ACK is received

TCP Strategies

Nagle

This segment can also be sent if either (a) you collect enough small segments to fill a MSS segment, or (b) they have collectively exceeded half the destination's advertised window size

This leaves open the definition of "small"

Variants choose anything from "1 byte" to "any segment shorter than the maximum segment size"

TCP Strategies

Nagle

Note that when window scaling is in effect, “small” must be at least the size of the window scale factor, as we can’t advertise a window smaller than that

But that won’t be a constraint until the scale is bigger than a segment, e.g., $2^{10} = 1024$, but $2^{11} = 2048 > 1500$

TCP Strategies

Nagle

This is a very simple strategy and reduces the number of tinygrams without introducing extra perceived delay (over that delay there is there already)

The faster ACKs come back, the more tinygrams can be sent

When there is congestion, so ACKs return more slowly, fewer tinygrams are sent

TCP Strategies

Nagle

Nagle can reduce the number of segments significantly when the network is heavily loaded

On the other hand, sometimes buffering up tinygrams is not a good idea: e.g., in a graphical interface over a network, each mouse movement becomes a tinygram. Buffering the segments would cause the cursor to jump erratically

Nagle can be turned off for such cases

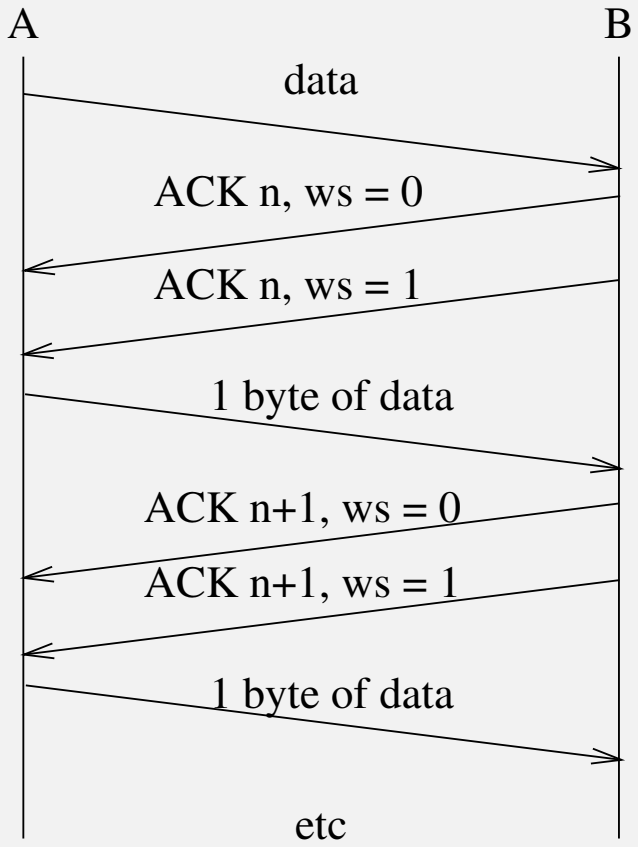
TCP Strategies

Silly Window Syndrome

Another problem with tinygrams is manifested as *silly window syndrome*

TCP Strategies

Silly Window Syndrome



Silly Window Syndrome

A is sending data to B, but B's buffer is nearly full and B is reading only one byte at a time; B's buffer fills, and B ACKs with a window of 0; A holds off sending more data; B reads a byte; B sends a window update segment, size 1; A get this and sends as much data as possible, i.e., 1 byte; B ACKs with window 0; B reads a byte; B sends an update, size 1; A sends 1 byte; And so on

TCP Strategies

Silly Window Syndrome

We are back to the two segment per byte high overhead: this is silly window syndrome

Better is for B not to send an update of 1, but wait until there is more space

Clarke's algorithm to avoid SWS is in the server

> *never send an update for a window of 1; only advertise a new window when either (a) there is enough space for a full segment, or (b) the buffer is half empty*

TCP Strategies

Congestion

Nagle (in the client) and SWS (in the server) fit together naturally

Note that TCP code doesn't have to implement Nagle or SWS or delayed ACKs or any of these strategies: it's just a good idea if it does!

TCP Strategies

Congestion

Nagle and SWS are good for when there is a small amount of data being transmitted

We need to look at the case of sending large amounts of data

We want the data to get to the destination as fast as possible, but we now have to consider not just the ability of the destination to cope, but also the capacity of the network itself

TCP Strategies

Congestion

Congestion happens when more data is being sent than the *network* can handle: routers will drop packets if there is not enough onward bandwidth to cope

There are several strategies in TCP to help deal with and avoid congestion

The first issue is how to spot congestion, given that it might be happening in a part of the network many hops away from both source and destination

TCP Strategies

Congestion

We watch for segment loss

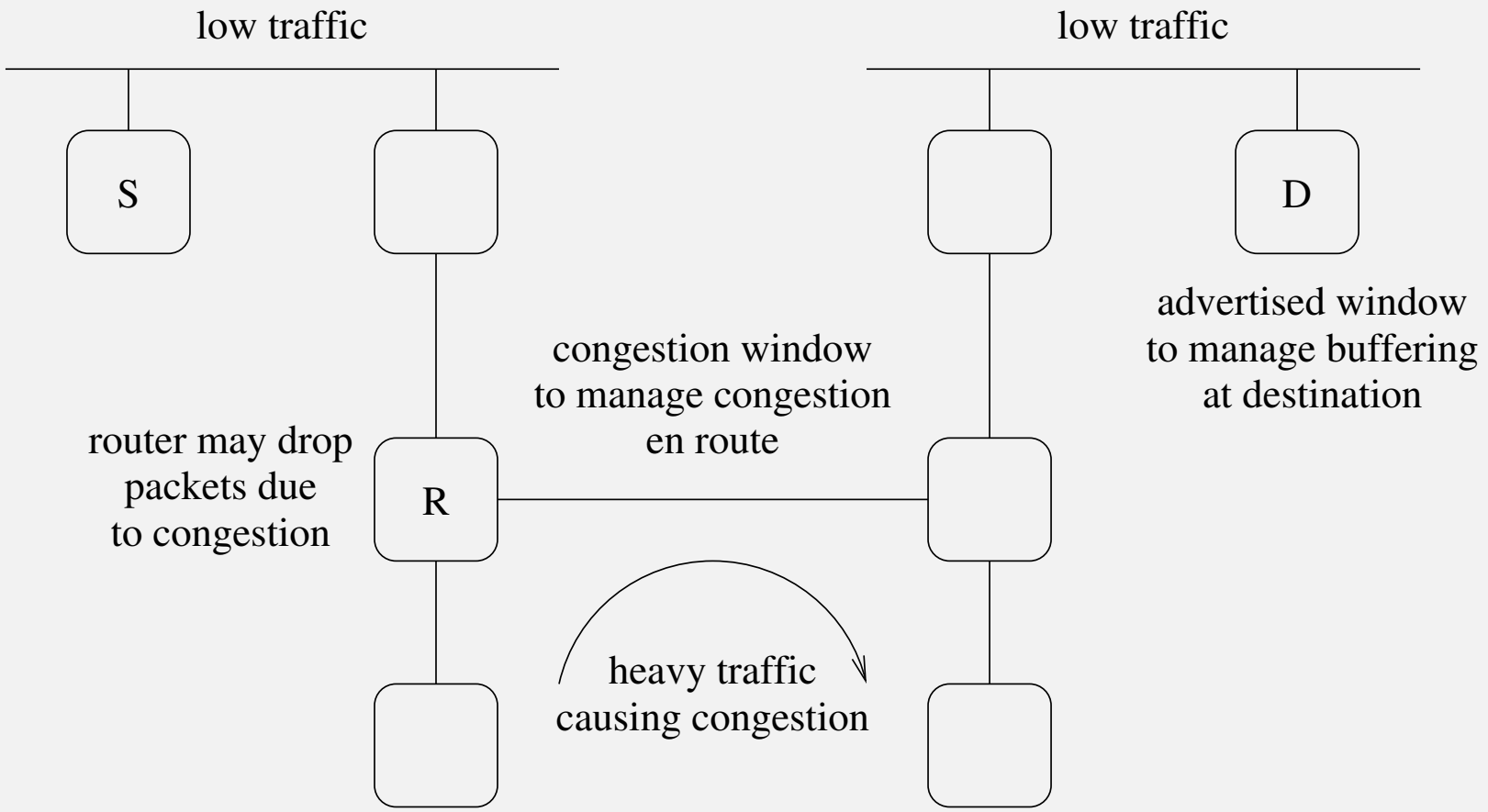
Segments can be lost though errors in transmission or being dropped at a congested router (or at the destination)

Poor transmission is less usual these days, so we can assume loss is due to congestion (which is common these days)

Thus TCP treats missing or duplicate ACKs as a sign of congestion

Exercise A missing ACK is understandable as a sign of congestion: reflect briefly on why *duplicate* ACKs can be caused by congestion

TCP Strategies



Congestion somewhere on the path

Congestion can happen in a router due to lack of capacity in an onward link; a router will drop a packet if it can’t cope

TCP Strategies

Congestion

Just as the advertised window deals with “congestion in the destination” (it’s not really congestion), we have the *congestion window* for congestion in the network

So how do we determine the congestion window? It’s not a thing the source or destination can know directly

We do this by sending segments and watching what ACKs we get

TCP Strategies

Congestion

If we have a lot of data to send we do not want to wait for each ACK before sending the next segment

Better is to send several segments and then wait to see from the ACKs which were safely received

TCP Strategies

Congestion

But sending too many segments at once is bad when the network is congested: our segments will be dropped. We'll just be making things worse for everyone, including ourselves

So, if we have an estimate of the capacity of the network (the congestion window), we will be sending many segments at once, but not too many

If we get it right, we will have a continual stream of segments going out and ACKs coming back

TCP Strategies

Slow Start & Congestion Avoidance

We estimate the network congestion by watching the number of ACKs coming back

This estimate controls the congestion window

This is an another constraint on sending additional to the advertised window: it's a bad idea to send more data than indicated by the either window

TCP Strategies

Slow Start & Congestion Avoidance

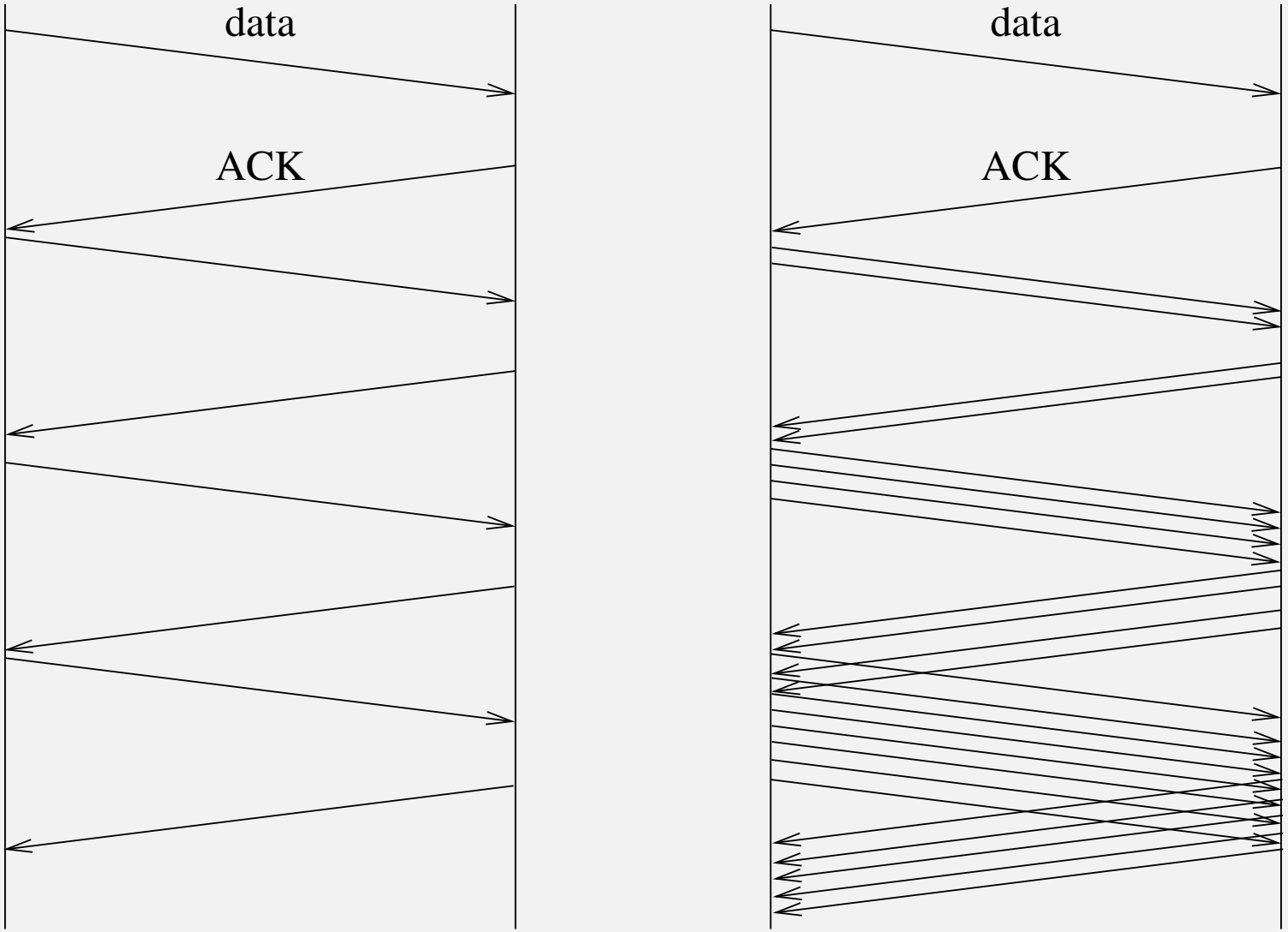
We describe a basic flow control strategy (RFC2001/RFC2581) that estimates the congestion window; many modifications exist (TCP Tahoe, TCP Reno, TCP Vegas, . . .)

The *congestion window* (`cwnd`) is initialised to the maximum segment (MSS) size of the destination

A variable, `ssthresh`, the *threshold*, is initialised to 64KB (say)

Every time a timely ACK is received, the congestion window is increased by one segment

TCP Strategies

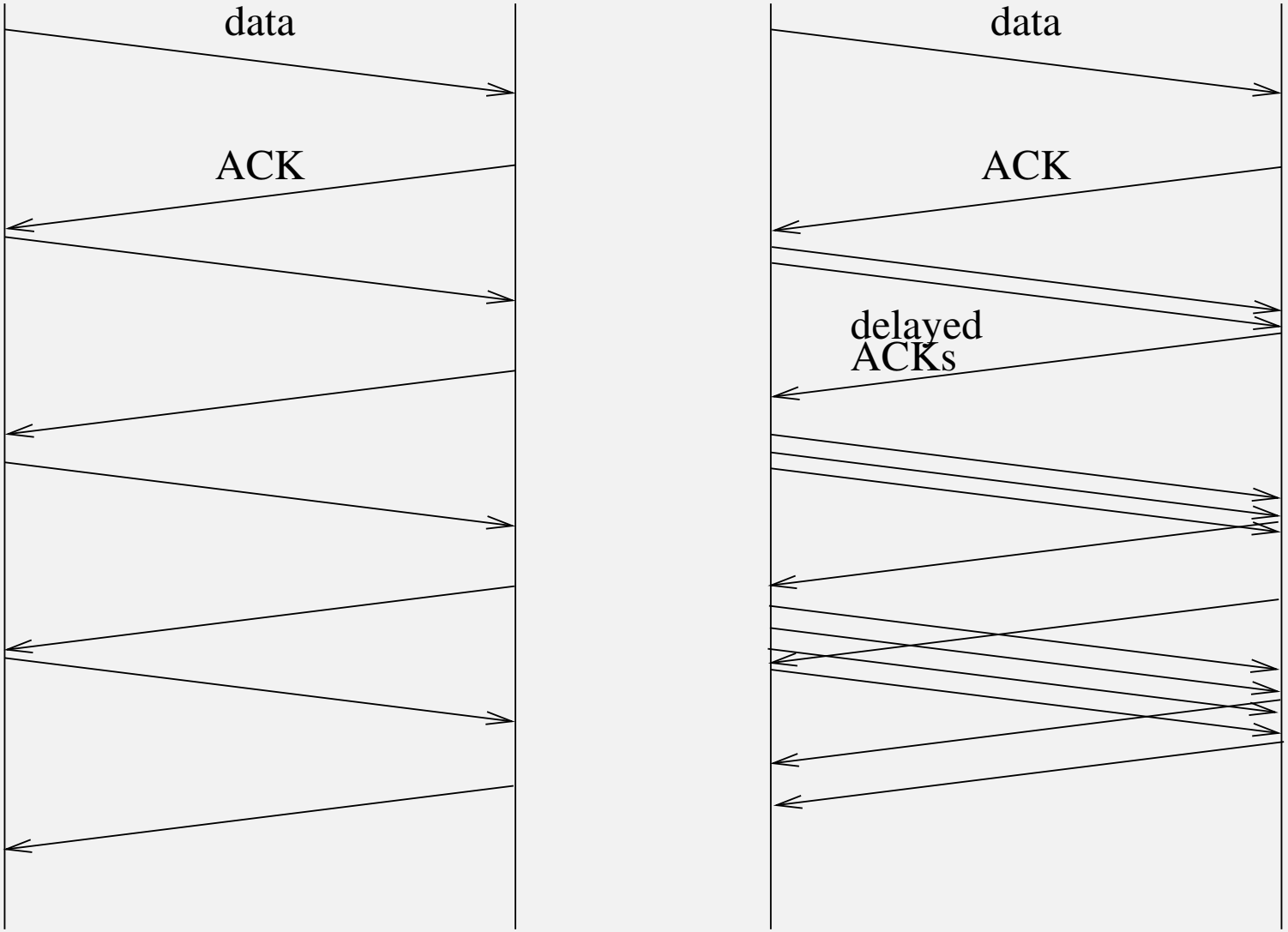


Poor use of bandwidth

Slow Start (no delayed ACKs)

Slow Start with no delayed ACKs

TCP Strategies



Poor use of bandwidth

Slow Start

Slow Start with delayed ACKs

TCP Strategies

Slow Start & Congestion Avoidance

So initially we send one segment

Then two at a time

Then four. . .

