Networking 2: Models, Methods and Metal

I.G.Batten@bham.ac.uk
https://www.batten.eu.org/~igb

Check Panopto!

- Is it running?
- Is it running?
- Seriously, is it running?

Contents

What do we want to do?

- We want to link network elements together so they can exchange data:
    - In infinite amounts
    - Infinitely quickly
    - With zero error rates
- With that, there is no need to make engineering trade-offs, and we can all go home

Reality Called

- Just as hi-fi nerds talk about “straight wires with gain”, but then accept they can’t have it, we have to accept that data travels at finite speeds through networks of finite capacity, and errors happen.
- We can characterise a flow of data in terms of:
  - volume per second (bits per second)
  - latency (seconds)
  - error rate (errors per bit)
- With suitably large error bars on these numbers

Volume

- Bandwidth, capacity, etc
- How many bits per second can I put in one end and have emerge from the other

Latency

- How long does it take for a particular bit to travel through the network (which may be defined to include parts of the end stations, so we might be more interested in application-to-application latency).
    - The speed of light imposes a lower bound, a lower bound high enough we will have to worry about it.

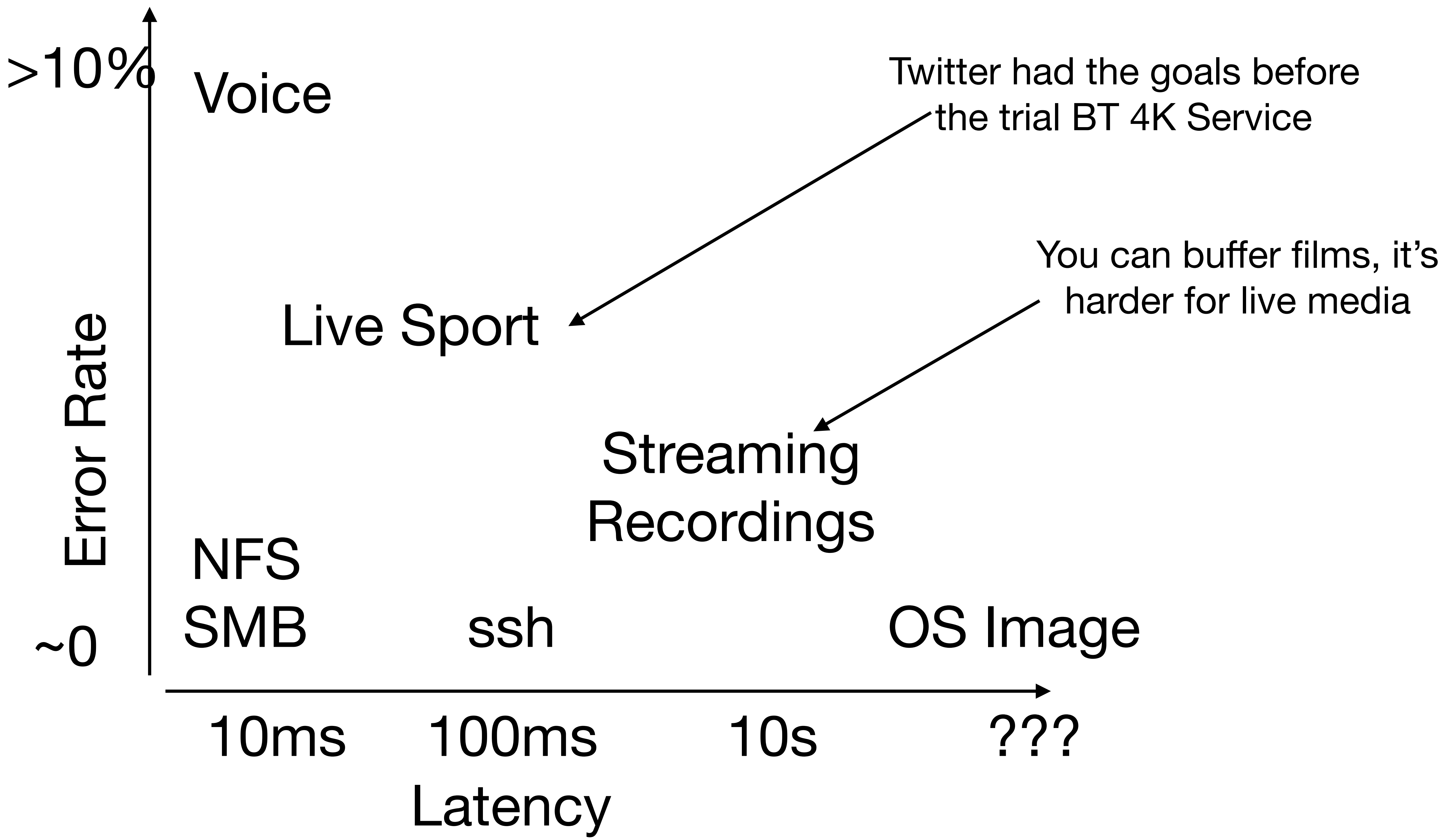
Error Rate

- Will be a property of the medium, technology and potential interference
  - Data loss is an error
- Cosmic Ray, mains impulse noise, cable and connector issues, etc, etc.
- Determines how much error checking and correction we can justify, and ultimately there will be an uncorrected error rate we have to accept.