This is called *slow start*

TCP Strategies

Slow Start & Congestion Avoidance

It is actually a near-exponential increase in the congestion window over time

It is “slow” in comparison with an earlier version of TCP that started by blasting out segments as fast as possible before the performance of the network was known

In slow start, the increase continues until we reach the current threshold `ssthresh` or returning ACKs are duplicated or timed out

TCP Strategies

Slow Start & Congestion Avoidance

Of course, the rate is also limited by the advertised window of the destination: we can only send the minimum of the current congestion window and the advertised window

Note that the congestion window is a limit set by the sender, while the advertised window is a limit set by the receiver

TCP Strategies

Slow Start & Congestion Avoidance

If we reach `ssthresh` without a problem, we change to the *congestion avoidance* phase

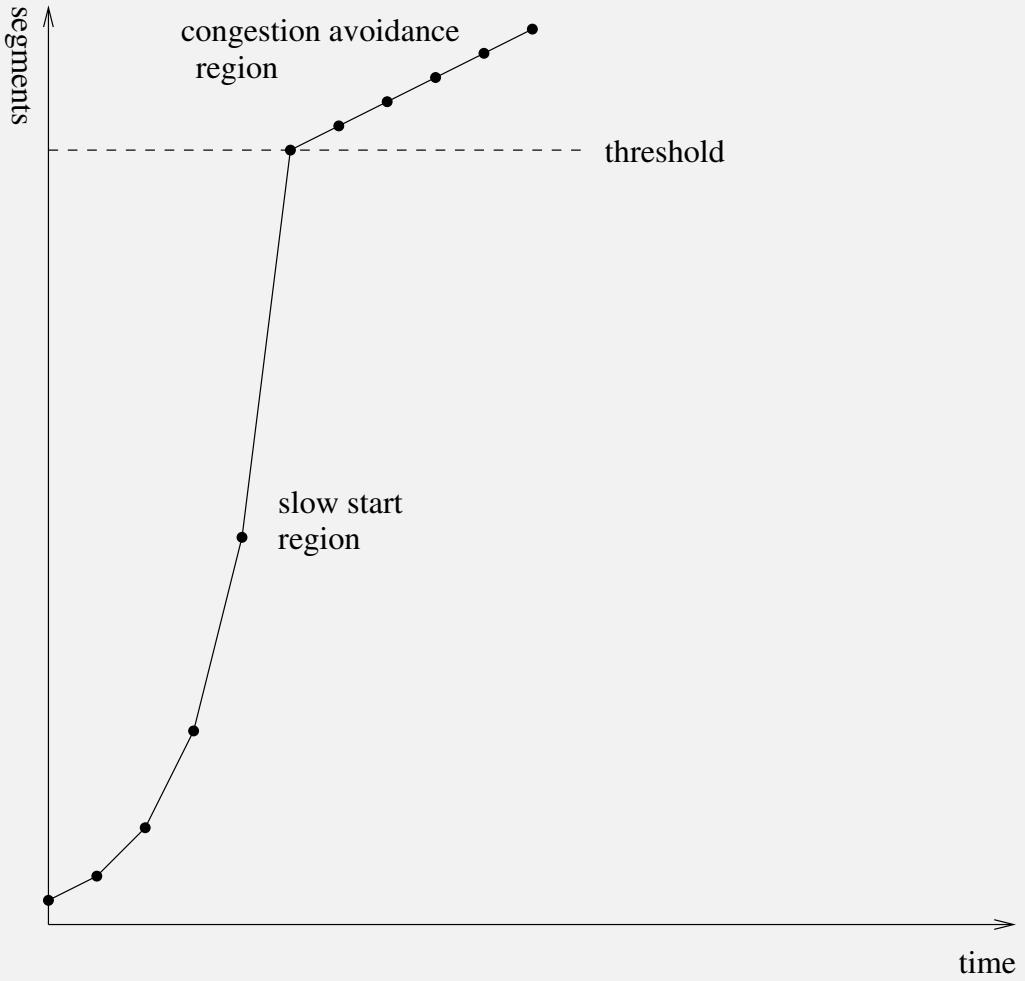
Now we increase the congestion window `cwnd` by one segment for each round trip time (RTT)

So one per burst of segments

This is now a linear increase over time

TCP Strategies

Slow Start & Congestion Avoidance



Slow start and congestion avoidance regions

TCP Strategies

Slow Start & Congestion Avoidance

Eventually the network's limit will be reached and a congested router somewhere will start dropping segments

The sender will see this when either (a) it gets some duplicate ACKs, or (b) there is a timeout waiting for ACKs

Note we might be in either of the slow start or the congestion avoidance phases when congestion occurs: particularly if `ssthresh` was initially set very large, as its often done these days

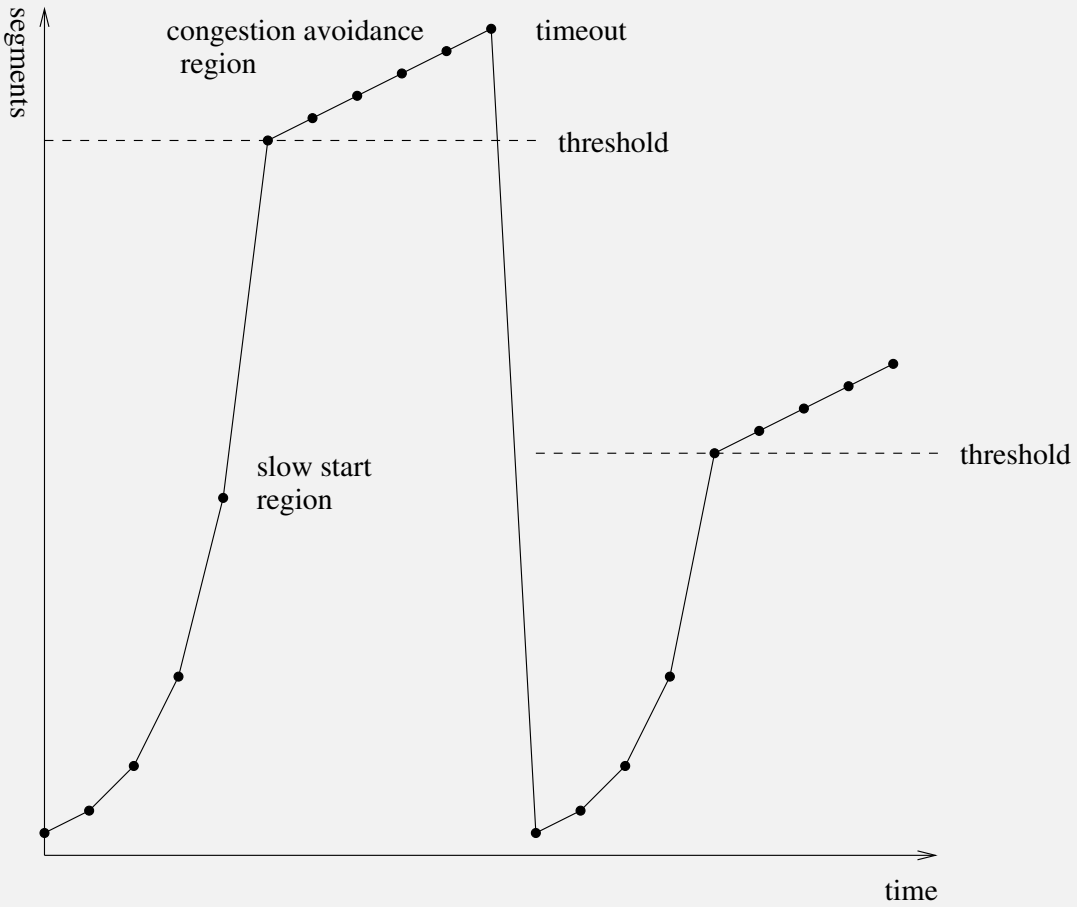
TCP Strategies

Slow Start & Congestion Avoidance

When congestion is detected

- the threshold `ssthresh` is set to half the current transmit size. This is the smaller of the current congestion window and the advertised window. Also, this is rounded up to a minimum of two segments
- if it was a timeout, the congestion window `cwnd` is set back to one segment, and go back into slow start
- when ACKs start coming through, we resume increasing the congestion window again, according to whether we were in slow start or congestion avoidance (i.e., whether `cwnd` is less than `ssthresh` or not)

TCP Strategies



Converging on the optimum rate

The sender eventually converges on a rate that is neither too fast, nor too slow

TCP Strategies

Slow Start & Congestion Avoidance

And it is dynamic

If conditions on the network change, it will soon adapt to the new rate, be it faster or slower

If there is no congestion on the network, the rate increases until it reaches the advertised window: the limiting factor is then the destination, not the network

This strategy is very effective: get the flow up quickly, but don't overshoot network capacity. Also, back off quickly and try again when a loss happens

TCP Strategies

Fast Retransmit

As previously mentioned, when an out-of-order segment is received the TCP protocol calls for an immediate (possibly duplicate) ACK: it must not be delayed

Thus the sender will start seeing duplicate ACKs

This is to inform the sender as soon as possible that something is wrong

Jacobson's Fast Retransmit strategy builds on the idea that the receipt of several duplicated ACKs is indicative of a lost segment

TCP Strategies

Fast Retransmit

Recall that the argument is that one or two duplicate ACKs might simply be due to out-of-order delivery, as IP is unreliable

Three or more is taken to mean something is wrong

If this happens, the sender should retransmit the indicated segment immediately: fast retransmit

TCP Strategies

Fast Recovery

Next, Jacobsen says do not go into slow start but do congestion avoidance: this is the *fast recovery strategy*

We don't want slow start as the duplicate ACKs indicate that later data have reached the destination and is buffered there

So data is still arriving (mostly) and we don't want to abruptly cut the flow by doing slow start

Fast Retransmit & Fast Recovery are quite effective at getting the flow going again after a loss

Exercise Read RFC2001 for the details we have not mentioned here

TCP Strategies

Congestion

There have been many tweaks to this basic flow control strategy

- Larger initial `ssthresh`
- Larger initial `cwnd`
- Slow start counting number of segments ACKed, not just the number of ACKs
- Treating duplicate ACKs like a timeout
- On timeout, setting `cwnd` to half `ssthresh`, not just 1 segment
- Fast recovery: wait for the ACK of the entire transmit window before entering congestion avoidance
- Many more

TCP Strategies

Congestion

Exercise Read about other strategies, such as TCP Reno, TCP Vegas, TCP New Reno, TCP Hybla, BIC, CUBIC, etc.

TCP Strategies

Congestion

And other *kinds* of congestion strategy exist and are used

For example, BBR (specifically BBRv3) from Google is not (primarily) loss based, but develops a model of the state of the network by monitoring RTTs and the achieved bandwidth of a connection

It remembers and uses past behaviour as a predictor: not just the current ACK loss behaviour

Of course, this involves a lot of CPU cycles and could not have been done in the early days of the Internet

Exercise Read about this

TCP Strategies

Congestion

Other strategies involve the routers — they are where the congestion is happening, after all!

Particularly *Explicit Congestion Notification* (ECN), which aims to indicate congestion *before* it happens by routers setting flags in the IP TOS/DS header when they think congestion is imminent, so that the hosts get forewarning and can slow down

Exercise Read about ECN and its use of flags in both the IP header and the TCP header

TCP Strategies

Congestion

Exercise Read about Random Early Detection/Drop (RED), which is also used in routers

Exercise We use ICMP to indicate other kinds of errors, but why is it not a good idea to use ICMP when a router drops a packet due to congestion?

TCP Strategies

`tcpdump`

Exercise Use `tcpdump` to watch these strategies in operation. The `netcat` program is an easy way to set up connections and send data

TCP Strategies

Path MTU Discovery

The next strategy we have seen already: it is aimed at getting the largest segment size a connection can handle. But not too large

IP layer fragmentation is expensive, so we employ path MTU discovery: but now we need to look at it from a TCP perspective

TCP has (potentially) more information: namely the optional MSS header sent in the setup handshake

TCP Strategies

Path MTU Discovery

We can send segments of decreasing size, starting with the minimum of the MSS of the sending interface and the MSS announced by the other end, or 536 if the other end did not give an MSS

And with the IP flag DF (Don't Fragment) set

Note the cross-layer activity here!

TCP Strategies

Path MTU Discovery

If an ICMP error “fragmentation needed but DF set” happens during a TCP connection, the congestion window should remain unchanged, but it should only resend one segment before ACKs start appearing again

This is to reflect the fact that it’s not congestion at fault here, but we do need to back off a bit to allow ACKs to start coming through again

It is recommended you try a larger MTU once in a while, e.g., every 10 minutes, as routes can vary dynamically

TCP Timers

Next: TCP has several timers. We have seen

- 2MSL
- Delayed ACK

These are just the start!

TCP Timers

Retransmission Timer

We now consider the timer that determines when to resend in the absence of an ACK: a *retransmission timeout* (RTO)

- too short a time is wasteful on slow but otherwise reliable networks
- too long a time is poor for the data rate

And we want a dynamic behaviour that adapts to changing conditions rather than a simple fixed timeout

TCP Timers

Retransmission Timer

If the network slows down (e.g., heavy other traffic causes less bandwidth for your packets) the timeout should increase

If the network speeds up (e.g., other traffic reduces) the timeout should decrease

Jacobson gave an easy algorithm: keep a variable, the *round trip time* RTT for each connection

RTT is the best current estimate for the time of a segment going out and the ACK returning

If we haven't received an ACK in approximately this time, deem it lost

TCP Timers

Retransmission Timer

In more detail: when a segment is sent, its timer starts

If the ACK returns before the timeout, TCP looks at the actual round trip time M and updates RTT using

$$RTT = \alpha RTT + (1 - \alpha)M$$

α is a smoothing factor, usually 7/8 for easy arithmetic

TCP Timers

Retransmission Timer

Thus RTT increases or decreases smoothly as conditions change and doesn't get too upset by the occasional straggler that is unusually late (or early)

Next, we need to determine a timeout interval given RTT

This should take the standard deviation of the RTT into account: if the measured RTTs have a large deviation it makes sense to have a larger timeout

True standard deviations are tricky to compute quickly (square roots), so Jacobson suggested using the *mean deviation*

TCP Timers

Retransmission Timer

Mean deviation:

$$D = \beta D + (1 - \beta)|RTT - M|$$

D is close to the standard deviation and is much easier to calculate quickly

A typical value for β is 3/4

TCP Timers

Retransmission Timer

The timeout value is set to

$$T = RTT + 4D$$

The 4 and the values for α, β were found to be good in practice

When sending a segment (or, in practice, a burst of segments) set the timer to expire after time T

TCP Timers

Retransmission Timer

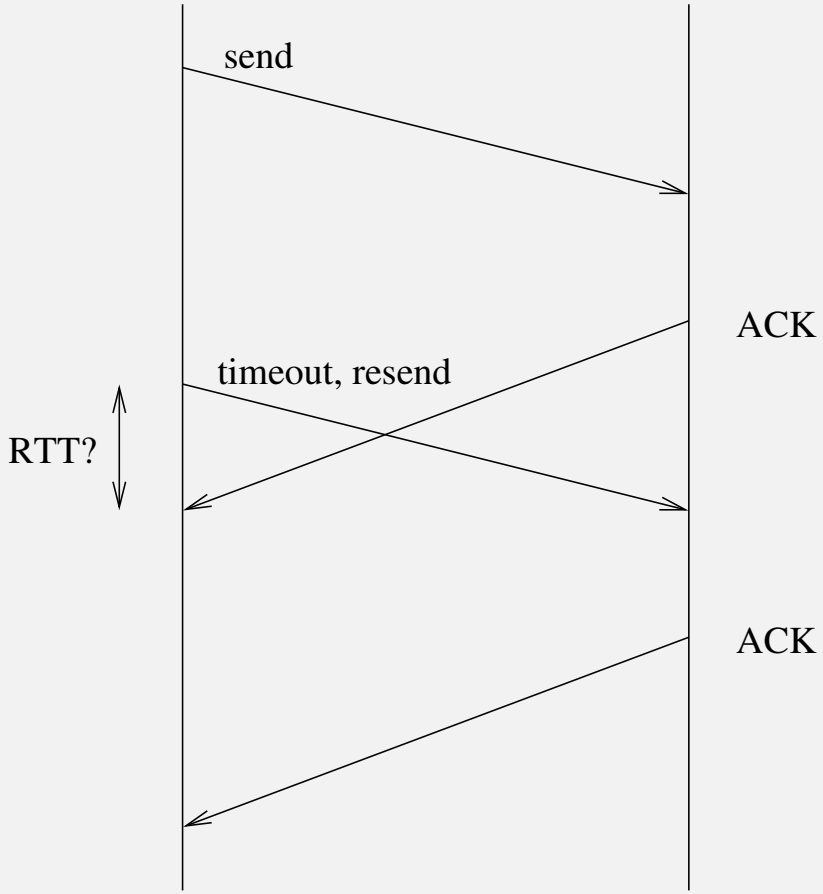
What if the timer expires before the ACK is received?