Different Data: Different Requirements

- Voice can accept low bandwidth and high errors, but latency is a problem
- File Transfer of operating system images is about reliability above everything else
- You can look at similar trade-offs for different types of data

Trade Offs


>10%
~0
Error Rate
Voice
Live Sport
Streaming Recordings
NFS SMB
ssh
OS Image
10ms
100ms
10s
???
Latency
Twitter had the goals before the trial BT 4K Service
You can buffer films, it's harder for live media

Over time

- Link bandwidth tends to infinity (terabits per second is trivial with DWDM), but is still limited by cost
- Latency is affected both by processing times, which decrease, but also the speed of light which is always there

So what is the effect of the speed of light on a link between London and Birmingham? London and California? $c = 3 \times 10^8\ \mathrm{ms}^{-1}$.

- Raw error rates remain about the same, but we have more processing power and spare bandwidth so can do better error correction
- So let's look at some hardware.

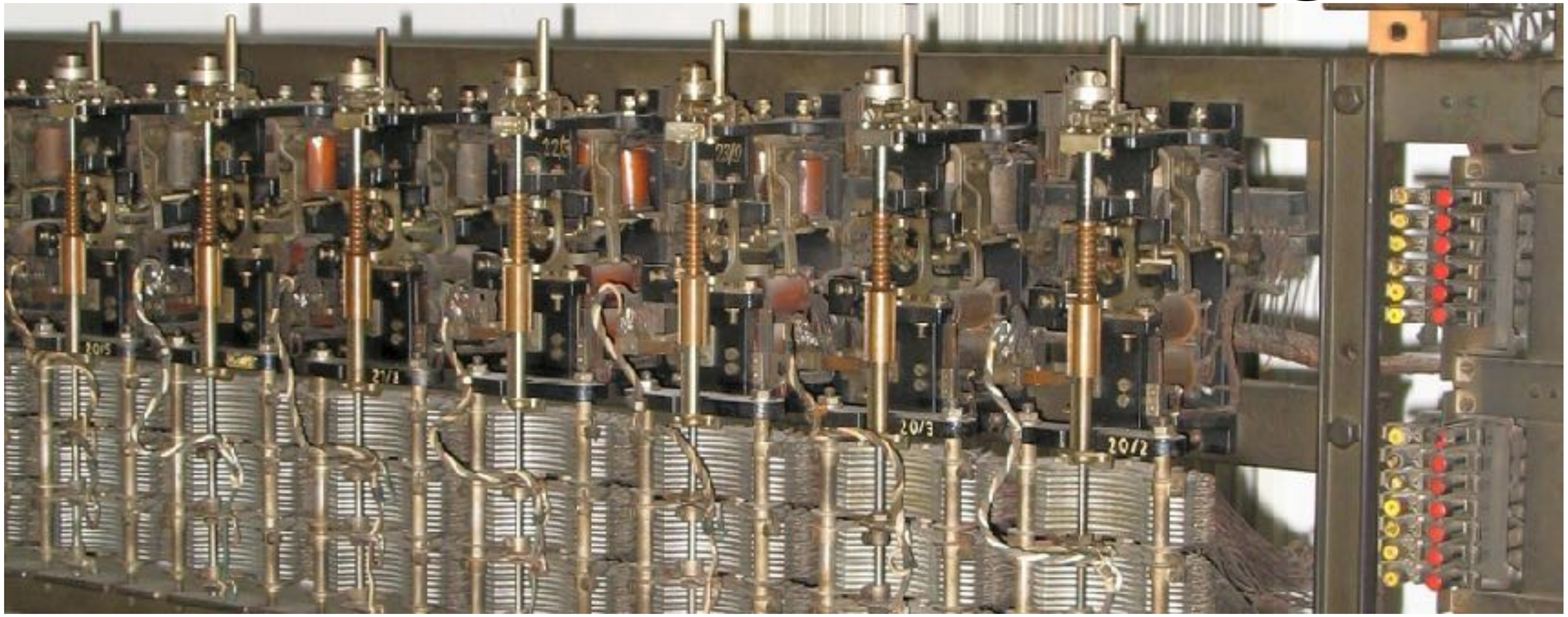
“Never underestimate the bandwidth of a jumbo full of tape”

- Standard 2.5” SSD laptop drive is 10 x 7 x 0.9 cm, so volume is $63cm^3$, weighs 130g.
- $1m^3$ will therefore hold 15800 such drives, total weight 1.6t
- Revenue payload of a 747 is ~300t, ~$500m^3$, so we are weight limited to $180m^3$ of disk drives. 180 x 15800 = 2.8 x 10^6. Recall, 1TB is 10^{12} bytes.
- 2.8 million x 1TB = 2800PB, 2.8 x 10^{18} bytes. Over an 8 hour flight, 2.8 x 10^5 seconds, that’s ~10^{13} bytes per second, ~10^{14} bits per second. ~100Tbits per second is pretty fast: it’s the whole UK/US bandwidth, and then some. 1t, 1 hour = ~2.5Tbps, which makes an estate car to London pretty handy for backups.
- Latency not great for playing Call of Young People’s games.
- Check my maths…all those zeros.
- And of course, most of the weight is the casing. If we did this with M2 PCIe drives, the numbers would be larger.

But…

- Waiting for the plane to come isn’t great for anything
- You could have a conversation by exchanging tapes of voice, or tapes of data, but it isn’t going to be quick.
- So doing it electrically is useful
- Let’s skip over a manual plugboard exchange, and move straight to…

Circuit Switching



- Old telephones: exchanges linked physical wires together so that the microphone at one end was continuously connected to the speaker at the other end, via suitable amps and multiplexors

Problems

- Very inefficient: ties up a duplex circuit even when one or both parties are quiescent
- Multiplexing is very complicated and expensive using 1950s technology
  - You have to treat the wire as a radio and use multiple carriers, which is hard to do reliably

Packet Switching

- Proposed independently by Paul Baran in the USA (RAND) and Donald Davies in the UK (NPL).
    - Davies had worked as an assistant to Klaus Fuchs at Birmingham on Tube Alloys
    - Baran was working on survivable architectures for post-nuclear communications

Packet Switching

- Divide the data stream up into small units, called "packets".
- Give each packet some identifying information
- Switch the packets over your network until they get to the destination
    - "Switch" is American railroad usage for what are "points" or "turnouts" in UK railway usage.

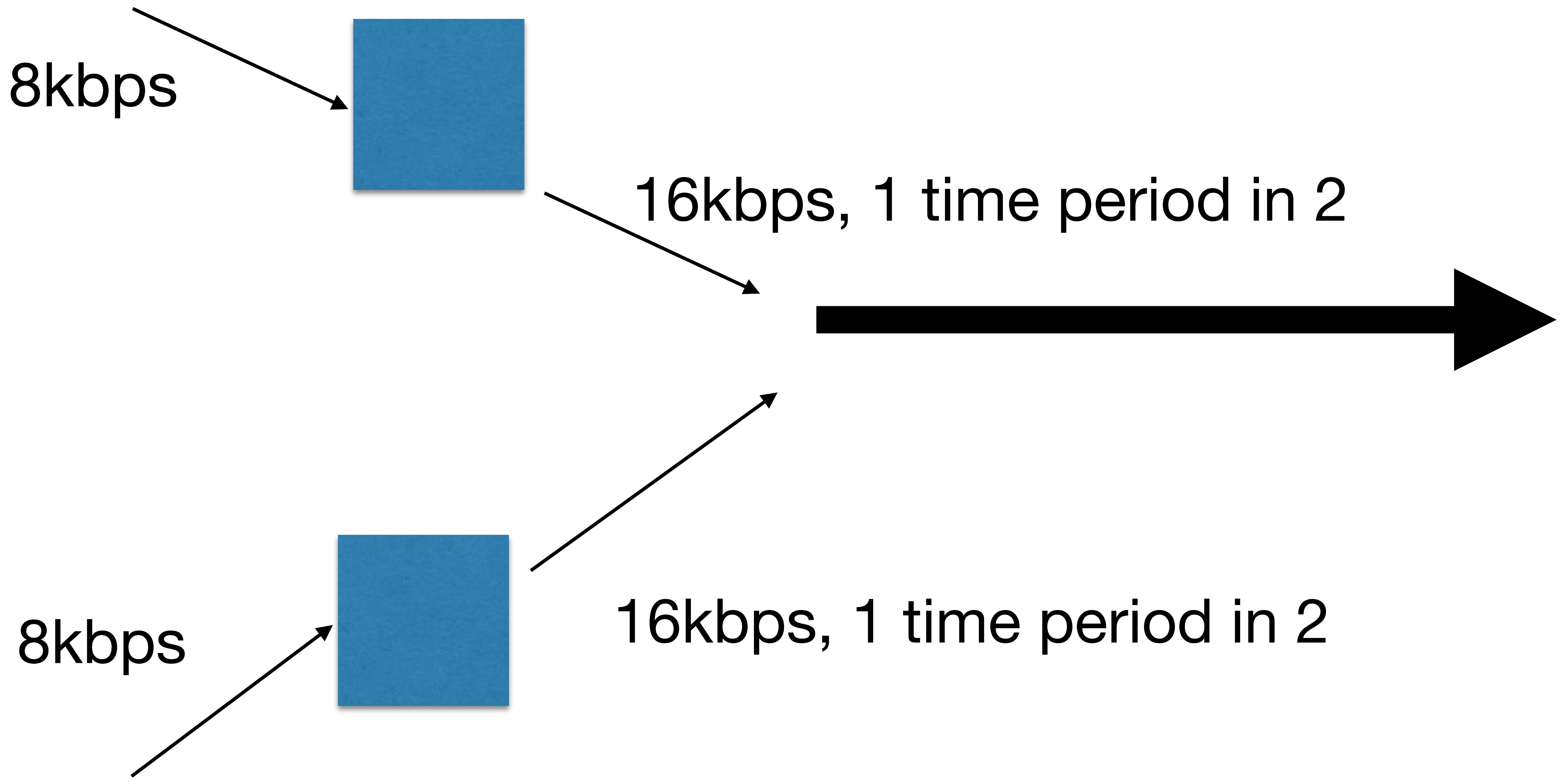
Advantages

- You are multiplexing in the **time** domain, rather than multiplexing in the **frequency** domain. It's easier to build, in general.
    - You delay the packet until there is room on the line, introducing **latency**.
- When no data is being sent, no (or at least few) resources are consumed.
- You therefore get **statistical gain** on bandwidth.
- You can re-route data around failed switches, for resilience.