- we resend the segment, of course
- but we also need to update RTT somehow

But we can't use RTT of the resent segment as we might get the somewhat delayed ACK of the original segment, not of the resent segment

TCP Timers

Retransmission Timer



Retransmission Ambiguity

This is the *retransmission ambiguity problem*

TCP Timers

Retransmission Timer

The measured RTT would be much too small

Karn's algorithm is to double the timeout T on each failure, but do not adjust RTT

When segments start getting through normal RTT updates continue and RTT quickly reaches the appropriate value

This doubling is called *exponential backoff*

Alternatively, as is common these days, we have the option header timestamp and this solves the retransmission ambiguity directly

TCP Timers

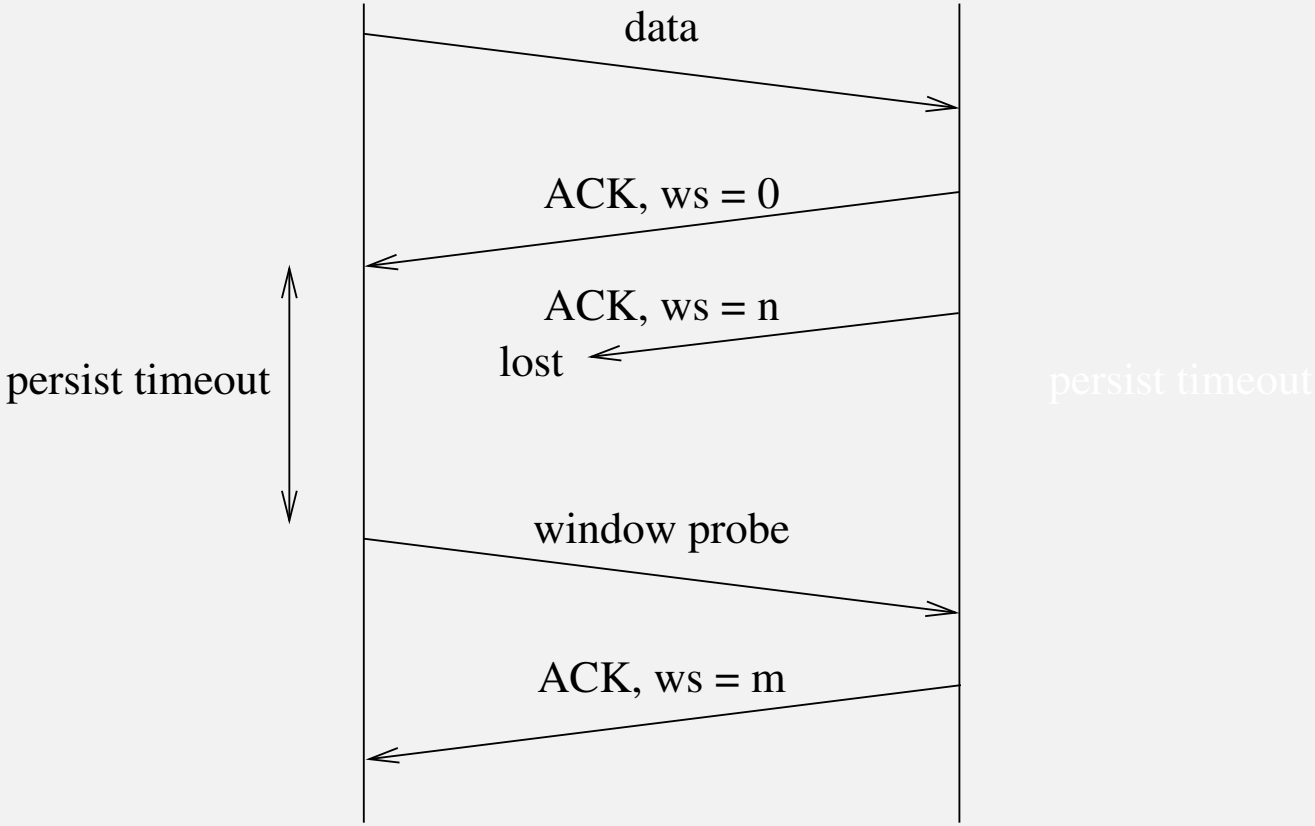
Persist Timer

The next timer in TCP is the *persist timer*, sometimes called the *persistence timer*

Its role is to prevent deadlock through the loss of window update segments

TCP Timers

Persist Timer



Persist timer

A sends to B; B replies with an ACK and a window size of 0; A gets the ACK and holds off sending to B; B frees up some buffer space and sends a window update to A; This is lost; Now

TCP Timers

Persist Timer

The persist timer starts with something like 1.5 sec, doubling with each probe and is rounded up or down to lie within 5 to 60 seconds

So the timeouts are 5, 5, 6, 12, 24, 48, 60, 60, 60, . . .

The persist mechanism never gives up, sending window probes until either the window opens, or the connection closes

The persist timer is unset when a non-zero window is received

TCP Timers

Keepalive Timer

Yet another timer in TCP is the *keepalive*

This one is an optional part of the TCP/IP standard, and some implementations do not have it as it is occasionally regarded as controversial

When a TCP connection is idle no packets flow between source and destination

So part of the path could break and be restored and the connection is none the wiser

This gives us a bit of resilience against flaky networks

TCP Timers

Keepalive Timer

On the other hand, sometimes a server wants to know if a client is still alive: each client TCP connection uses some resources in the server (buffers, timers, etc.)

If the client has crashed these resources could better be used elsewhere

To do this the server sets a keepalive timer when the connection goes idle

A typical value is 2 hours

TCP Timers

Keepalive Timer

When the timer expires, the server can send a *keepalive probe*

This is simply an empty segment (i.e., no data)

If the server gets an ACK, everything is OK

If not, the server might conclude the client is no longer active

TCP Timers

Keepalive Timer

There are four cases

1. the client is up and running: the keepalive probe is ACKed and everybody is happy. The keepalive timer is reset to 2 hours
2. the client has crashed or is otherwise not responding to TCP: the server gets no ACK and resends after 75 seconds. After 10 probes, 75 seconds apart, if there is no response, the server terminates the connection with “connection timed out” sent to the server application

TCP Timers

Keepalive Timer

3. the client has crashed and rebooted. The client gets the probe and responds with a RST. The server gets the RST and terminates the connection with “connection reset by peer” sent to the application
4. the client is up and running, but is unreachable, e.g., broken routing. This is indistinguishable from case 2, so the same events ensue

TCP Timers

Keepalive Timer

There are several reasons not to use keepalive

- they can cause a generally good connection to be closed because of a temporary failure of a router
- they use bandwidth
- some network operators charge per packet

TCP Timers

Keepalive Timer

The latter two are not particularly good arguments as the cost is just a couple of packets every 2 hours

It is usually possible to disable keepalive in the application: some people think that keepalive should not be in the TCP layer, but should be handled by the application layer (i.e., the non-existent session layer)

TCP Strategies

Many other strategies to improve throughput have been proposed

Some have been widely adopted

Exercise Read about the problems of *long fat pipes*

Exercise Read about Protect Against Wrapped Sequence numbers (PAWS), Selective Acknowledgement (SACK)

TCP Extensions

Exercise Multipath TCP (MPTCP) has been suggested both for extra performance, failover and for mobile hosts that roam between, say, cellular and Wi-Fi (used in iOS7). It layers one MPTCP connection over one or more TCP connections, e.g., using both the cellular and Wi-Fi links simultaneously for one MPTCP connection

Exercise And potential alternatives to TCP. Read about TCP for Transactions (TTCP), Stream Control Transmission Protocol (SCTP), Datagram Congestion Control Protocol (DCCP)

TCP Alternatives

QUIC (originally “quick UDP Internet connection”, now just a name, not an acronym) is a Google-originated alternative to TCP (RFC9000)

Originally designed as a transport layer for HTTP/3 (the next version of HTTP), QUIC can be used as a general transport protocol

It is reliable, connection oriented, has congestion control, is encrypted and authenticated and is transmitted within UDP datagrams (port 443, mostly)

TCP Alternatives

The last is important as routers have a tendency to mess with (or drop) packets if they don't recognise the protocol

There have been several new protocols in the past that have failed to gain popular use as routers would not recognise them

In fact, the QUIC header is encrypted (inside the UDP packet) to prevent routers inspecting or trying to modify it

TCP Alternatives

Note: QUIC uses UDP purely to avoid router problems: it would be better to layer directly over IP, but history won't let us do that

QUIC is *not* a lightweight protocol: it is as heavyweight as TCP+TLS

It is "quick" in the sense of "fast", not "simple"

TCP Alternatives

Support for QUIC is growing in OSs and applications, for example the Chrome browser uses QUIC whenever possible to fetch Web pages

It has a 3 way opening handshake, like TCP, but this handshake also negotiates encryption

This saves time over the current schemes that open TCP and then establishes encryption (see TLS, later)

TCP Alternatives

Multiple data streams are multiplexed over a single connection, again saving time over TCP that would need to start up a connection for each stream

For example, a Web page might fetch dozens of items (text, images, JavaScript, . . .) from the same server

These could all be sent within a *single* QUIC connection

TCP Alternatives

Current browsers do try to multiplex multiple streams over a single TCP connection, but this causes problems as an error in one stream causes TCP's error mechanisms to kick in, affecting *all streams* in the connection, even if the other streams had no error in themselves

QUIC does this multiplexing more efficiently, never stopping a good stream within a connection

QUIC manages errors at the stream level, not the connection level

TCP Alternatives

And:

- more sophisticated ACK mechanisms
- connection migration, e.g., WiFi to cellular
- sophisticated flow control (still under development)
- and lots of other stuff building on the knowledge gained since TCP was first invented

TCP Alternatives

QUIC is growing, but it will be a long time before it replaces TCP (lots of code to rewrite!)

And TCP with TLS has had decades of tuning, so QUIC has a lot of work to do to catch up

Exercise Read about how QUIC reduces connection overheads and about the *head-of-line blocking* problem

Exercise Read about SPDY, the predecessor to QUIC, and its relationship to HTTP/2

Exercise Read about the *middlebox* (router) problem and why it means that new protocols will have a hard time on the Internet

UDP Alternatives

Exercise And don't forget UDP: UDPLite, RUDP, UDT, etc.

TCP

TCP is a huge success: from 1200 bits/sec telephone lines to gigabit networks and beyond it has turned out to be massively flexible and scalable

It took a lot of work, though!

TCP

Here is a small part of the output from `ss -io` (socket statistics) on a Linux machine:

```
tcp   ESTAB 0 0 172.16.2.1:34956  34.117.14.220:https
timer:(keepalive,31sec,0)
ts sack cubic wscale:7,7 rto:220 rtt:18.341/0.5 ato:40 mss:1368
pmtu:1420 rcvmss:647 advmss:1368 cwnd:2 ssthresh:7
bytes_sent:7179 bytes_retrans:240 bytes_acked:6939
bytes_received:6747 segs_out:515 segs_in:508 data_segs_out:198
data_segs_in:188 send 1.19Mbps lastsnd:28652 lastrcv:29228
lastack:28632 pacing_rate 2.39Mbps delivery_rate 634kbps
delivered:191 app_limited busy:32268ms retrans:0/8
rcv_space:13800 rcv_ssthresh:64156 minrtt:17.318
```

DNS

Next: back to IP addresses (again!)

IP addresses (v4 or v6) are bunches of bits: but we are used to addressing machines by names such as `linux.bath.ac.uk`

The Internet would probably be a lot less easy to use if we had to refer to `212.58.233.253` rather than `www.bbc.co.uk`

The IP addresses are essential as they are hardware independent and have structure that aids routing

But the names are what makes things like the Web usable

Note we now have *three* kinds of addresses: physical, network and human

DNS

So we have a repeat of the hardware-software address gap that ARP is for: there must also be a mechanism for turning `linux.bath.ac.uk` into `138.38.3.40`

In the early Internet every host had a file containing the names and addresses of all machines on the Internet

A simple look into this table sufficed

We can't do that now, though!

Exercise The file lives on as `/etc/hosts` under Unix. Look at this file

DNS

An ARP-style discovery broadcast would have to go to the entire Internet, so this approach is also infeasible

So there is another protocol, the *domain name system* (DNS) to do this

There is no sensible, secure and economic way to have a single database that contains all the names and addresses on the Internet, so this information has to be distributed amongst a large number of machines called DNS servers

That is, machines running a DNS server program

Aside

In the usual way, when talking about these things, we blur the distinction between the service and the host it lives on. The host might well be running other services, such as DCHP, as well

Exercise Find out the services your home Access Point runs

DNS

Names are hierarchical: `linux.bath.ac.uk` is a name of a machine in the *domain* `bath.ac.uk`

`bath.ac.uk` is a network in the domain `ac.uk`, and so on

`ac.uk` is the name for the JANET network, and is in the domain `uk`

`uk` is in the domain `.` (*dot* or *root*)

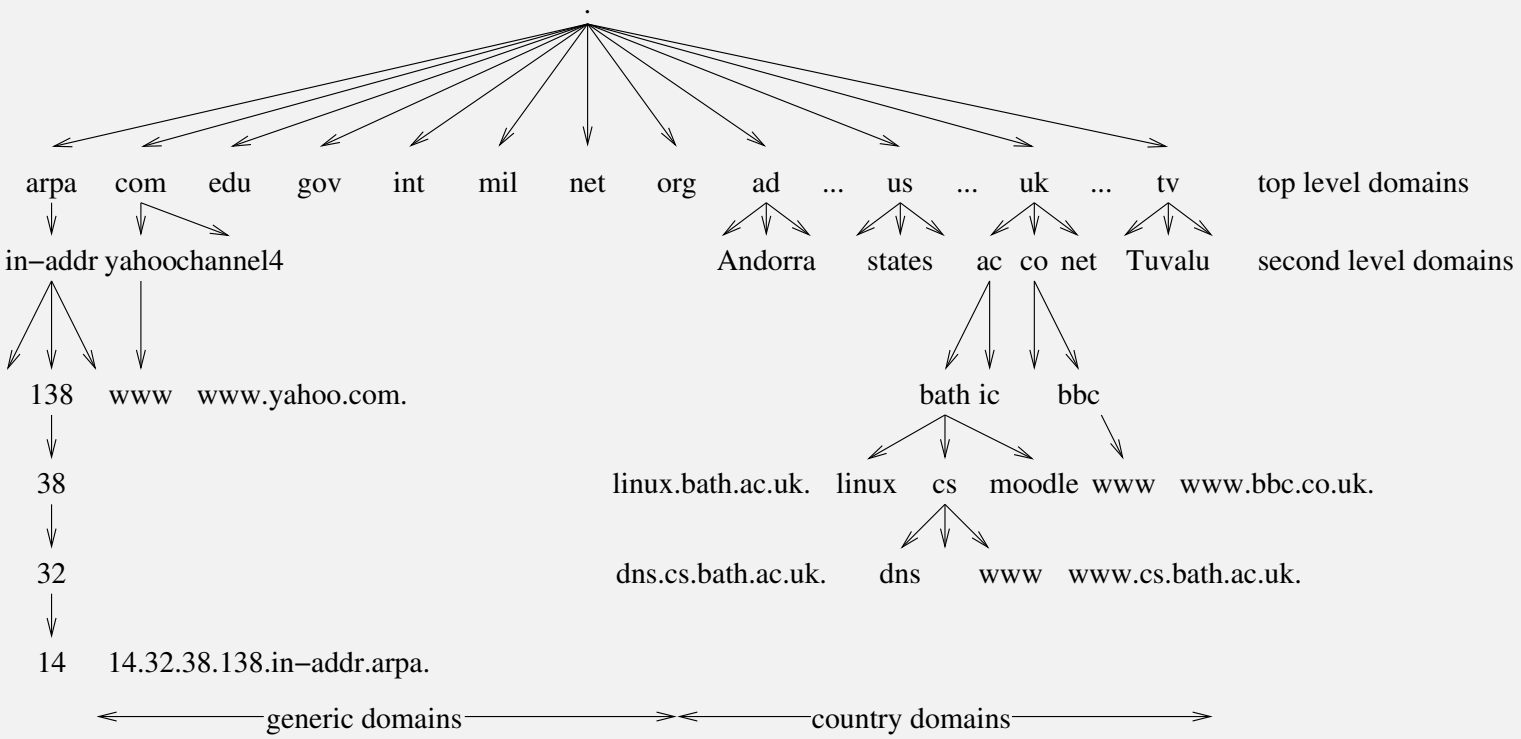
Each node in this tree is called a *label*

DNS

`www.llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch.com`
is a valid name

Labels may be up to 63 bytes long

DNS



The DNS hierarchy

DNS

It is this tree we use to distribute the database

The root of the tree is called `.` (dot) and this label is currently managed by the *Internet Corporation for Assigned Names and Numbers* (ICANN)

They manage the *top level domains* (TLDs), including `com` and `uk`

“Manage” means they keep the lookup tables for that level and they say who can get labels in the next level

They get to charge money for labels in the next level

But the important bit is that they *delegate* management of lower labels to other organisations

DNS

Labels under `uk` are managed by the UK *Network Information Centre* (NIC), currently run by a UK company called Nominet

Note that ICANN gets to say that Nominet is in charge of `uk`

Nominet delegates labels under `ac.uk` to the *Joint Information Systems Committee* (Jisc) (previously *United Kingdom Education and Research Networking Association* (UKERNA))

Labels under `bath.ac.uk` are managed by DDAT for the University of Bath

Labels under `cs.bath.ac.uk` are managed by the ~~Department of Computer Science~~ DDAT

DNS

At each level, the relevant organisation keeps a list of labels it has delegated and who is responsible for them

At the lowest levels, for the leaves of the tree (the hosts), the relevant organisation keeps the name-to-IP-address mapping of the hosts

So, starting a dot, we can work our way down the tree to find the machine we want

DNS

All this is done by *name servers*

These are just computers that run the DNS protocol

E.g., `ns1.bath.ac.uk` is (the name of) a name server for the University

Also there is `ns2.bath.ac.uk`, for resilience and spreading load

They contain replicas of the same information

A few years ago, there was a convention of having an off-site replica, too, e.g., Bath used `ns2.ja.net`

DNS

DNS servers do two things

(a) Keep the table of name to IP address mappings for the domain

(b) Help the local hosts when they need to do a lookup of non-local addresses

The former is “just” a lookup in a database

If a host on the Bath network needs to look up a local name, `linux.bath.ac.uk`, say, it sends a request to one of the local servers, e.g., `ns2.bath.ac.uk`

And the local server will look it up and return the answer

DNS

How does that host know which machine to ask? In particular, how does it know the local DNS server's IP address?

Every host has the IP addresses of one or more local nameservers, typically in a file (e.g., `/etc/resolv.conf`) that was set up by an administrator or was created by DHCP (or SLAAC)

So it can send the DNS request to one of these servers

DNS

The second function is more interesting: we shall look at an example

If a host in the University requires a name lookup of a non-local name, say `news.bbc.co.uk`, it sends a DNS request to the **local** DNS server, `ns1.bath.ac.uk`, say

DNS

In this example, the Bath server does not have responsibility for the name `news.bbc.co.uk`, but it will helpfully do a lookup for us. It will run down the DNS tree in a *recursive lookup*

The Bath server `ns1` sends a *start of authority* (SOA) request to a random *top level* server to find who is responsible for the `uk` label

DNS

At the top level, there are about 80 machines dotted about the world that serve the DNS root; they all have identical information

They have names like `a.root-servers.net` (`198.41.0.4`), `b.root-servers.net` (`170.247.170.2`), and so on

And the local nameserver has the IP addresses of these machines in a file: otherwise we would never get started!

The selected top level server responds to the Bath nameserver with something like `ns1.nic.uk`, together with its IP address

This a machine that is responsible for `uk` labels

DNS

The Bath server now sends a SOA request to `ns1.nic.uk` for `co.uk`

It gets a reply `ns1.nic.uk`. It just so happens this is responsible for both `uk` and `co.uk`

So the server sends a SOA request to `ns1.nic.uk` for `bbc.co.uk`

It gets `ns.bbc.co.uk`

DNS

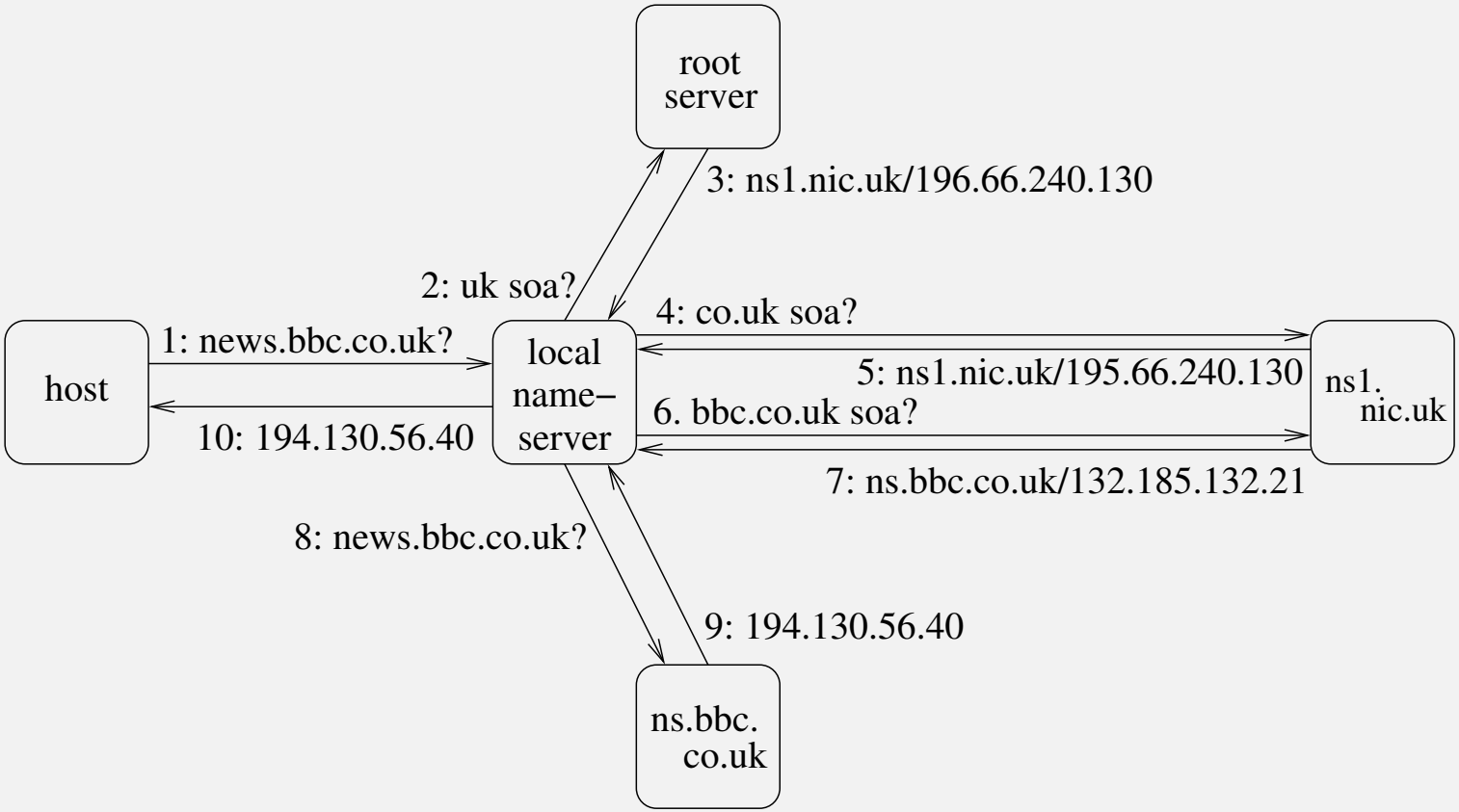
It now knows the server reponsible for the address we want

The server sends an *address* (A) request for `news.bbc.co.uk` to `ns.bbc.co.uk`

It gets the IP address `194.130.56.40`

Finally, the Bath server can relay this back to the original requesting host

DNS



DNS lookup

DNS

Of course, all these responses are cached by the local server so that it doesn't have to go through a complete lookup every time

The next request for `news.bbc.co.uk` can be answered directly by the local nameserver

Similarly, given a request for `www.bbc.co.uk`, it can go directly to `ns.bbc.co.uk`

DNS

As name mappings are not permanent, each reply has a *time to live* attached to it: this indicates how long the server should keep the information before asking again

Stable, long lived associations will have a long TTL

Short lived associations will have a short TTL

DNS

The original asking host could do this lookup process itself but the benefit of using the organisation's name server is that it might well have done some or all of the steps already for some other request and cached them

This means a faster response and a decrease in network traffic

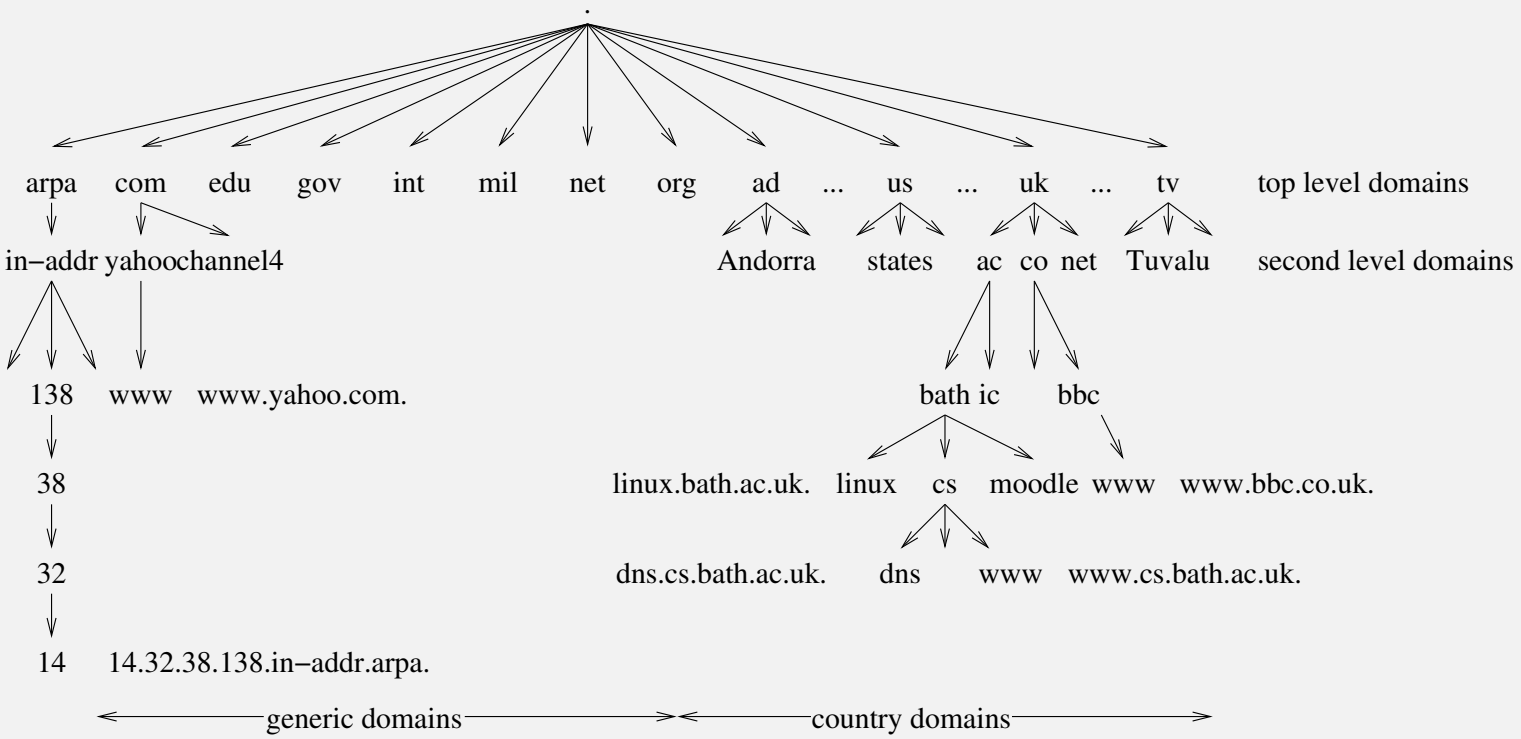
And less work for the host

DNS

Sometimes we want to do the reverse lookup: given an IP address find a name

There is a part of the tree dedicated to this with TLD `arpa`

DNS



The DNS hierarchy

DNS

If we have the IP address `138.38.32.14` we can do a request for `14.32.38.138.in-addr.arpa`

The same kind of recursive lookup as before will lead us to finding that Bath is responsible for `38.138.in-addr.arpa`

And a couple more steps takes us to a *pointer* (PTR) request for `14.32.38.138.in-addr.arpa` from the Bath server

We get `clan.bath.ac.uk`

DNS

DNS does

- A address: name to IP address
- PTR pointer: IP address to name (IPv6 uses the `ip6.arpa` branch)
- AAAA address: name to IPv6 address
- SOA start of authority: name to responsible name server
- MX mail server: name to a mail server for that domain. E.g., `bath.ac.uk` used to have mail server `138.38.32.14`

And many more: about 50 in total, though few are used frequently

DNS

DNS is remarkably successful and scales easily to the billions of hosts on the Internet

When a new organisation plugs into the Internet, it brings along its own DNS servers, thus increasing the scale of DNS

Lookup usually takes a few milliseconds: caching of all levels in the local DNS server is a big help here

DNS is actually a many-many relationship of names and addresses

- One address can have several names

DNS

Both `news.bbc.co.uk` and `newswww.bbc.net.uk` resolve to `212.58.226.33`

```
news.bbc.co.uk.          1619    IN  CNAME   newswww.bbc.net.uk.
newswww.bbc.net.uk.      77      IN  A       212.58.226.33
```

`news.bbc.co.uk` is an *alias* for the real *canonical name* `newswww.bbc.net.uk`

This allows us tricks, like moving the Web server to different hosts, even different ISPs or different countries, but keeping its public name the same

DNS

- One name can have several IP addresses associated. This allows *load sharing*, e.g., a Web server can actually be several machines spread about anywhere in the world

DNS

```
www.microsoft.com.       68      IN  CNAME   toggle.www.ms.akadns.net.
toggle.www.ms.akadns.net. 285    IN  CNAME   g.www.ms.akadns.net.
g.www.ms.akadns.net.     285     IN  CNAME   lb1.www.ms.akadns.net.
lb1.www.ms.akadns.net.   285     IN  A       207.46.19.190
lb1.www.ms.akadns.net.   285     IN  A       207.46.19.254
lb1.www.ms.akadns.net.   285     IN  A       207.46.192.254
lb1.www.ms.akadns.net.   285     IN  A       207.46.193.254
```

`www.microsoft.com` is an alias: its canonical name is `toggle.www.ms.akadns.net`; But that is an alias for `g.www.ms.akadns.net`; And that is an alias for `lb1.www.ms.akadns.net`; And that refers to four different addresses

DNS

And a DNS server can give out different lookups dependent on who is asking

A recent lookup for `www.microsoft.com` returned `104.78.177.250` from one location, and `2.18.85.172` from another

This allows for load sharing; and also can be used for *geofencing*: giving different services to clients in different place, e.g., videos that are licensed only for certain regions

DNS

Exercise Redo these lookups to see how they currently turn out

Exercise Compare having DNS give out multiple IP addresses for a given name against giving out different addresses for different clients against using anycast addresses

DNS

In C programs, if we need to look up an address (v4 or v6), we can use a simple function call to `getaddrinfo()` that hides all this complexity from us

Other languages will have similar APIs

Exercise Old code using the obsolete v4-specific `gethostbyname()` is one of the sticking points in the transition to IPv6. Read about this

Under Linux the `dig` tool can be used to do direct lookups

DNS

```
% dig news.bbc.co.uk
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15281
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;news.bbc.co.uk.                        IN      A

;; ANSWER SECTION:
news.bbc.co.uk.         193     IN      CNAME   newswww.bbc.net.uk.
newswww.bbc.net.uk.     76      IN      A       212.58.226.73

;; AUTHORITY SECTION:
bbc.net.uk.             85129   IN      NS      ns0.thdo.bbc.co.uk.
bbc.net.uk.             85129   IN      NS      ns0.rbsov.bbc.co.uk.

;; ADDITIONAL SECTION:
ns0.thdo.bbc.co.uk.     9490    IN      A       212.58.224.20
ns0.rbsov.bbc.co.uk.    9490    IN      A       212.58.227.47
```

DNS

Quite often a DNS server will reply with more information than we requested, e.g., the lookup of the CNAME `newswww.bbc.net.uk` in the above

This means we don't have to do an additional query to get the actual IP address we were looking for

DNS

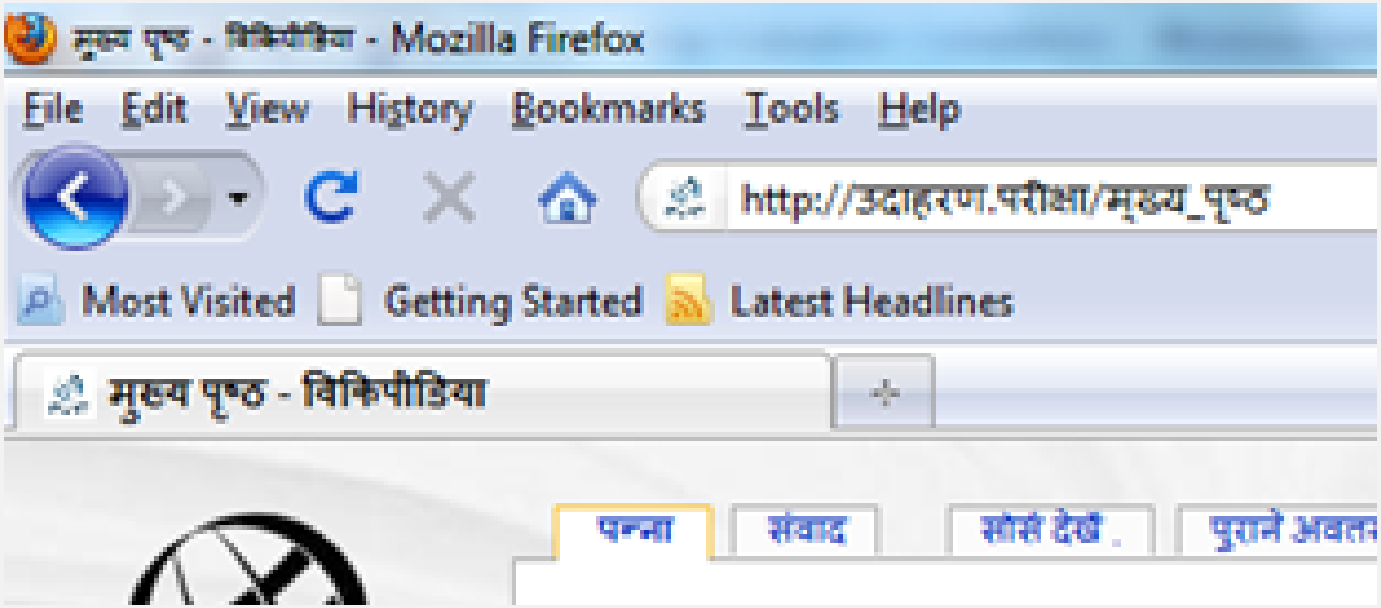
```
% dig -x 212.58.226.73
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32413
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 3

;; QUESTION SECTION:
;73.226.58.212.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
73.226.58.212.in-addr.arpa. 50081 IN    PTR     newslb305.telhc.bbc.co.uk.

;; AUTHORITY SECTION:
226.58.212.in-addr.arpa. 50081  IN      NS      ns1.thdo.bbc.co.uk.
226.58.212.in-addr.arpa. 50081  IN      NS      ns1.thny.bbc.co.uk.
226.58.212.in-addr.arpa. 50081  IN      NS      ns1.bbc.co.uk.
226.58.212.in-addr.arpa. 50081  IN      NS      ns.bbc.co.uk.

;; ADDITIONAL SECTION:
ns1.bbc.co.uk.          311     IN      A       132.185.132.21
ns1.thny.bbc.co.uk.     32106   IN      A       212.58.227.48
ns1.thdo.bbc.co.uk.     33051   IN      A       212.58.224.21
```

DNS

```
% dig +trace www.google.com
.                       180695  IN      NS      E.ROOT-SERVERS.NET.
.                       180695  IN      NS      J.ROOT-SERVERS.NET.
.                       180695  IN      NS      I.ROOT-SERVERS.NET.
...
com.                    172800  IN      NS      A.GTLD-SERVERS.NET.
com.                    172800  IN      NS      B.GTLD-SERVERS.NET.
com.                    172800  IN      NS      C.GTLD-SERVERS.NET.
...
google.com.             172800  IN      NS      ns1.google.com.
google.com.             172800  IN      NS      ns2.google.com.
google.com.             172800  IN      NS      ns3.google.com.
...
www.google.com.         604800  IN      CNAME   www.l.google.com.
l.google.com.           86400   IN      NS      b.l.google.com.
l.google.com.           86400   IN      NS      d.l.google.com.
...
```

DNS

DNS labels can be in arbitrary character sets, not just Latin



A non-Latin DNS name

From `blog.icann.org`

DNS

DNS is a very successful protocol: fast and it's relatively easy for system administrators to manage the DNS servers

DNS names are a big source of money, so often a source of contention over who should be in control of what

Exercise Read about the controversies behind the introduction of new top-level DNS labels like `me` and `search`

DNS

Exercise Try looking up

- AAAA for `bath.ac.uk`
- AAAA for `facebook.com`
- SOA for `bath.ac.uk`
- A for `moodle.bath.ac.uk`
- And so on

DNS

Exercise And

- MX for `bath.ac.uk`

What is the Uni's mail server physical location?

What are the privacy/security/legal aspects?

Then try MX for `bath.edu`, an address the University uses for alumni (graduates and past staff)

DNS

Exercise Read about the public DNS servers like `1.1.1.1`, `8.8.8.8` and others; why would you want to use them over your local DNS server?

Exercise What are the privacy/security/legal aspects of using public servers?