Time Domain

- Suppose I have some data streams, each of 8kbps (8000 bits per second).
- I have a line which runs at 16000 bits per second
- I can send them all by buffering data arriving at 8kbps, then when the buffer is full sending it on at 16 kps (the output will be idle 50% of the time).
- I can do this a second time, using the other 50% of the line.
- There will be some latency: the first bit in each block is delayed by the time taken until the last bit arrives, and then the time taken to send the block at the higher speed.

Time Domain


8kbps
16kbps, 1 time period in 2
8kbps
16kbps, 1 time period in 2

Contrast, Frequency Domain

- I can distinguish two signals on a medium by their frequency.
  - Low and high pitched sound in air, red and blue light in fibre, 96MHz and 98MHz radio signals in a vacuum or on a wire.
  - I can modulate that signal (or “carrier”) with some data by varying the amplitude (“AM”), the frequency (“FM”), the width of some pulses (“PWM”).
  - On the receiving side I separate the two carriers, and then decode the modulation.

Frequency Domain

- Imagine two lighthouses flashing messages in morse, one in red, one in green, close enough you can't visually separate the lights.
- You would be able to distinguish the two different messages by using appropriate filters on your telescope.

FDM v TDM

- Doesn’t introduce systematic latency (which is why, amongst other reasons, Radio 4 on FM runs ahead of Radio 4 on DAB).
- But is harder to engineer and, for physics reasons (“sidebands”, if you’re interested), the use of available capacity is less efficient with real signals.
- All the networking we’ll be talking about, apart from at the transmission level, is time-domain.

Statistical Gain

- Not many people hammer full line rate 24x7: people pause to sleep, catch breath, etc.
- Even hammering at full line rate is limited by the other end and other network delays.
- So you can sell 100 people 10Mbps each, and only provision (100-x)% of (100 x 10M), x in [0, 100).
- Typically, for residential, x might be as high as 95 (ie, you only provision 5% of the aggregate bandwidth), and has been as high as 98 (2%, 1:50).
- Also called “contention ratio”, “overbooking”, “over commitment”.

Problems

- Packets can get lost, delayed, re-ordered.
- You (usually) have to break data up into packets and stick them together when they arrive

Presentation

- The carrier network, the service you buy in to convey your data, can offer a variety of services.
- Let’s simplify wildly and look just at **virtual circuits** and **datagrams**.
  - Amusingly, the telegram, from which datagrams take their name, hasn’t been available commercially for decades. I have never seen or sent one, and I don’t think my parents (born 1935) have either.

Virtual Circuits

- Endpoints tell the network to establish a connection
- The network sets up a path to the destination, and gives back some token to identify it
- For the duration, that token identifies a “virtual circuit” linking two endpoints
- It is then torn down when the endpoint has finished with it
- Network has to know about every connection in progress
- Each packet in a connection follows the same route
- Network tries to sort out ordering and packet loss/duplication, but doesn’t always guarantee it
- The user of the circuit doesn’t have to worry about then fine details, but a checksum would be a good idea once in a while.

Datagram Services

- Each packet contains complete addressing information
- Each packet is considered as a separate item by the network (conceptually, at least: more later)
- Endpoints are responsible for dealing with issues of loss, duplication, corruption: your packet might get delivered at some point, that is all you know.
- Network doesn’t (need to) know about connections: it just routes packets

Netheads v Bellheads

- “Bellheads” (people who work for telcos) like virtual circuits: they can shape and groom traffic, provide added value, run complex and interesting protocols
- “Netheads” (people who go to IETF meetings) like datagram services: it stops the telco from shaping, grooming…
- Last thirty years are a history of conflict between the two camps.