Exercise Find out how to buy a DNS name

DNS

DNS requests are normally sent using UDP, and fit in a single UDP datagram

When the reply fits within 512 bytes, UDP is used for the reply by the server

To keep datagrams small, if the reply is more than 512 bytes, the server sends a reply with a “truncated” flag set, and the client resends the request but using TCP

Now, UDP is fast but unreliable, and 512 byte datagrams won’t be fragmented (recall: must be able to send 576 bytes IP), so there is little complication if a DNS datagram is lost: the source will just send a new request

DNS

UDP is preferred as it is fast with little overhead; while a TCP connection has a considerable setup cost

So small and fast wherever possible; but slower and reliable if needed

DNS

DNS is good, but has problems

- There is no security or authentication: if I get a reply saying that `138.38.32.14` is the address for `www.bath.ac.uk` can I trust this?
- Some server on the lookup path might have been subverted and made to hand out the wrong address; or a router may have carefully rewritten a DNS reply
- Then I could receive a spoof IP address leading to someone else's web page: a problem if, say, I was looking for the page of a bank or shop
- The spoof page could be made to look identical to the real bank web page, inviting me to enter my id and password

DNS

Exercise The security company RSA was attacked by DNS spoofing. Read about this

Exercise April 2018: routes to Amazon DNS servers were faked so that DNS requests were sent to fake servers. Read about this

Exercise Your home connection probably uses your ISP as a DNS server. ISPs have been known to rewrite DNS replies. They also might block access to other public DNS servers. Read about this

DNS

A partial solution exists in *Secure DNS* (DNSSec), which uses cryptography to authenticate DNS lookups

Not much in use as it was introduced when people still thought that cryptography was too expensive to use

DNS

And neither DNS nor DNSSec provide any privacy: anyone listening can see what hosts you are trying to contact by monitoring your DNS requests

So there has been a move by some people to use DNS over TLS (see TLS later and RFC7858) that encrypts DNS requests and replies

This has a fair overhead over plain DNS, but provides both authentication and privacy

DNS

More surprisingly (at first), there is also DNS over HTTPS (DoH)

This sounds silly as HTTPS is a heavyweight protocol designed to move large Web pages, not lightweight DNS requests

But:

- The overhead can be managed fairly well (see HTTP *Keep-Alive* and TLS reconnections)
- While an uncooperative ISP could block the DNS over TLS assigned port (853) the HTTPS port (443) cannot be blocked without a lot of collateral damage to normal browsing

DNS

Exercise CIDR makes PTR lookups harder as netmasks, and therefore the delegations, are no longer on a byte boundary. Read RFC2317 on how CNAMES are used to solve this problem

Exercise Read about how (or if) DoH is supported in your browser

Exercise Read about *DNS over Twitter* and *DNS over SMS*

Presentation

If I gave you the four bytes

`1010000 1101100 1100001 1101110`

(which are 80, 108, 97, 110 in decimal), what did I mean?

Is this the encoding of an integer?

If so, signed, unsigned, 2s complement?

Least significant byte first or most significant byte first?

Presentation

Or is it floating point number?

Or is it a string of four characters?

In ASCII? Or some other encoding like EBCDIC, or UTF-8?

Presentation

From another point of view:

I want to send `"Plan"` to you. What do I send?

If we both use ASCII to encode characters, I might send four bytes 80, 108, 97, 110

If we both use EBCDIC to encode characters, I might send four bytes 215, 147, 129, 149

If we use some other encoding, it might need more than four bytes

Presentation

What do I do if we use different encodings? Perhaps my machine uses ASCII while yours uses EBCDIC

What do I do if I don’t know what encoding you use?

Presentation

This is the problem of *presentation*

Bits are just bits unless they have some agreed-on meaning

And the agreeing is the difficult part

Particularly as some people forget that not everyone uses the same representations for everything

Presentation

The job of the presentation layer is to ensure that the data at one end of a connection is interpreted in the same way when it reaches the other end of the connection

It is about preservation of *meaning*

So if I send you the number `3.14`, you get the number `3.14`

Even if we use different representations of floating point numbers

Presentation

If I send you the string `"cat"`, you get the string `"cat"`

Even if we use different ways of encoding characters

Even if we are using different programming languages that encode strings in different ways

Presentation

If I send you a picture containing a particular blue, you get a picture with the same blue

Even if we are using different representations of pictures

Even if we are using different picture viewers

Photographers get very wound up about this particular problem!

Exercise Create a plain text (`txt`) file on MacOS or Linux, and view that file on Windows using Notepad. What is happening?

Addendum May 2018: Microsoft has finally fixed this problem, after only 30 years

Presentation

We have many ways of encoding data

For example, how do we encode the letter ’A’? One popular way is to use a 7 bit number, namely 65

The *American Standard Code for Information Interchange* (ASCII) is one standard for encoding letters, digits and various punctuation marks

However, it is not the only standard and that is precisely the problem

Presentation

When the Internet began IBM's *Extended Binary-Coded Decimal Interchange Code* (EBCDIC) was still widely used

The purpose of EBCDIC is the same as ASCII: encoding characters as numbers

The problem is that a file containing the (decimal) byte values

$$80, 108, 97, 110$$

would be interpreted as "Plan" on an ASCII system, but "&%/ >" on an EBCDIC system

In ASCII, the value 108 means the character 'l'

In EBCDIC, the value 108 means the character '%'

Presentation

Philosophy

The *presentation problem* is to ensure that we have the same *meaning* on any system

We can easily copy bits from system to system, but our *interpretation* of those bits changes from system to system

So to make our interpretation consistent we might have to *change the bits*

But not only *how* to change them, but *when*

Presentation

Philosophy

If the file 80, 108, 97, 110 is a text file, we must change the values to ensure consistent interpretation

If this is a list of the IQs of four people, we must *not* change the values

Everything depends on the final interpretation of the data: this is a subtle point and is why presentation issues are often ignored or incorrectly implemented

Presentation

Note that IP *does not address presentation*, and leaves it to the application

This means that presentation must be addressed by the programmer in their all their applications

In the early Internet all the machine were the same, so presentation was not realised to be a problem

Today, things are very different

And programmers are still forgetting this is an issue

Presentation

These days most people have more-or-less settled on ASCII as the encoding to use for simple Latin/Roman letters and digits

So presentation issues are minimal for these kinds of text data

On the other hand, other character sets (Chinese, Russian, Klingon, etc.) are in the ascendant, with the *Universal Coded Character Set* (UCS) plus *Unicode* being the chosen representation

Presentation

UCS/Unicode

UCS (ISO 10646) is a character encoding that uses 31 bits instead of just 7

This gives ample room for all the characters in all the written languages in the world

It is a big table that says “this value represents this character”

Unicode takes UCS and adds details like direction of writing (left-to-right or right-to-left or bidirectional), defining alphabetic orders, which are capital letters, and so on

Presentation

UCS/Unicode

Unicode only uses UCS values from 0 to 10FFFF

A maximum of $17 \times 2^{16} = 1,114,112$ *code points*

A code point can denote a character or a character modifier, e.g., a variant or a combining character like an accent

For example, é is a single character, while é is two code points: e followed by a combining character ’

2,048 code points are excluded (the surrogate values D800–DFFF for backwards compatability with UTF-16, below), so the number of representable characters (more properly: *graphemes*) is just 1,112,064

Presentation

UCS/Unicode

And then there is the *glyph*, the visible rendering of the grapheme in some font: é and é

Code points can be written as “U+hex”, e.g., U+C2A3 for the index of code point ’£’)

Presentation

UCS/Unicode

But using 4 bytes per character would not be appreciated by many programmers since it would

- break the “one character is one byte” assumption many programs make
- make data files four times as large when the original data were encoded in ASCII, and
- the zero byte is often conventionally used to mean “end of string” so a value such as (hex) 12 34 00 78 is open to misinterpretation

Presentation

UCS/Unicode

So some *encoding systems* are defined: they implement UCS but use differing numbers of bytes to encode the index into its big table of characters

Some systems are backwardly compatible with ASCII in the sense that values 00 to 7f are the same as their ASCII equivalents

The simplest method, *Unicode Transformation Format* 32 (UTF-32, also called UCS-4), simply uses four bytes per character and embeds ASCII in UCS by merely adding three 0 bytes before every ASCII byte

Presentation

UCS/Unicode

`tiger` in ASCII is five bytes: 116 106 103 101 114

`tiger` in UTF-32 is 20 bytes: 0 0 0 116 0 0 0 106 0 0 0 103 0 0 0 101 0 0 0 114

 is 0 0 128 1 0 0 134 78

This has the expansion and zero problems

But is convenient if we are working with individual characters (rather than strings) as 32 bit values

For example, indexing into an array of characters is very easy: exactly like indexing into an array of 32-bit integers

Presentation

UCS/Unicode

Less inflationary is UCS-2, that uses *two bytes* per character and prepends a single 0 byte before each ASCII character

This only doubles the size of an ASCII file

Still has the zero problem

Presentation

UCS/Unicode

UCS-2 was devised for an earlier 16 bit coding (now called the *Basic Multilingual Plane*, or BMP), that was soon found to be too small (not enough characters)

UCS-2 can't represent all possible UCS values. Not even all Unicode values

Thus the need for UTF-16 which uses *pairs* of UCS-2 values to extend the encoding range

UTF-16 can represent all Unicode values, but at the cost of some complexity

Presentation

UCS/Unicode

It uses pairs of 16 bit values in the range D800 to DFFF (*surrogate pairs*) to encode the extended values

Given a pair of surrogate values x in the range D800-DBFF and y in the range DC00-DFFF

- Get 10 high bits from x − D800
- Get 10 low bits from y − DC00
- Concatenate these bits to get a 20 bit value
- Add hex 10000 to get the UCS value

Exercise Compare this to byte stuffing

Presentation

UCS/Unicode

The surrogate values (and which is high and low) can easily be identified in a byte stream: important if you are dipping into the middle of a string

It does punch a hole in Unicode from D800 to DFFF that can't be used as characters

The Unicode consortium guarantees never to allocate characters in that range

UTF-16 is quite popular in use, e.g., Java, C# and various versions of the Windows OS use it for their internal representations of strings

Presentation

UCS/Unicode

The most important representation, UTF-8, represents all ASCII (7 bit) values as themselves while still being able to represent the full UCS range

UCS values 00000000 to 0000007F are encoded as single bytes 00 to 7f. Thus an ASCII file is a valid UTF-8 file

So, for example, the byte 3F in UTF-8-encoded a file encodes for UCS index 0000003F

UCS values 00000080 to 000007FF become two bytes 110xxxxx 10xxxxxx. The last 11 bits from the UCS values are copied across

Presentation

UCS/Unicode

So '£', UCS 000000A3, binary

$$00000000\ 00000000\ 00000000\ 10100011$$

becomes 11000010 10100011 (C2A3), since

$$00010/100011 \rightarrow 110/0010\ 10/100011$$

And thus the *two* bytes C2A3 in a file encode the UCS index 000000A3

Presentation

UCS/Unicode

Generally we can encode:

UCS range (hex)	Encoding (binary)
00000000-0000007F	0xxxxxxx
00000080-000007FF	110xxxxx 10xxxxxx
00000800-0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
00010000-001FFFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
00200000-03FFFFFF	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
04000000-7FFFFFFF	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

Presentation

UCS/Unicode

This table is more than UTF-8 requires

The UTF-8 encoding is only defined for values up 10FFFF, for compatibility with Unicode and UTF-16

So only the first four rows of the table

Presentation

UCS/Unicode

Unicode range (hex)	Encoding (binary)
00000000-0000007F	0xxxxxxx
00000080-000007FF	110xxxxx 10xxxxxx
00000800-0000FFFF	1110xxxx 10xxxxxx 10xxxxxx
00010000-0010FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Presentation

UCS/Unicode

A full 31-bit range would require up to 6 bytes to encode

Unicode will need at most four (and so will fit in a 32 bit `int`)

Most common characters only require three or fewer; a majority in use need two or fewer

And ASCII values only require one byte

An ASCII file is already a UTF-8 file and there is no expansion of data when regarding it as UCS

Presentation

UCS/Unicode

- ASCII values represent themselves
- No ASCII character appears as a sub-part of any other character
- The convention of using 0 as end of string still works
- The length of each non-ASCII character is given by the number of leading 1 bits
- When dipping at random into a UTF-8 encoded file it is easy to find the start of the next character: just search until you find a byte starting with bits 0 or 11
- In the same way, if a byte is lost (e.g., discarded as corrupt) it is easy to re-synchronise
- All UCS values can be encoded
- The comparison order of UCS is preserved

Presentation

UCS/Unicode

- UTF-16 does not preserve UCS comparison order
- both UTF-8 and UTF-16 need up to four bytes to represent Unicode values
- UTF-8 is byte order independent
- UTF-16 comes in big (UTF-16BE) and little endian (UTF-16LE) variants as well as plain UTF-16, when files can employ a special *byte order mark* (BOM, U+FEFF) at their start to establish order
- UTF-32 is big endian

Presentation

UCS/Unicode

- UTF-8 is more efficient on Western character sets; UTF-16 is more efficient on Asian character sets (note that most computer code is written in ASCII)
- As they are variable length encodings, neither UTF-8 nor UTF-16 allow indexing directly into a string

The advantages of UTF-8 are such that UTF-16 should be retired, but this may take some time

Presentation

UCS/Unicode

Exercise Have a look at how (or if) your favourite programming language supports UCS or Unicode. E.g., C programmers have `wchar_t`

Exercise A typical programming language has variables syntax that “start with a letter, then letters and digits”. How would this work in Unicode?

Exercise Read about the *Punycode* encoding

Exercise Unicode is split into 17 *planes*. Read about this

Presentation

Presentation for characters seems like a solved problem, but it's not: at the very least we need to get people to actually *use* the standard

Many programmers and programming languages still assume everything is ASCII

And there's a huge amount of legacy code and data out there that assumes ASCII

Exercise Other encodings are available. Find out the encodings used on various web pages from across the world

Presentation

Numbers

Another presentation problem is the byte order used for representing numbers

An integer is typically represented using four bytes: but how those bytes are used varies

Some machines use *big endian* format: this stores the most significant byte of an integer (the *big end*) at the lowest machine address, less significant bytes at increasing addresses

Others use *little endian* format: the least significant byte (*little end*) is stored at the lowest machine address, more significant bytes at increasing addresses

Presentation

Numbers

If a machine receives four bytes 00 00 00 2A, does that mean the integer 42 (hex 0000002A) or the integer 704643072 (hex 2A000000)?

Other arrangements are possible, too

A typical solution so that everyone agrees on order is to pick a single order (*the network byte order*) and always transmit bytes in that order

Presentation

Numbers

When a machine wants to send a value, it converts it to network byte order

When a machine receives a value it converts it to its native order

The *de facto* order used on most networks is big endian

Presentation

Numbers

A big endian machine has nothing to do when sending or receiving

A little endian machine must reverse the order of the bytes as it sends or receives

A little endian machine always converts, even when connected to another little endian machine

This is simpler than having a protocol to negotiate endianness and having separate chunks of code for each combination

Presentation

Numbers

Then there is the problem for other types of numerical data, e.g., floating point

Here there is not only the byte order problem, but which and how many bits are used for exponents and mantissas and so on

Fortunately, most have plumped for the IEEE standard floating point representations

This fixes which bits are used for what, but leaves open the endian question

The floating point endian is usually the same as the integer endian, but doesn't have to be!

Presentation

The End of the Line

It would be easy to think that presentation is easy and is irrelevant or has been solved: not so

For example: how to represent the end of a line in a text file?

- Unix-derived systems use a linefeed (LF, character 10 in ASCII)
- Windows systems use a carriage return (CR, ASCII 13) followed by a LF
- Pre-MacOS X used a single CR

Presentation

The End of the Line

So to copy a file from one system to another you must know whether

- it is a text file and so you must do the translations, or
- it is not a text file, so you should not translate

If we are still fumbling an issue as simple as this, just think on the general case!

Presentation

The End of the Line

Exercise Read about XDR as an encoding system

Exercise Read about the Multipurpose Internet Mail Extension (MIME)

Applications

Next we should talk about the Application layer — but time is too short to talk about things that should be reasonably familiar to you, such as the Web and email

There are very many applications that run over the IP from the well-known things like the Web and email, to the near-invisible (but very important) applications that do everyday things like serving files or controlling industrial devices

Applications

We could easily spend weeks covering application layer protocols, e.g., HTML, the protocol that fetches Web pages; or SMTP, the protocol that delivers email

But Instead we will move to a subject that doesn't have a specific layer, namely security

Exercise Read up on your favourite applications and how they employ IP

Security

IP (both the protocol and implementations) was originally developed in a “safe” academic environment

So little thought was given to security or authentication

And early code left a lot to be desired in programming habits, giving us some fragile implementations

But fast development led to IP’s early acceptance and success

Security

And this also meant experimental and poorly debugged code was rapidly incorporated into a large number of systems

So there are generic bugs that tend to appear in many products

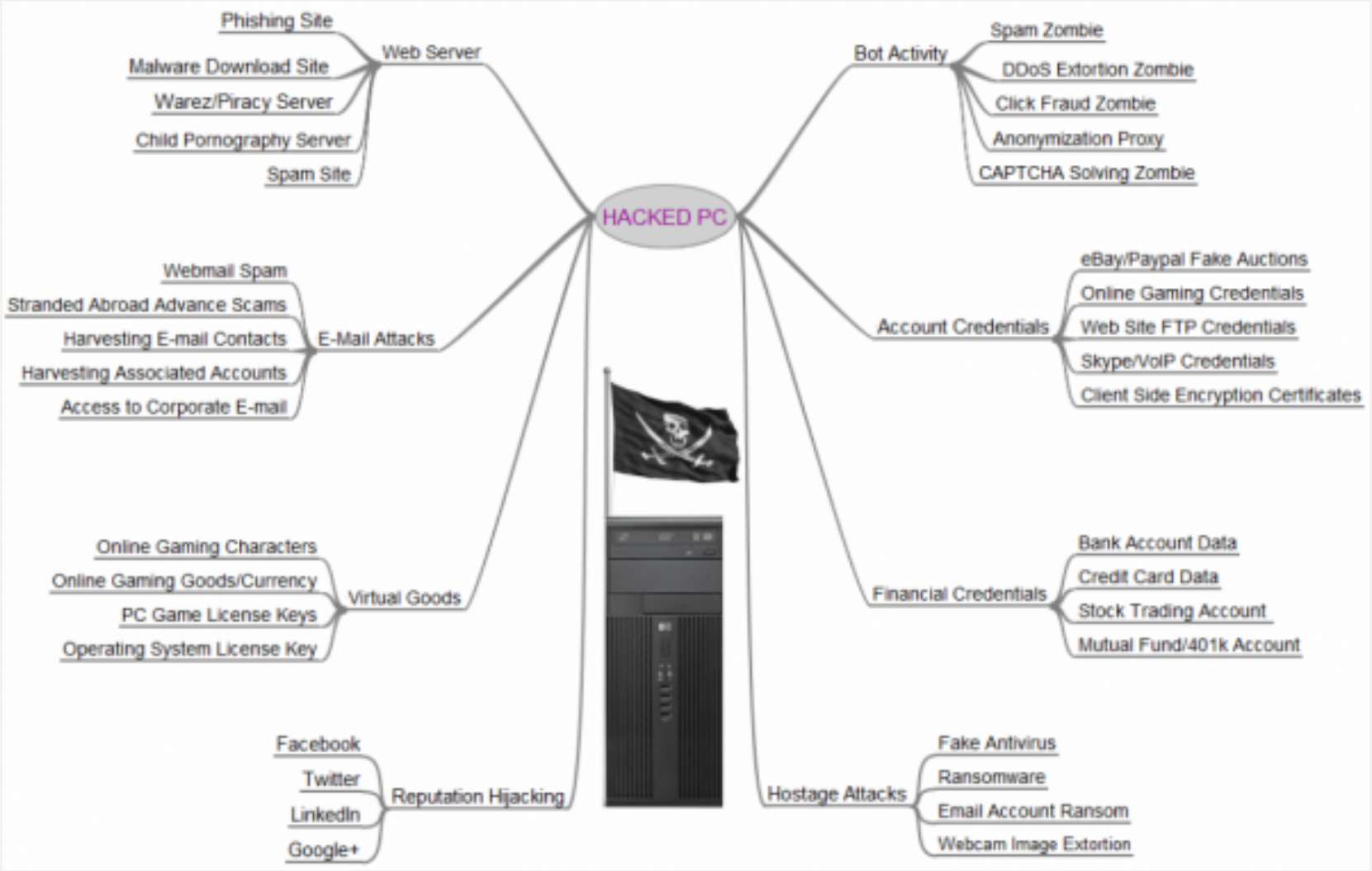
Some are fairly benign, such as TTL being used as a hop count

Security

Other are less so and can be exploited, perhaps to

- crash the machine
- tie up the machine with so much bogus data that real traffic can't get through: called *denial of service*
- gain control over the machine, which can then be used attack a more important target or send spam

Security



Uses for a hacked PC

From: `krebsonsecurity.com`

Security

If someone says “The innocent have nothing to hide”, ask them for their credit card number and date of birth

The point is that we all have things to hide from people who would use information to harm us, financially or otherwise

Remember not all those looking at your traffic have your benefit in mind

Security

Today, the Internet is not a safe place

People are trying to use it for monetary, political or other gain

Or simply wanting to be a nuisance

Thus we must protect ourselves against these issues

Security

Technology plays a large part in this

But psychology of the users is just as important

Why bother attacking a machine when you can attack the human element?