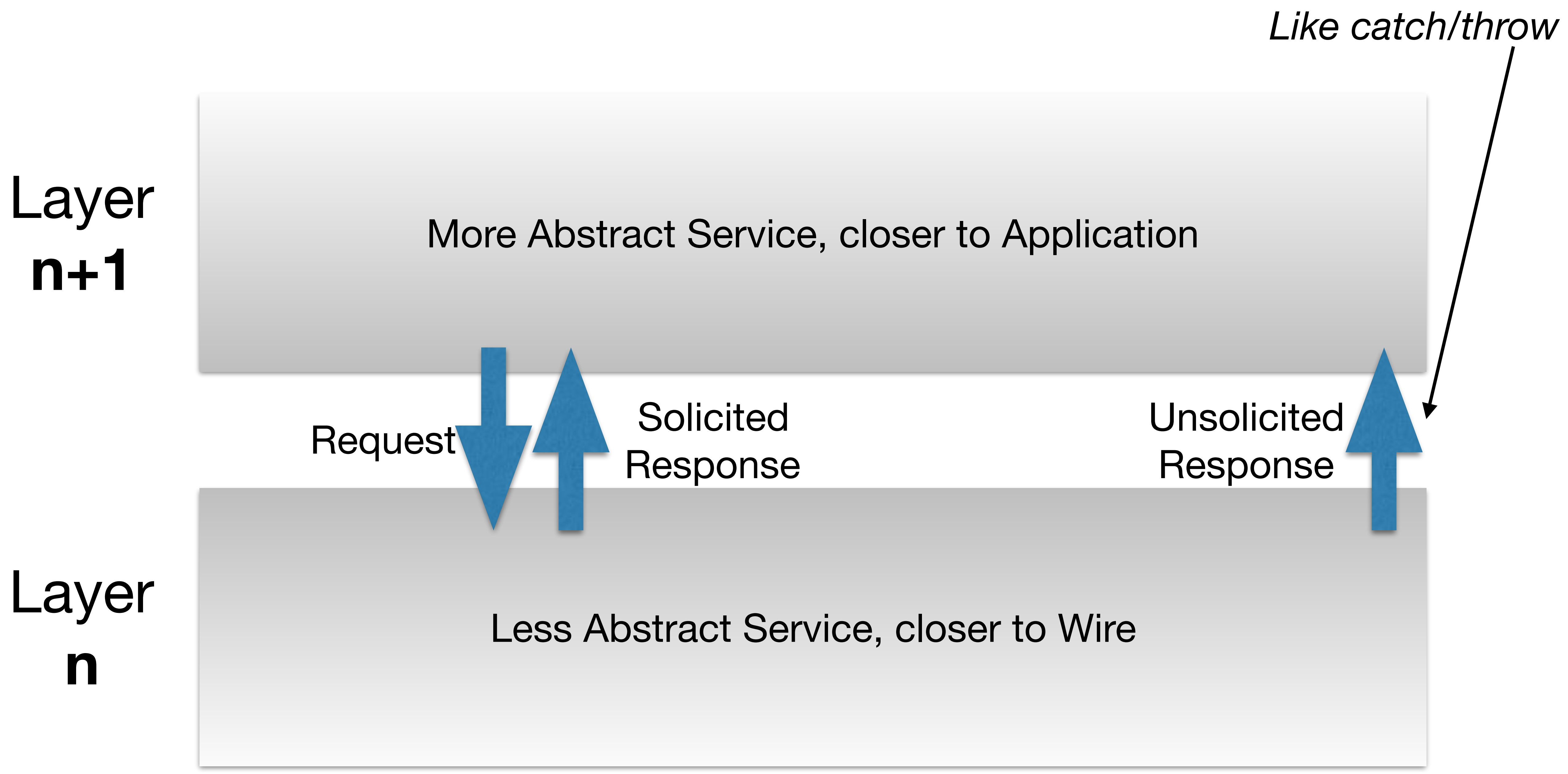
Layering

- What I’ve just described is the very basic services made available by networks: virtual circuits and datagrams.
- But applications generally want something with guarantees: you send a file, it gets there undamaged or you know something went wrong.
- And you want your programs to be able to operate over different sorts of network without radical changes.

Layering

- So the idea arises of thinking of a network in terms of “layers” or “a stack”: a succession of interfaces which start with the services needed by real applications, and progressively get closer and closer to volts and flashing laser diodes.
- Each layer provides services to those above it, and makes use of services from those below it. And each task only appears in one layer.

Layers


Like catch/throw
Layer n+1
More Abstract Service, closer to Application
Request
Solicited Response
Unsolicited Response
Layer n
Less Abstract Service, closer to Wire

Anything that obeys the requests can be used as a replacement

Competing Models

- “OSI” (Open Systems Interconnect) was a failed project, mostly European, to build a standard suite of networking protocols from within the telecommunications community. We will look later at why it failed. It was competing with proprietary systems and with…
- … the DoD, aka ARPA, aka TCP/IP suite that we will be using as our main case study

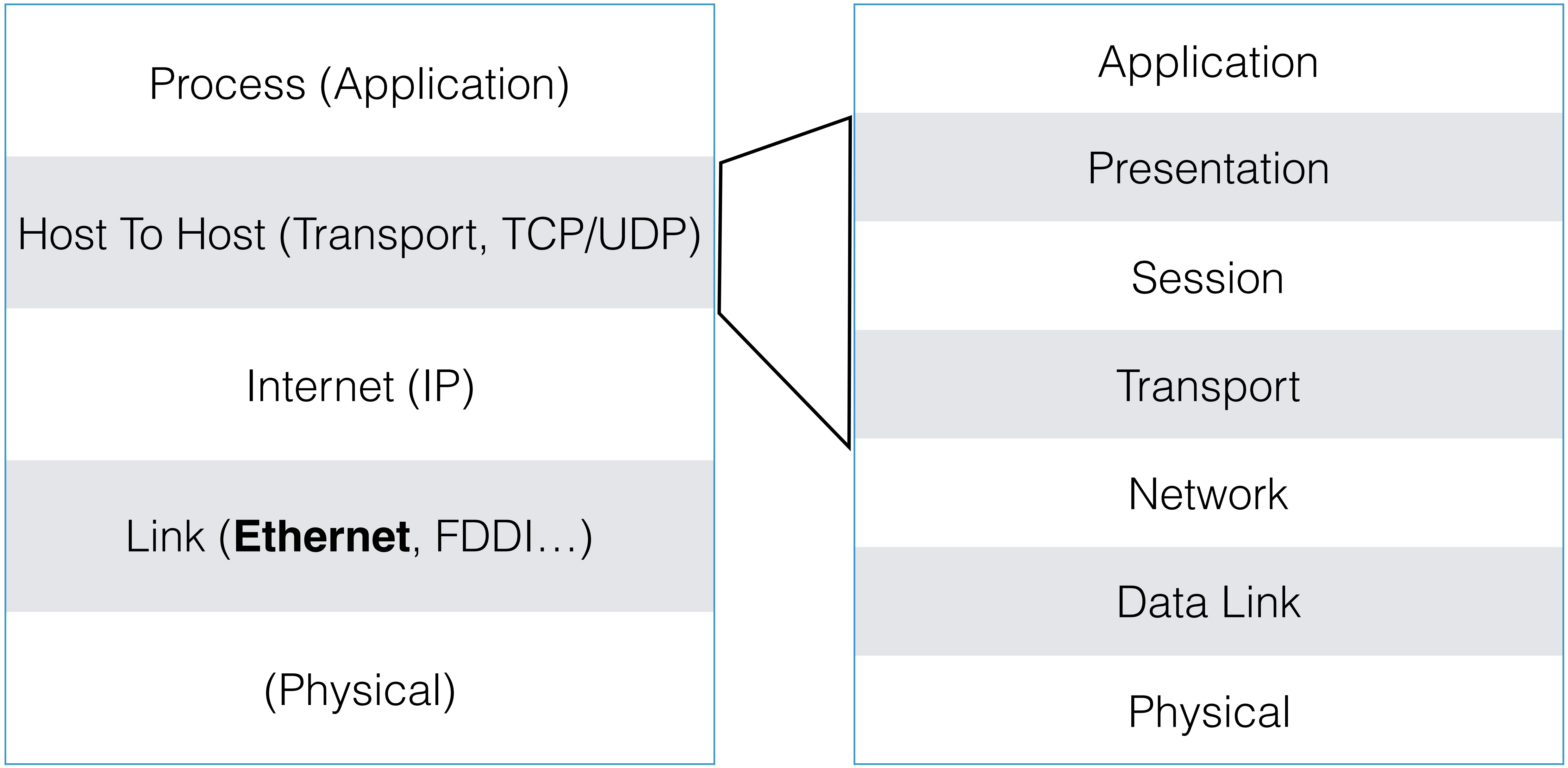
OSI Model

- The model came long before any successful implementation.
    - A ~~cynic~~ historian would argue there were never any successful implementations.
- But the model achieved widespread traction as the model of how computer networks either *should* or *do* work.