Such as phoning a support engineer or administrator and pretending to be a user who has forgotten their password

Or sending someone an email and getting them to click a link or run some code

Or simply putting fake news in a Facebook post

We shall return to this kind of attack, but shall start with some attacks on the technology

Security

Flooding Attacks

SYN floods. A denial of service attack

A TCP connection starts with a SYN. The server sends a SYN+ACK, which the client ACKs

The server must save a chunk of information about the initial SYN so it can recognise the client ACK as part of the new connection; and the options, like SACK, MSS

Security

A SYN flood is where an attacker sends very many active open SYN segments and never completes the handshake

The SYN segments might come from a single source, but more likely from very many hacked computers in a *distributed* denial of service attack

The hacked machines comprise a *botnet*, controlled by the hacker(s)

The individual hosts are sometimes called *zombies*

Security

Each SYN received consumes resources on the server that are not released until a suitable timeout period has passed

Thus the server can run out of resources and not be able to respond to real connection requests

Security

So the overload reduces the level of service for genuine users, often to zero

This has been used many times, particularly in extortion attacks against commercial (e.g., betting) sites to get them to pay a ransom

These days also used to exert political pressure against companies, people, or governments

A DDOS attack might be several GB/s of SYNs: attacks of TB/s are becoming more common

Security

Since the start of the [ransom DDoS] campaign, show-of-force attacks have grown from 200+ Gbps in August to 500+ Gbps by mid-September, then ballooned to 800+ Gbps by February 2021

Akamai

Security

Note: the return addresses on the bogus SYNs will be forged to implicate some other machine(s) in the attack

And the server's handshake ACKs would go to it, instead

Thus flooding a secondary target or targets

Security

Remedies include the server more aggressively dropping half-open connections when resources are low

Say oldest first, or at random

Real connections might get dropped, but since most of the SYNs are bogus, the probabilities are that attack connections are dropped

Security

Alternatively, use *syncookies*

Store no information for a new connection on the server, but encode it in the server's initial sequence number (ISN) for this connection

So the ISN is not random, but now encodes some information: it is called a *syncookie*

When (or if!) the client ACK gets back, we can decode the returned sequence number to retrieve the information

Now resources can safely be allocated to this presumably valid connection

This is good as it consumes no resources in the server until they are definitely needed

Security

But it is tricky to encode enough information in the 32 bits of the ISN

And it must be encrypted to prevent spoofing

Also it is not big enough to include any negotiated options, such as “SACK available”

So syncookies are only used when the load gets high

Optional features, like SACK, are not used under SYN attack

The loss of SACK is no big deal when we have to cope with a SYN flood

Security

UDP can be used, too

In a *UDP Flood* attack, the attacker(s) simply send very many UDP packets to the victim

The victim OS is then overwhelmed by the need to read and process the packets and respond to them by returning an ICMP Destination Unreachable

A mitigation is to have a limit on the rate of ICMP error returns

Exercise Read about the *Low Orbit Ion Cannon* (LOIC)

Security

Recent botnets have used the *Internet of Things* (IoT), which is connected devices like security cameras, thermostats, doorbells, child monitors and so on

They are often poorly secured, are still using default passwords, or are running old, vulnerable software

The *Mirai* botnet has been implicated in a DDOS attack of over 1TB/s

This was *DNS amplification* attack: the subverted devices make DNS lookup requests to servers with a reply address forged to that of the victim

Security

The DNS replies, which are much larger than the requests, are sent to the victim, causing a DOS

And because the packets are coming from DNS servers, it is again hard to tell who initiated the attack

This is another flooding attack using UDP

There are similar flooding attacks using other public services (such as time servers (NTP) and directory servers (LDAP)) exist

Security

“The ’S’ in IoT stands for security”

Anon

Security

Implementation attacks

These exploit bugs in IP implementations

Some hosts were vulnerable to oversized ping packets: the *Ping of Death*

These were sent as forged fragments that, when reassembled, were much larger than expected and overflowed OS buffers in the receiving host

The usual result is a crash: another denial of service

Security

To mitigate, we should just ignore ICMP packets that claim to be larger than the MTU: such packets are never generated naturally

Or fix the reassembly code

Modern implementations check sizes are sensible before trying to reassemble fragments

Security

October 2020: Microsoft report a newly-discovered Ping of Death vulnerability in their IPv6 networking code

Actually not a “ping” but an ICMP Router Advertisement, but it is easy to invoke and can crash (blue screen) any unpatched Windows

You might ask how there are still bugs like this in modern operating systems?

Security

Fragment bombs are like SYN floods in effect

Too many fragments for packets that are never completed and so can't be reassembled

This overflows fragment buffer space (where fragments are kept pending reassembly) and likely causes a denial of service, even a crash

Again, implementations need to timeout and drop old fragments

Security

Many other exploits of implementation exist

Usually from the implementers making invalid assumptions about IP and assuming packets are all well-formed and correct

- Jolt (aka sPING): fragmented ICMP packets
- Land attack. The source addresses on TCP SYNs are the same as the destination. The server tries to respond to itself
- Teardrop. Overlapping fragments cause problems on reassembly

Security

- New Teardrop (aka Bonk, Boink, Teardrop2). Overlapping fragments on a UDP packet reassemble to form a packet with an invalid header
- Zero length fragments. In some implementations these were stored but never used. Thus storage was exhausted
- And so on

Making a robust implementation is very hard!

Social Engineering Attacks

These are a pre-computer attack, formerly known as *confidence tricks*

If the machine is too hard to attack, attack the user instead

Often this is much easier than a machine attack

Social Engineering Attacks

It could be as simple as phoning up a systems administrator and persuading them to give you a password to their machine

- Pretend to be a supervisor and threaten to sack them if they do not comply
- Pretend to be a distraught user who has lost their password
- Anything else to unbalance them or get their sympathy

This is much easier than trying to crack a password by brute force

Social Engineering Attacks

Another attack is phishing

This is a form of impersonation to try and convince the user to hand over valuable information, such as credit card numbers

Social Engineering Attacks

A typical phishing attack is:

- the victim receives an email purporting to be from their bank asking them to update their personal details. The email provides a convenient WWW link
- The page looks plausibly like the bank's
- The victim enters their details and sends them off
- The email and Web page are fakes, so now the details are in the hands of criminals

Social Engineering Attacks

Similarly for many other attacks, such as The *419* or *Nigerian* fraud named after the South African police code used to identify this approach

Exercise Read about these

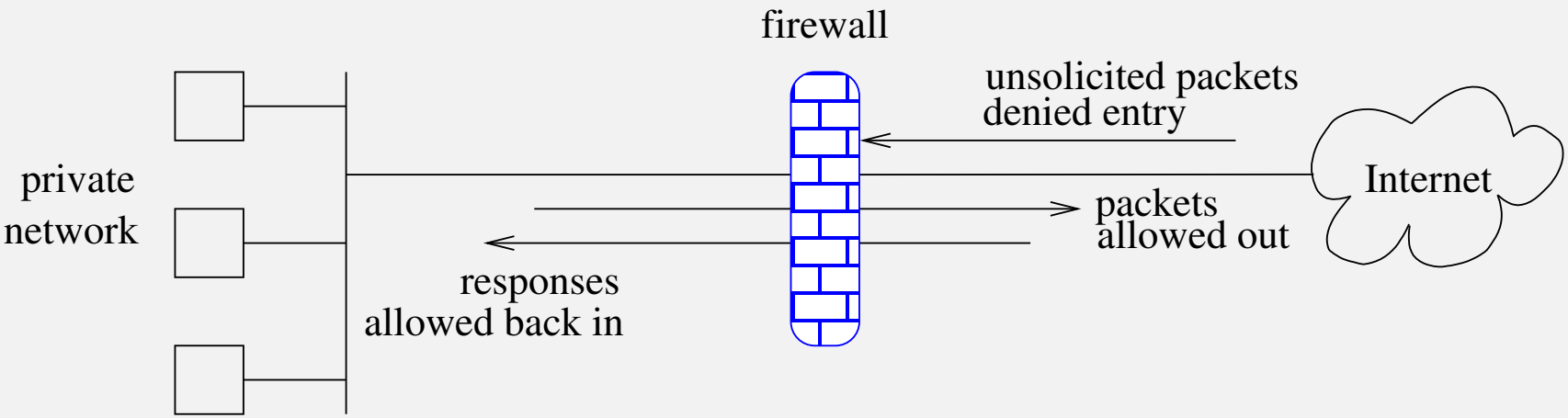
Firewalls

One way to reduce the impact of an attack is to prevent bad packets reaching the host in the first place

A *firewall* is a router/gateway that sits between a private network and the wider Internet and tries to protect the private network from attacks

It might be an ordinary router running firewall software, but specialised firewall hardware also exists

Firewalls



Firewall monitoring packets

The firewall inspects each packet as it enters and decides what to do with it. It might: Pass the packet through unchanged; Pass the packet through, but modified in some way, e.g., with the TOS bits changed or addresses changed with NAT; Drop the packet and send an ICMP back, e.g., “port unreachable”; Silently drop the packet; Or many other possibilities

Firewalls

Dropping silently is a good defence against probes from malicious sources looking for vulnerable services

The normal response to a packet sent to a closed UDP port is ICMP “port unreachable”; while TCP should send a RST

But this has the side effect of telling the sender that this machine is up and running and worth probing further

Silence can make the attacker believe there is no machine at that address at all

Exercise Learn about scanning tools like `nmap`

Firewalls

Firewalling can be applied at any layer

The most common and useful are

- *packet filters* work in the data link, network and transport layers at the individual packet level, making decisions based on protocol (TCP or UDP, etc.), source and destination addresses, port numbers, TOS bits and so on
- *application layer* firewalls work in the application layer and can use information that the applications use, e.g., HTTP filters can make decisions at the Web page level

Firewalls

There are also

- *application proxies* which also work in the application layer and act as an intermediate between the application and the server. They can also make use of application layer information

A Web proxy for an institution might receive all HTTP requests from host within the organisation and choose to relay them onwards, or not, based the details of the HTTP request

Firewalls

Packet filters are fast, efficient and transparent to the application but do not have the discrimination available to application proxies

Application layer filters are slower but more flexible

And proxies require some configuration in the application, e.g., setting up a Web browser to use a proxy

Of course, you can combine things: have a packet filter transparently rewrite packets to the Web to go via a proxy

Firewalls

Firewalls can protect services from attack from outside

E.g., not forwarding inward TCP packets that have destination port 21 will disallow external use of FTP into the private network

This relieves some of the pressure of making all the FTP servers on the private network secure, but does not help against attacks from *inside* the firewall

A safe default installation is to not forward anything inwards to effect maximum protection of the private network

This same protection is also a side-effect of NAT

Of course, NAT works nicely alongside firewalling

Firewalls

There can also be outward or *egress* filtering

E.g., we can force the use of a HTTP proxy by internal hosts by blocking port 80

More subtly, we could use NAT to rewrite connections to port 80 to the server running the proxy

The proxy can then implement an application layer policy, e.g., disallowing access to certain web pages

Another, harsher, way of doing this is for the firewall to drop the packet and return a RST

Public wireless networks often block outward port 25 (SMTP) to prevent users sending spam

Firewalls

However, configuring a firewall is difficult and not to be taken lightly

Capturing the many and varied requirements of a network is subtle and easy to get wrong

Firewalls

Exercise Some attacks get through the firewall by using a phishing attack to get a user to download and run some code. This code can then reach outwards through the firewall. Read about this

Exercise Some firewalls are configured to let in some traffic. For example, allowing an external connection to a security camera, so that you can remotely view your home. But if you can connect, so can others. Read about this

Exercise Some appliances, e.g., security cameras, connect outward to servers so that you can remotely view via the server. But if you can connect, so can others. Read about this

Security and Authentication in IP

The IP was not designed with security in mind

By default, the content of emails and web pages are readable as they travel to their destination

It is easy to write programs that trawl through millions of emails as they pass through a router

As a lot of sensitive and valuable data travels over the Internet these days we need to fix this

We need both security (encryption) and authentication

Security and Authentication in IP

We can apply these at any layer, e.g., in the IP model:

- Application. The application or the user can encrypt the data. For example, you might use PGP to encrypt an email before sending it. Or the application might have in-built encryption
- Transport. SSL/TLS is described shortly. If trusting the user/application is too problematic we can get the transport layer to encrypt for us
- Network. At this layer we have IPSec, also described shortly
- Data link. We can have encryption even in the data link layer. E.g., WPA is used to obscure wireless communications

Security and Authentication in IP

Encryption is just a small part in making a system secure as there are many other factors

Human factors are very important: we mustn't forget social engineering attacks

Also, there is no point in having a military-grade encryption system if you have an easily-guessable password

Security and Authentication in IP

There are actually two problems to address:

- Secrecy
- Authentication

Secrecy is familiar, but authentication is more fundamental

Security and Authentication in IP

Secrecy is

> *make sure that this data is not readable by anyone other than the recipient*

Authentication is

> *make sure the recipient is who I think they are*

Security and Authentication in IP

There is no point sending a strongly encrypted message only to find you sent it to the wrong person

Perhaps you are buying over the Web and you want to send your credit card details to Acme Widget Company

So you negotiate a military-grade encryption key with them and send the details, happy in the knowledge that no-one else can read them

Later you discover the Web page was a fake and your details are now in the hands of criminals

You must have some way of determining if someone is *who they say they are*: this is authentication

Security and Authentication in IP

In the real world we use documents like passports and driving licences to identify people

In the Internet world we do the same, except now the documents are chunks of mathematical data

For details go to a crypto course!

Aside

If we were doing things properly, we should also talk about *authorisation* at this point

After authentication there is the question of whether this entity is allowed access to some resource

For example, in WPA-PSK, a correct password is usually taken as authorisation; in WPA-Enterprise the server will have a list of allowed users+passwords

Security and Authentication in IP

We have already considered a link layer security: WPA

We now look at a few others, including IPSec (network layer), PPTP and L2TP (link layer), SSL/TLS (transport layer)

In the normal way of blurring layers when thinking about functionality, PPTP and L2TP are regarded as link layer, even though they layer over IP

Security and Authentication in IP

IPSec

IPSec can be used to set up secure point-to-point links (see VPNs, later), but can also be used to secure and authenticate individual connections when the other end supports it: *opportunistic encryption*

IPSec consists of several protocols and defines several IP optional header fields

Security and Authentication in IP

IPSec

Secrecy is implemented by *Encapsulating Security Payload* (ESP)

Authentication by *Authentication Header* (AH)

Keys are managed by *Internet Key Exchange* (IKE) which itself uses the *Internet Security Association and Key Management Protocol* (ISAKMP)

Security and Authentication in IP

IPSec

IPSec authenticates connections, not users

You do not use it to login, but to ensure the remote host really is Acme Widgets before you send data (money) to them

Both ESP and AH require a secret shared key to work

Security and Authentication in IP

IPSec

This key can be

- pre-agreed (*manual keying*)
- negotiated by IKE

IKE can itself use a pre-agreed key to deliver the ESP/AH key, or use a public-key certificate mechanism

Security and Authentication in IP

IPSec

Normally there is one IKE process per host and it manages all exchanges for that host

When a new IPSec/IP connection is started an IKE exchange will take place before the IPSec can continue

This may take some time: even enough to cause a TCP timeout on slow machines

Security and Authentication in IP

IPSec

IPSec

- is directly inside the IP layer (optional headers), so UDP and TCP are easily layered transparently on top
- clearly only applies to IP
- AH does authentication, while ESP does secrecy and authentication. Pure authentication is OK if you do not need secrecy, but pure secrecy is open to impersonation attacks without some authentication

Security and Authentication in IP

IPSec

- ESP has a trailer as well as a header: this can contain padding to hide the length of the original packet
- ESP only authenticates the payload, while AH authenticates all the packet, excepting the mutable fields (TTL etc.) that change en route
- applies to both IPv4 and IPv6

Security and Authentication in IP

IPSec

Problems:

- initial connection overhead is high
- IPSec is tricky to set up and manage
- It works at the OS host level, and so needs a competent administrator
- and also does not have the flexibility that (say) SSL/TLS has, allowing each application to be managed independently

Security and Authentication in IP

IPSec

The general view of IPSec is that it is slow: though this view was formed some years ago when computers were much slower than now

In fact, after the connection setup, IPSec is not that bad in terms of speed

But the configuration question remains

Exercise The University uses IPsec: investigate

Security and Authentication in IP

IPSec

A new alternative to IPSec that is growing in popularity is *WireGuard*

It is (primarily) a point-to-point connection that is high performance, using modern design and algorithms in a small, auditable, codebase (about 1% the size of the IPSec code)

It layers over UDP to provide an encrypted, authenticated network layer

It is very easy to set up

Exercise Read about WireGuard

Security and Authentication in IP

VPNs

Some systems are based around creating a *Virtual Private Networks* (VPN)