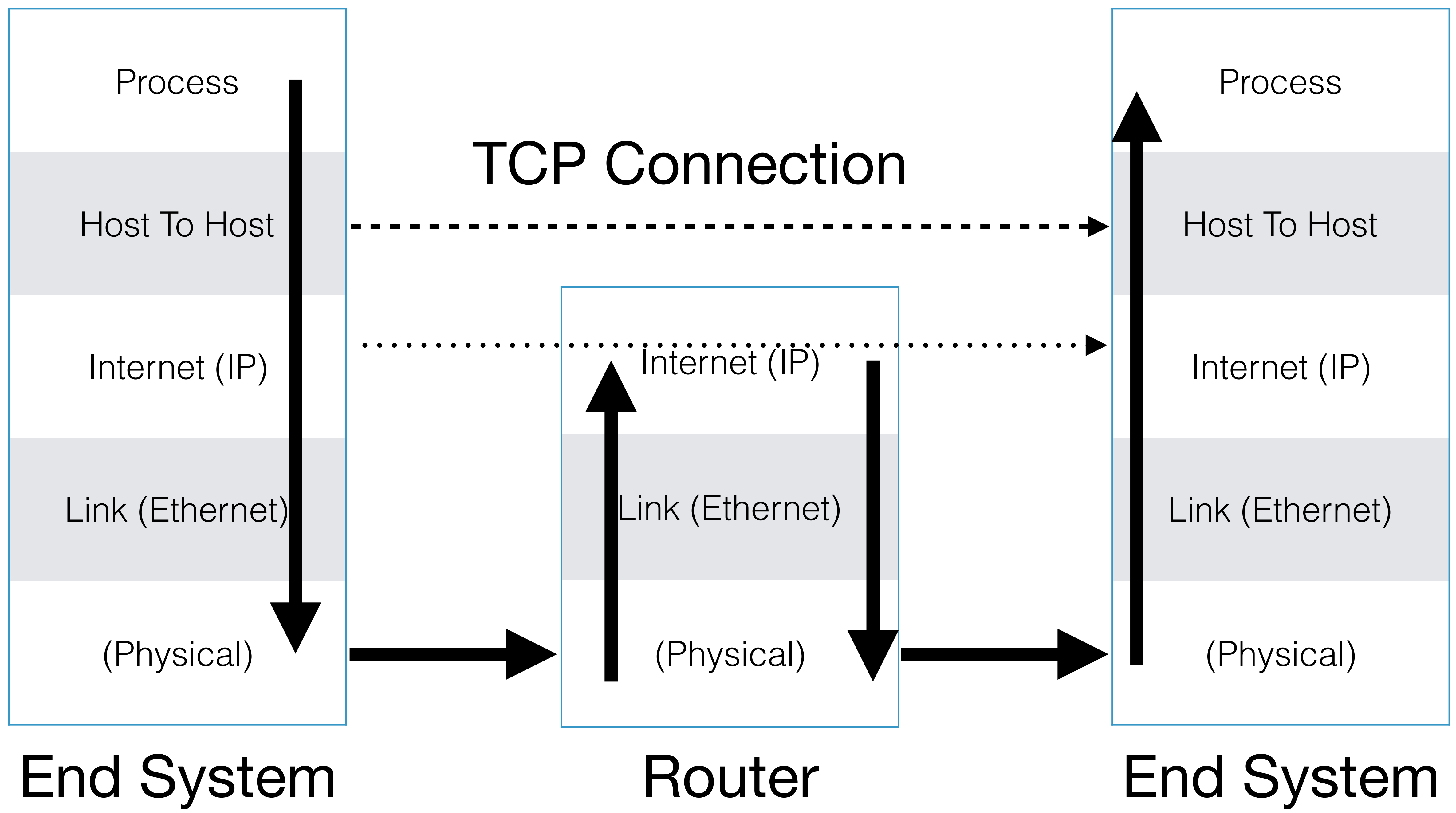
DoD Model

- The DoD model is a post-hoc rationalisation of established practice, in part in order to provide a way to compare the TCP/IP architecture with the OSI proposals.
- Practice and implementation came first, the model long afterwards.

DoD v OSI


Process (Application)
Host To Host (Transport, TCP/UDP)
Internet (IP)
Link (**Ethernet**, FDDI…)
(Physical)
Application
Presentation
Session
Transport
Network
Data Link
Physical

The DoD Model



The DoD Model: Applications

- Applications are running code that do real, useful work, and the protocols that they use.
  - SMTP and IMAP for mail, HTTP for web, ssh for remote logon…
- They need services to move data from computer to computer.

The DoD Model: Transport

- A transport layer moves data between two end systems via a network whose topology and other properties the transport layer doesn’t know much about.
  - TCP for streams of data, UDP for packets
- The transport layer will guarantee various properties: reliability, sequenced delivery, etc, etc (or will disclaim responsibility for these things).
- The transport layer will permit communication between multiple entities (applications, usually) running on the same end systems.

The DoD Model: Internet / Subnet / Network Layer

- This layer moves packets between end systems, but doesn’t offer any guarantees about reliability. It handles choosing which link to use to get the data closer to its destination.
- This layer also doesn’t deal with any concept of multiple applications: separating the data between two different applications is the layer above’s responsibility
    - IPv4, IPv6