A VPN allows a machine to appear to be on another network by means of tunnelling

Recall tunnelling: where one protocol is layered over another so the lower protocol can transport the upper protocol transparently over a network that might not normally carry the upper protocol

VPNs are *private* as they add encryption of the data in the tunnel to provide security

Security and Authentication in IP

VPNs

Traffic from the host travels through the tunnel to the network, where it can be routed as if it had originated there

This allows the host to use the services on the network as if it were local to that network

This is good for *teleworkers*

For example, IPSec and WireGuard are VPNs

Security and Authentication in IP

VPNs

In overview a VPN:

- is software that creates a new virtual network interface
- when packets need to use the VPN they are routed out on that interface
- the packets are encrypted (in the kernel or in user level software)
- this data is now sent out over a real interface, e.g., using UDP

Security and Authentication in IP

VPNs

At the receiving end:

- data arrives in (UDP probably) packets on a real interface
- it is decrypted and presented as packets on the virtual VPN interface
- which are read by the server application

As far as the client and server are concerned, they are operating as normal, sending and receiving packing on a normal interface

Security and Authentication in IP

VPNs

There are two common setups for VPNs that treat data from the client in different ways

- where *all* client traffic goes out through the VPN, and the server end of the VPN routes it onwards as appropriate
- where only traffic destined for the server's network goes through the VPN. Other traffic from the source goes directly to its destination in the normal way. This is sometimes called a *split tunnel*

Security and Authentication in IP

VPNs

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
...
default         home.gateway.ho 0.0.0.0         UG    0      0        0 wlan0
cs.bath.ac.uk   172.16.0.1      255.255.255.240 UG    2      0        0 tun0
fire.cs.bath.ac home.gateway.ho 255.255.255.255 UGH   0      0        0 wlan0
...
```

A packet destined for the CS network goes through (virtual) interface `tun0`, which actually sends packets to the VPN software on the local machine; This is encrypted and encapsulated in a packet with destination `fire` on the CS network; And port number that of the VPN software on the remote server; This packet then goes through the host (`H`) route to CS through the real interface `wlan0`; The routes are checked longest mask first, to prevent an infinite loop! The VPN software on the remote server gets the packet and decapsulates it; It

Security and Authentication in IP

VPNs

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
...
default         172.16.0.1      0.0.0.0         UG    0      0        0 tun0
cs.bath.ac.uk   172.16.0.1      255.255.255.240 UG    2      0        0 tun0
fire.cs.bath.ac home.gateway.ho 255.255.255.255 UGH   0      0        0 wlan0
...
```

In comparison, the other setup; Here all default traffic goes through interface `tun0`; This is encapsulated in the same way; And routed through the tunnel via the real interface; The VPN software on the remote network gets the packet, deencapsulates it, rewrites its source address and forwards it to the destination which now might be anywhere, not just on its local network; Returning packets are sent back through the VPN as before

Security and Authentication in IP

VPNs

In the first setup you get secure access to the work network, but to the rest of the world you are at home (and not secure)

The second makes you to appear to everyone to be at work as all your packets have a work IP address on them

Security and Authentication in IP

VPNs

Exercise Find out how to use the University's VPN from your home

Exercise The `www.newscientist.com` website reserves some content for subscribers. It uses the requesting IP address to check for access rights. The University is a subscriber. Use a VPN to get at this content from your home

Exercise Find out how (or if) your favourite VPN can be configured as a split tunnel

Security and Authentication in IP

VPNs

The *Point-to-Point Tunneling Protocol* (PPTP) was devised by Microsoft to support VPNs

It

- tunnels IP over PPP over the *Generic Routing Encapsulation* protocol (GRE) over IP and sends connection control messages over a separate TCP connection
- layers only over IP

Security and Authentication in IP

VPNs

It

- can encapsulate other protocols such as IPX (Internetwork Packet Exchange, NovelI) and NetBIOS/NetBEUI (Network BIOS, NetBIOS Extended User Interface, Microsoft)
- uses PPP for authentication
- can use *Microsoft Point-to-Point Encryption* (MPPE) for privacy, combined with *Challenge-Handshake Authentication Protocol* (MS-CHAP) for authentication
- is simple to set up

Security and Authentication in IP

VPNs

On the other hand, PPTP is regarded as insecure as the authentication mechanism (MS-CHAP) can be broken

A later version (MS-CHAPv2) fixes some, but not all of the holes

Security and Authentication in IP

VPNs

The *Layer 2 Tunneling Protocol* (L2TP) combines features of PPTP and *Layer 2 Forwarding* (L2F) developed by Cisco Systems Inc

Security and Authentication in IP

VPNs

It

- tunnels IP over PPP over L2TP over UDP
- is intended to be used over ATM, Frame Relay and X.25 networks
- has no native encryption and must rely on, say, IPSec for secrecy
- mainly uses PPP for authentication, but can also use ESP from IPSec
- is believed to be more secure than PPTP
- is simple to set up

Security and Authentication in IP

VPNs

While L2TP and PPTP were popularised by Microsoft, other operating systems prefer other solutions

We shall talk about *OpenVPN*, later

Security and Authentication in IP

Tunnelling TCP

Note that all these tunnels layer over UDP, not TCP

This is because tunnelling TCP over TCP is usually a bad idea

TCP has a large overhead to gain reliability: there's no point in paying this cost twice

Plus, each TCP has its own idea of timeouts and retransmits and they can start to fight each other: the retransmit of one TCP will be viewed as a duplicate by the other TCP

Thus most VPNs tunnel over UDP

Security and Authentication in IP

Tunnelling TCP

Exercise Read about the *Secure Socket Tunneling Protocol* (SSTP) and the *TCP meltdown problem*

Security and Authentication in IP

VPNs

Generic problems of VPNs include

- there is encryption and authentication header overhead in every packet: this may cause extra packets or extra fragmentation
- there is overhead in the time taken to encrypt or authenticate the packets

Security and Authentication in IP

VPNs

- some routers or ISPs make decisions based on the type of traffic (e.g., video or HTTP): encryption hides this and makes efficient routing harder
- some ISPs like to charge more for, or manage certain kinds of traffic (e.g., bittorrent and video) and this hides the kind of traffic. So some ISPs have blanket bans on VPNs
- in VPNs speed is secondary to security, but people will not use them if they are too slow

Security and Authentication in IP

And, as previously mentioned, any kind of security is viewed with suspicion by law enforcement agencies

Note that a VPN can make you appear to be in a different country

Good for evading country-locked content (geofencing), but bad for law enforcement and people who want to track what you are doing

Security and Authentication in IP

For example, the *Investigatory Powers Act 2016*, and its update, *The Data Retention and Acquisition Regulations 2018*, a law that can require your ISP to log every website you visit and every recipient of emails and phone calls (your *Internet Connection Records*)

Security and Authentication in IP

Some core information is accessible, without warrant, by certain people:

- account reference
- a source and port address, a destination IP and port address
- a time/date + duration
- partial URLs (only part containing server name, no content)

So, the metadata and not the data

An interception warrant is needed for more, e.g., content

Security and Authentication in IP

The law also gives the security services new powers to hack computers and, e.g., pressure service providers not to support end-to-end encryption

This is a very contentious law, not only because it may be in contravention of EU privacy laws, and that might mean the UK cannot legally process data from the EU

Exercise Read the list of 50 or so authorities that can access your web history, without warrant

Security and Authentication in IP

VPNs

Many other VPN implementations exist

- *Crypto IP Encapsulation* (CIPE). A lightweight point-to-point protocol that layers over UDP
- `ssh`. This remote login protocol also has a VPN mode, but it layers over TCP
- OpenVPN (discussed later) tunnels over the transport layer SSL/TLS

Security and Authentication in IP

VPNs

The cryptographic quality of these varies widely: CIPE is generally judged to be not much better than PPTP

In real life, PPTP and OpenVPN are common; Wireguard is growing in popularity; the others are rarely seen

Exercise Read about the vulnerabilities in PPTP

Security and Authentication in IP

Transport Layer

Transport Layer security is used much more than Network Layer security

The *Secure Socket Layer* (SSL) and its update *Transport Layer Security* (TLS) implement a security layer over the transport layer (usually TCP)

And you use this new layer instead of TCP

Note that SSL is no longer recommended as it has flaws in the protocols

You should only use TLS, preferably versions 1.2 or later

Security and Authentication in IP

TLS

This security layer is above TCP, so it must be in the application layer

It provides security of the data and authentication of the remote host

After a TCP connection has been made a TLS handshake in the application authenticates the connection and negotiates a secret key

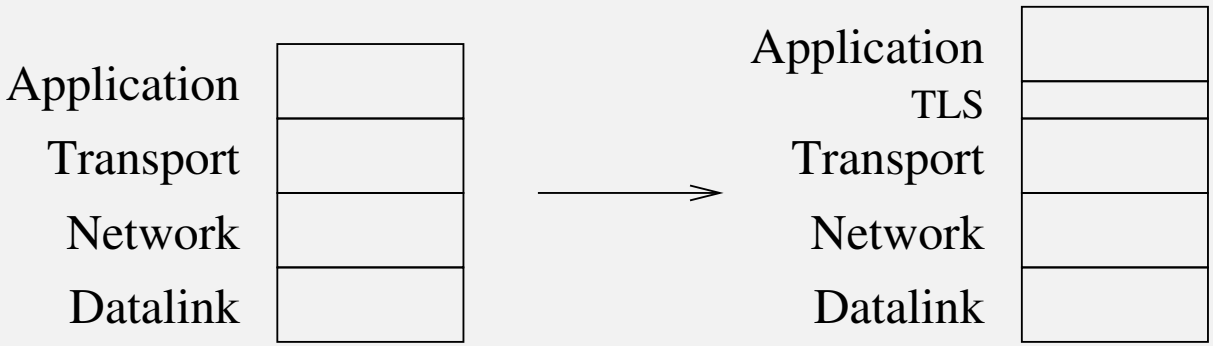
The key is then used to encrypt subsequent data sent over the TCP connection

Security and Authentication in IP

TLS

TLS provides a new transport layer that can be used very much like TCP (reliable, connection oriented, etc.)

Sometimes called a *shim* layer as it sits between two other layers



TLS Shim

Security and Authentication in IP

TLS

The client can authenticate the server through the use of public-key certificates

During the handshake the client application receives a certificate from the server that it can authenticate in a variety of ways

For example, Web browsers often contain a selection of master certificates from *certification authorities* that it can use to check the certificate from the server

Exercise Examine your browser to see which certification authorities it uses

Security and Authentication in IP

TLS

Similarly, if it wishes, the server can request a certificate from the client to authenticate the client a similar way

It would be possible to use this instead of the usual “login and password” mechanism that servers often use to authenticate the client

Unfortunately, the requirements of administration of the certificates is much beyond the skill of the average user

Which is why login and password is still widely used to authenticate clients to the server

Security and Authentication in IP

TLS

Transport layer security is very flexible, but requires the application programmer to understand and use the function calls to set up certificate checking and the handshake

That is, the programmer must invoke this layer in their application: and correct use of TLS is not trivial

Their program can then read and write via the secure connection they get from this instead of reading and writing directly from the TCP socket

Security and Authentication in IP

```
s = socket(PF_INET, SOCK_STREAM, 0); // TCP socket
...
// Initiate TCP connection to server
connect(s, (struct sockaddr *)&addr, ... );
...
read(s, buf, 1024); // read data
...
```

Security and Authentication in IP

```
s = socket(PF_INET, SOCK_STREAM, 0); // TCP socket
...
connect(s, (struct sockaddr *)&caddr, ... );
...
ssl = SSL_new(ctx); // context contains info about ciphers
SSL_set_fd(ssl, s); // associate socket with ssl struct
...
SSL_connect(ssl); // do the SSL handshake
...
if SSL_get_verify_result(ssl) != X509_V_OK { // authenticate
... bad certificate ...
}
...
SSL_read(ssl, buf, 1024); // read data
...
```

Security and Authentication in IP

TLS

Many protocols can layer over TLS (instead of TCP) to give a secure version:

- HTTPS is HTTP (the protocol to fetch Web pages) layered over TLS
- SMTPS for SMTP (the protocol used to send email)
- IMAPS for IMAP (the protocol used to read email)
- Etc.

This is a relatively easy way of making secure protocols from insecure ones: just find the parts of code that read and write from IP sockets and change them to use TLS

Security and Authentication in IP

TLS

A few people regard TLS as a presentation layer between the application and the transport layer

An interesting point of view, as TLS does rearrange your data

But not a strong point of view, as TLS does not solve the other problems a presentation layer is supposed to address, e.g., character sets

Most people regard TLS as a transport layer

Security and Authentication in IP

TLS

Exercise Read about STARTTLS, a protocol to negotiate a TLS connection, as used by SMTP and IMAP

Exercise Contrast HTTPS with SHTTP, which is an extension of HTTP to include security

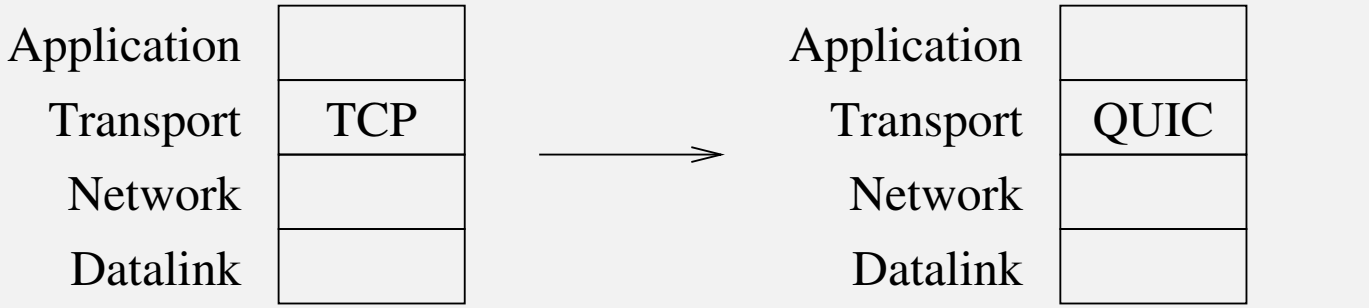
Exercise Read about HTTP/2, the latest version of HTTP, that encourages the use of TLS

Security and Authentication in IP

TLS

We should also mention QUIC (again) at this point

QUIC is a transport layer that replaces TCP + TLS



QUIC

Security and Authentication in IP

TLS

The opening handshake of QUIC does both the reliability setup *and* the security setup

Thus making QUIC faster to set up

In the future, QUIC will the transport layer of the Web (HTTP/3), and possibly other applications, too (e.g., DNS)

Security and Authentication in IP

TLS

HTTPS hides the requested URL and the content of a Web page returned: this is in the encrypted data of the HTTP request; but it cannot hide the IP address of the server

So an eavesdropper cannot tell if you are reading `www.example.com/good.html` or `www.example.com/bad.html`

They *can* tell you are looking at something on the host with the IP address of `www.example.com`

Traffic analysis of communications is a powerful tool that has been used for decades

Security and Authentication in IP

TLS

Some Websites (e.g., Tumblr) have multiple sub-sites hosted on the same IP address: called *virtual hosting*

For example, `good.tumblr.com/home.html` and `bad.tumblr.com/home.html` with both `good.tumblr.com` and `bad.tumblr.com` having the same IP address

The server name is included in the HTTP request and the server uses this to determine which sub-site the client wants

Security and Authentication in IP

TLS

```
GET /home.html HTTP/1.1
Host: bad.tumblr.com
User-Agent: curl/7.60.0
Accept: */*
```

HTTP request for `home.html` on (virtual) server `bad.tumblr.com`

HTTPS runs over TLS so this is hidden from an eavesdropper

Security and Authentication in IP

TLS

But the TLS handshake (before the HTTP request) requires an authentication certificate from the server that is based on the server name

Server Name Indication (SNI; RFC6066) is part of the HTTPS handshake that asks for a cerfificate for the server name (e.g., `bad.tumblr.com`) *in the clear*

As we don't yet have a shared secret key, this can't be encrypted

Security and Authentication in IP

TLS

So accesses to such sub-sites are trackable:

- in the DNS lookup of the sub-site name
- in the HTTPS SNI handshake that contains the name of the sub-site

Although the *content* of the Web pages is always hidden, which sites are being accessed can be tracked

Security and Authentication in IP

TLS

People are working on filling these gaps

We have already mentioned DNS over HTTPS (DoH; RFC8484) that hides the DNS lookup

Exercise Read about Encrypted SNI (eSNI) that hides the handshake

Exercise Read about Oblivious DNS over HTTPS (ODoH) that hides the DNS request from the DNS server(!)

Exercise Why are sites like Reddit that also have many sub-sites not affected by this?

Security and Authentication in IP

Aside

Think about the ways your Internet use can be tracked (by ISPs or others, for the Investigatory Powers Act; or just general snooping by bad actors) or manipulated (by bad actors, including some ISPs)

These include:

- Reading/manipulating your Web traffic or emails (unless you use HTTPS or an appropriate secure transport)
- Reading/manipulating your DNS requests (unless you use DoH or similar)
- Reading/manipulating your Server Name Indication traffic on TLS authentication certificates (unless you use eSNI)

Security and Authentication in IP

TLS

There are overheads in using TLS

- A one-off overhead of (re)writing the application code to use TLS
- A per-connection overhead of TLS setup messages and the associated computation for checking certificates
- A per-packet overhead of data expansion in the encryption (this effectively reduces the MTU)
- A per-packet overhead in the computation required to encrypt or decrypt the data

These costs are not huge, but you must make the choice of whether they are worthwhile

Security and Authentication in IP

TLS

Big providers have mostly moved their services to TLS by default, usually HTTPS

For example, Google now uses it to protect all of Gmail and Web searches

For such a large enterprise there is a significant cost in doing so, but the security gained makes it worth doing

And customers are starting to be more security conscious and are now demanding it be done

Exercise Compare using transport layer security against network layer security

Security and Authentication in IP

TLS

The usefulness of TLS does not stop there

OpenVPN uses TLS as a *datalink* layer

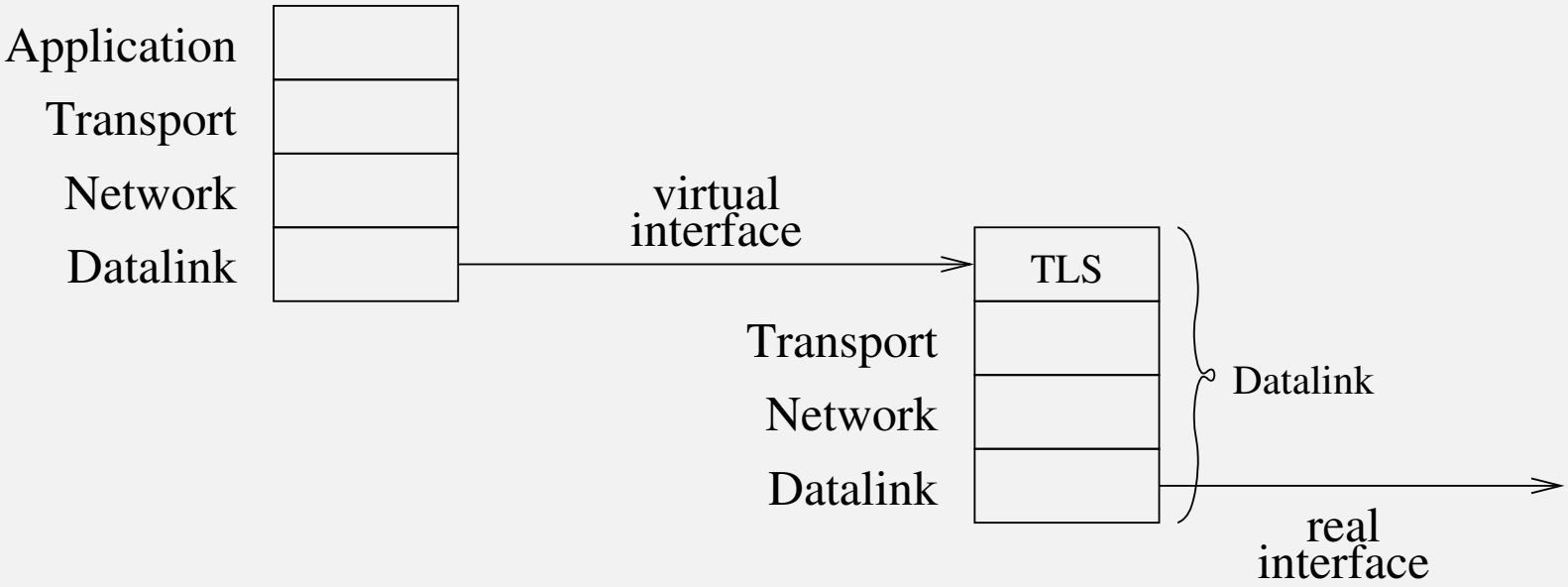
That is, it layers IP over TLS to build its private network

It creates a virtual network interface that the OS can pass IP packets to

The OpenVPN code then encrypts, authenticates and does whatever it needs (using TLS) before handing the result on to a “real” transport layer, usually UDP (as this is a VPN)

Security and Authentication in IP

TLS



TLS

Security and Authentication in IP

TLS

The encapsulated data then goes down through the normal transport and network layers and is transmitted over the real physical layer

At the receiving end, the real transport layer hands the data to OpenVPN which decrypts and passes the resulting IP packets to the OS to pass up the rest of the stack

Of course, it is layering that allows all this to work!

There is a cost of about 10% overhead in practice

Exercise Compare these costs with using a Network Layer approach to VPNs, such as IPSec or WireGuard

Security and Authentication in IP

TLS

And now we have the usual benefits of a VPN: user applications can be unsecured (but remember your data is only secure while inside the VPN, not if the final destination is somewhere in the wider Internet)

And the usual costs (For the geeks: the TLS code runs in user mode, so the data in each packet has to go between user mode and kernel mode several times)

Exercise Compare using an insecure login over a secure network against a secure login over an insecure network

Exercise And what about using a secure login on a secure network?

Security and Authentication in IP

TLS

A web browser looking at a page secured by HTTPS on a VPN on a home network might be layering

> Web page in HTML/CSS over HTTP over TLS over TCP over IP over TLS over UDP over IP over PPP over Ethernet over Cat6a

Networks

Ethernet

What are the physical encodings of bits on a 10Mb/s Ethernet?

A simple way would be 0V for 0 and 1V for 1, running at 10MHz

But this has a number of problems

Networks

Ethernet

1. An empty network and a stream of 0s looks the same

And so you could not do carrier sense

2. Bits need to be synchronised to prevent drifting out of step (was that 1000 or 999 0s?)

3. A long stream of 1s is a steady 1V: this is electrically a bad design, an average 0V is best

To connect devices easily you need an AC signal, not a DC one

Networks

Ethernet

So 10Mb/s Ethernet uses a *Manchester Encoding*

- Split the time interval for a bit into two parts
- Low then high voltage is a 0
- High then low voltage is a 1

So the average is 0V

-0.85V for low, +0.85V for high

This voltage is a compromise: a bigger voltage gives a more robust signal that will travel further, but it uses more power

Networks

Ethernet

Easy to synchronise: the transit through 0V is the middle of a bit

This does double the frequency of the signal to 20Mhz

We can use Cat 4 (or better) cable for this

Manchester encoding solves the above problems neatly and actually simplifies the hardware needed

It is described as *self clocking*, as the reading end does not need a clock to determine where the bits are

Networks

Ethernet

What of 100Mb/s Ethernet?

We can't use even Cat 5e cables with Manchester as it is only specified to 100MHz, and we would need 200MHz

Networks

Ethernet

Instead we start by encoding 4 data bits as 5 physical bits in a *4B/5B* encoding; e.g., 0000 become 11110

Input	4B/5B	Input	4B/5B
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

With some control patterns, e.g., IDLE 11111.

Networks

Ethernet

Hasn't 4B/5B made things worse: 5 bits where there were 4?

But now we use a *three* level physical encoding *MLT-3*

This has +, 0, and - levels ($\pm$0.85V), again using transitions to encode bits

Networks

Ethernet

Transitions are cyclical

- to 0
0 to +
+ to 0
0 to -

A transition marks a 1, no transition marks a 0

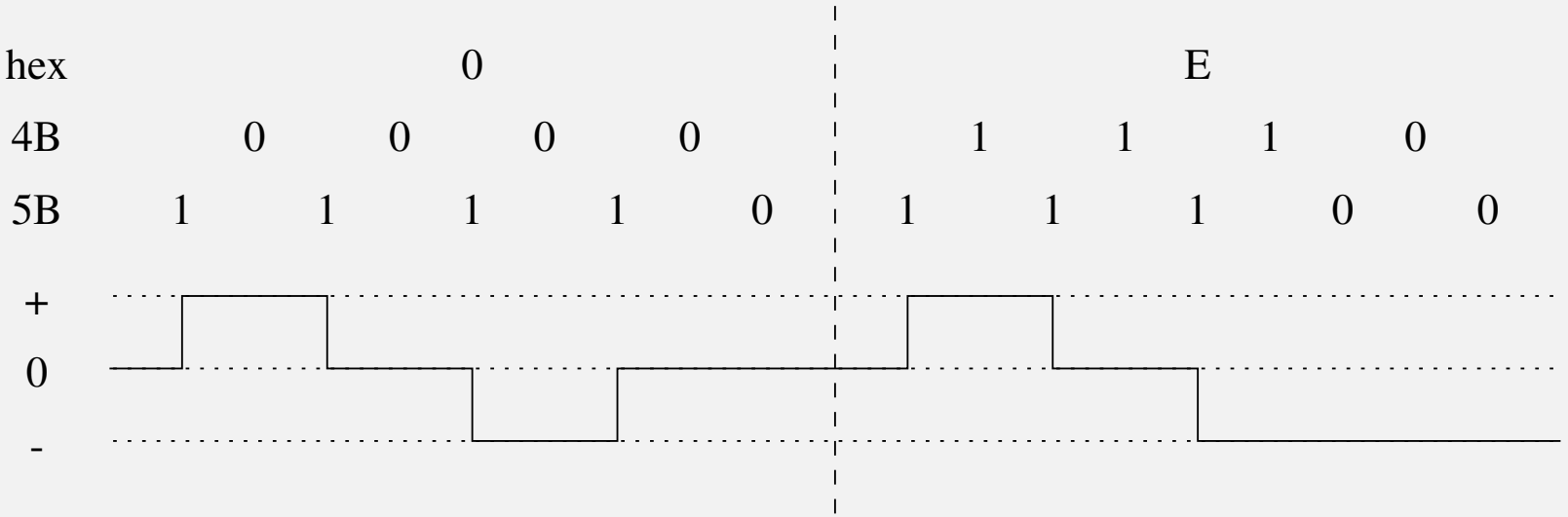
The 4B/5B translation ensures that every chunk of 5 symbols has at least two transitions, so average voltage is roughly 0

E.g., input 0000, with no transitions becomes 11110 with four transitions

Networks

Ethernet

An example. Hex value 0E = 0000 1110



MLT encoding

Networks

Ethernet

Some words:

A physical representation is called a *symbol*

Symbols need not be binary

And need not represent a whole number of bits

The *baud rate* is the number of symbols per second

Networks

Ethernet

100Mb/s Ethernet runs at up to 31.25MHz for a symbol rate of 125MBaud: all 1s output (IDLE) is four transitions (- to 0, 0 to +, + to 0, 0 to -) per cycle
(4 symbols/cycle $\times$ 31.25MHz $=$ 125MBaud)

This has a symbol rate of 125MBaud for a data rate of 100Mb/s: 80% efficient or 1 physical symbol is $4/5 = 0.8$ bits

Networks

Ethernet

For Gigabit Ethernet 1000Base-T: 8 bits become 4×3 physical bits in a continuously changing encoding (not a table lookup)

Each 3 bit chunk is encoded using transitions between 5 levels (PAM-5)

Over all four pairs in the cable simultaneously, in both directions on all pairs

10Gb Ethernet uses a PAM-16 over a very complicated coding (*Tomlinson-Harashima Precoding*)

(SATA and USB 3.0 use 8B/10B; USB 3.1 uses 128B/132B; etc.)

Networks

Ethernet

And then there is Ethernet over optical fibre. . .

Exercise Read about the physical encodings that are used in fibre

Bridging

Some time ago we talked about bridges joining networks

ARP bridging is fine for joining a pair of small networks, but less so for larger collections of networks

The IEEE 802.1d Ethernet Bridging standard addresses this, dealing with the cases of multiple routes between hosts

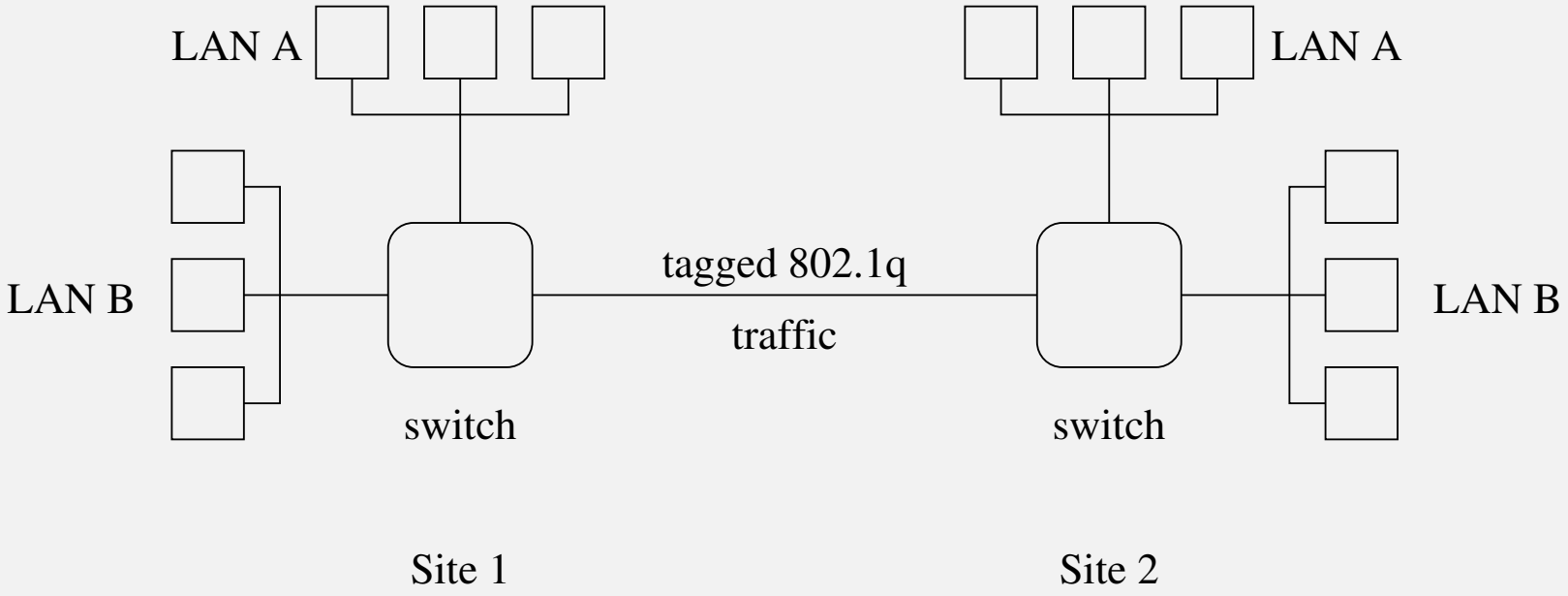
Virtual Bridging

And a common variety is 802.1q *virtual bridging*

More commonly called *Virtual LANs* (VLANs)

This is a kind of reverse of the ARP bridge: it allows more than one network to run on a *single* physical network

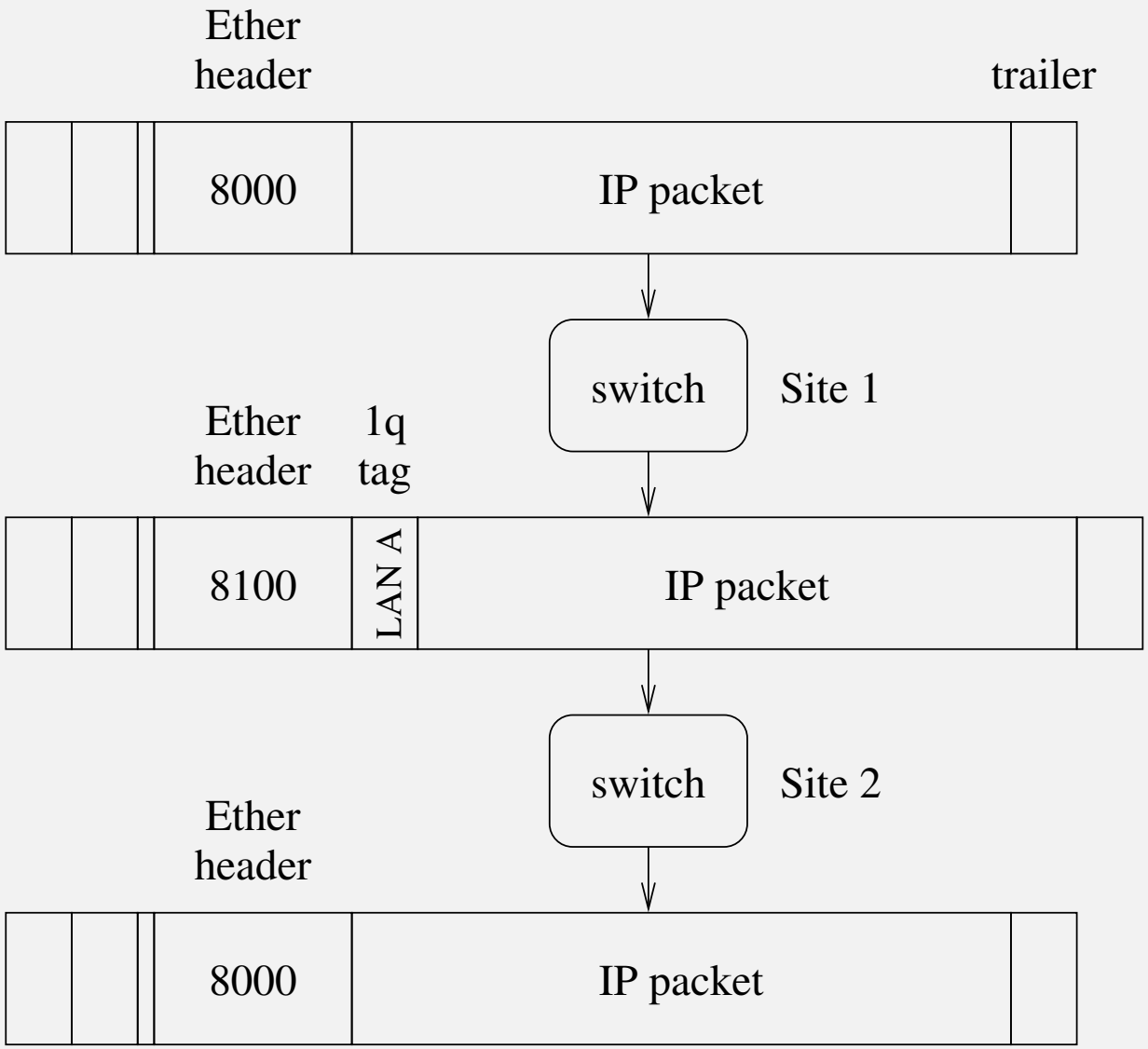
Virtual Bridging



Virtual bridge

A company has two separate sites 1 and 2 with a single dedicated link between them; They want to run two separate LANs, A and B, but not to buy a second link between the sites; They can use 802.1q *tagging*; A packet from LAN A in Site 1, say, arrives at the switch; The switch knows to route the packet over the remote link; it places a 802.1q *tag* on the frame; A tag

Virtual Bridging



Tagging packets in a VLAN

Virtual Bridging

This generalises well to many virtual LANs and allows many networks to share infrastructure, thus saving on cost

Note: this is quite different from *Virtual Private Networks* (VPNs), which we shall talk about later

Exercise Look up the structure of a VLAN tag

Exercise The University uses VLANs extensively. Find out about this

Exercise How does tagging interact with maximum frame sizes, e.g., in Ethernet?

The End

We started with layering models that suggested what a networking standard needs to consider, and where

How does reality match with these suggestions?

The End

Action	ISO	Reality
Error detection	datalink	datalink (Wi-Fi ACK); network (IP checksum) transport (UDP/TCP checksum); application possibly
Error recovery	datalink	datalink (Wi-Fi ACK); transport (TCP only); application possibly
Routing	network	network (IP)
Congestion	network	network (possibly IP routing); transport (TCP)
Flow control	network	transport (TCP)
Accounting	network	?
Quality of Service	network	network (IP Diffserv)

The End

Action	ISO	Reality
De/packetisation	transport	transport (UDP/TCP)
Reliability	transport	transport (TCP only); application, if needed
Session	session	network (IPv6 flow label); transport (TCP sequence numbers); application, if needed
Presentation	presentation	application
Application	application	application

The End

Exercise Read about “Layer 8”

The End

End of Lectures

Future sessions will be problems classes, and going through past papers