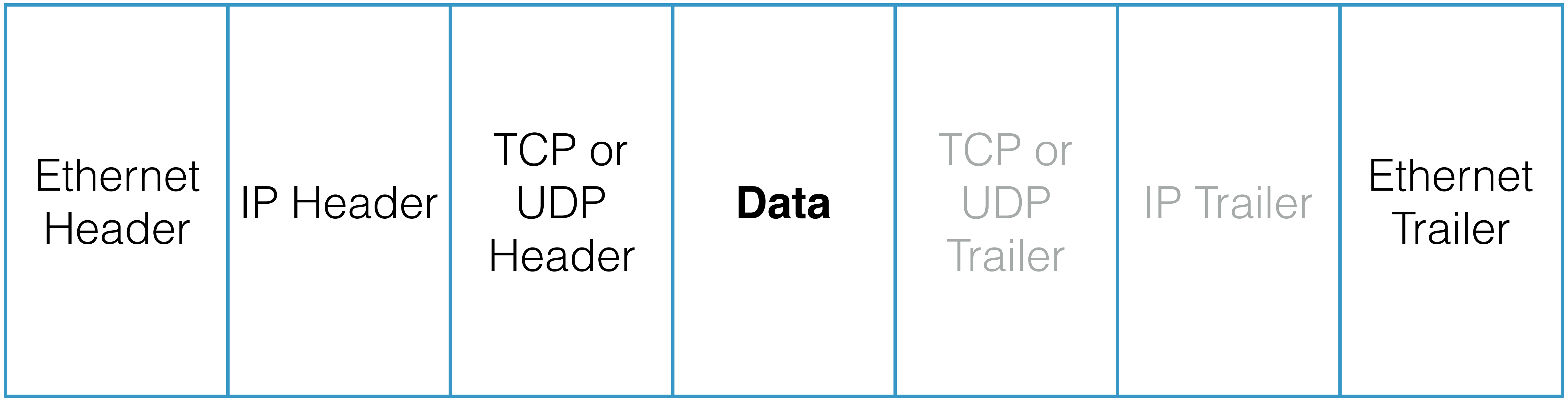
The DoD Model: Link Layer

- This layer moves data between one network element and the "next" element. This layer is concerned with packetisation, addressing and so on.
- There may be a variety of protocols in use
  - Ethernet is going to be our main focus, but ATM will sometimes still appear

(Physical Layer)

- This is the actual encoding used to shift bits over a distance. Ethernet can run over co-axial cable (if you are old), twisted pair, fibre optics of various sorts, radio…

Layers are Encapsulated in Packets



The OSI Model: extras

- The OSI model's main difference is that there are three layers where the DoD has one: transport's equivalent is presentation, session, transport.
- Presentation is intended to handle data encoding (converting integers for a neutral format everyone can use, for example).
- Session is meant to handle the relationship between multiple connections (say, restarting a failed file transfer)
- Transport is transport.
    - The DoD model pushes all the above up into libraries
- Experience and history says the extra layers are useless: applications require things that are too specific to provide as generic services. But that's another lecture.

Abstraction is **hard**

- The problems come when a layer closer to the application is tasked with providing a service which is a poor fit to the lower layers (like trying to write operating systems in Haskell).
- If your network just does datagrams, a strong transport service is hard (and will take us three/four lectures to describe)
- Conversely, if your network just does virtual circuits, sending a single datagram is very expensive.

TCP/IP Wins…

- Because there is one transport service for connections and one transport service for datagrams
  - TCP/IP has various experimental transports for special purposes, but they are not widely deployed.
- It is the responsibility of the implementor to make TCP and UDP work, in full, over whatever lower layers they are proposing.

TCP/IP wins…

- Because there is exactly one network layer, IP.
  - IPv4 and IPv6 for these purposes differ only in address length.
- It is the responsibility of physical and link layers to carry IP, and if they can’t, they can’t carry TCP or UDP.

OSI loses…

- Because it tried to be “efficient” and provided multiple transport services, ranging from a very thin one to put on top of virtual circuits (“TP0”) to a very complex one for datagrams (“TP4”).
- Because it was about telcos protecting existing business models and capital plant
- So there were **two** different network layers, CONS for connection-orientated (virtual circuit) services, and CLNS for connection-less (datagram) services. This was to keep telcos with different infrastructures happy.
- None of it interworked.

The Value Chain

- Telcos (largely) sell a service with a service level agreement
- ISPs (again, largely) are much more “best efforts”
  - They are relying on statistical gain, which works most of the time, but in Selly Oak, not so much

So…

Applications you can make money out of: Facebook (ads), NetFlix (subs), Online Banking (drives other business)	
Sale of Internet Connections to consumers and businesses, with large amounts of statistical gain and vague SLAs	Consumer ISPs
Carrying the data belonging to ISPs over a distance: they want an SLA, and only rent bandwidth they need after statistical gain	Telcos and Interconnect
Carrying data on single-haul links between premises, where you can't use any statistical gain as it's about wires and linecards.	Telcos

Equipment is Layered

- In two different ways
  - Different pieces of equipment do different jobs in the stack (although the distinctions are becoming blurred)
  - There are aspects of different layers in the same piece of equipment, and separating those makes design, construction and security easier.

Planes within Elements

- You can view equipment as having a management “plane”, a control “plane” and a traffic (or data) “plane”.
- This is telecoms language, and originally these were literally separate elements in the hardware, “plane” being jargon for one printed circuit board in a rack linked by some interconnect.

Planes

Management plane: where the GUI/CLI runs, where statistics and error reporting happens, where the device is configured. Implemented in software, running today on some general purpose operating system, typically Linux, or something similar.

Control plane: where decisions about routing policy are made. In voice switches (which we aren't going to talk about much) this is where calls are set up and cleared down. Almost always software, but might be running on a real-time executive.

Data plane: where the actual traffic is shipped. Might be done with special purpose hardware, might be done with exotic software running on exotic hardware ("network processors")

Different Hardware

- We can divide network hardware up by which part of the model they implement.
- “**Switches**” (hubs, bridges) understand the **Link Layer**. So an ethernet switch (we will talk about what makes it a switch later) switches ethernet packets between links. The interfaces are in some sense the same (might be different speeds, might be different media).
- “**Routers**” understand the **IP Layer** and move packets between potentially very different links.
- “**Hosts**” understand the **transport layer** and **application layer**, and are usually general purpose computers.

Blurring

- Hosts in fact understand everything, and make pretty good routers as well.
- These days, routers tend to have a switch integrated into them.
- And switches are increasingly “L3 aware” and make switching decisions based in part on the contents of IP headers.
- This is not exact science.